CHALMERS
UNIVERSITY OF TECHNOLOGY

Reinforcement Learning-based Model Predictive Control for Autonomous Truck Driving

A Hybrid Method for Highway Driving with Long Horizon Performance

Master's thesis in Systems, Control and Mechatronics

Hooman Hamze & Benjamin Houshmand

DEPARTMENT OF ELECTRICAL ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Reinforcement Learning-based Model Predictive Control for Autonomous Truck Driving

A Hybrid Method for Highway Driving with Long Horizon Performance

HOOMAN HAMZE

BENJAMIN HOUSHMAND

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Department of Computer Science and Engineering
Division of Data Science and AI
Chalmers University of Technology
Gothenburg, Sweden 2026

Reinforcement Learning-based Model Predictive Control for Autonomous Truck Driving
A Hybrid Method for Highway Driving with Long Horizon Performance
HOOMAN HAMZE
BENJAMIN HOUSHMAND

Supervisors:
Erik Börve, Volvo Group
Deepthi Pathare, Volvo Group

Examiners:
Nikolce Murgovski, Department of Electrical Engineering
Morteza Haghir Chehreghani, Department of Computer Science and Engineering

Master's Thesis 2026
Department of Electrical Engineering - Division of Systems and Control
Department of Computer Science and Engineering - Division of Data Science and AI
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: AI generated drawing of an autonomous truck driving on a highway among other vehicles.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Reinforcement Learning-based Model Predictive Control for Autonomous Truck Driving
A Hybrid Method with the Focus on Driving an Autonomous Truck Under Constrained Operating Conditions and Improving Long Horizon Performance
HOOMAN HAMZE
BENJAMIN HOUSHMAND
Department of Electrical Engineering
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

Autonomous vehicles have gained significant attention in recent years due to their potential to improve transportation safety, efficiency, and operational flexibility. In addition to increasing driver comfort and reducing human error, autonomous driving systems can enable continuous operation and improve fuel efficiency, which is particularly important for heavy-duty vehicles such as trucks. However, autonomous truck control remains a challenging problem due to complex traffic interactions, strict safety requirements, and the need for long-term decision-making under computational constraints.

This thesis investigates a hybrid control framework that combines Model Predictive Control (MPC) with Reinforcement Learning (RL) for autonomous truck control in highway driving scenarios. The proposed approach aims to exploit the safety, interpretability, and constraint-handling capabilities of MPC while benefiting from the long-term prediction and computational efficiency provided by RL-based value approximation methods.

We implement an MPC controller with various safety and operational constraints for driving in a simulated highway environment. A Deep Q-Network (DQN) based RL model is trained to estimate the value function, providing a faster and more accurate approximation of terminal cost. We also compare the performance of proposed framework with various variants such as rollout-based approximations. Furthermore, uncertainties and disturbances in the traffic environment are incorporated to investigate the robustness of the proposed framework under noisy and unpredictable conditions.

The results demonstrate that the proposed RL-MPC framework can successfully satisfy safety and operational constraints in different driving scenarios while improving long-term prediction and reducing overall control cost. The study further highlights the importance of terminal cost approximation and uncertainty handling in achieving robust and computationally efficient autonomous truck control.

Keywords: Autonomous Driving, Model Predictive Control, Reinforcement Learning, Rollout, Stochastic MPC, Dynamic Programming, Heavy Duty Vehicle

Acknowledgements

We would like to express our sincere gratitude to everyone who supported and contributed to this thesis throughout its development.
First and foremost, we would like to sincerely thank our supervisors at Volvo Group, Erik Börve and Deepthi Pathare, for their continuous guidance, valuable feedback, and support during the course of this project. Their technical insights, discussions, and encouragement were essential throughout the research and development process.
We would also like to express our gratitude to our examiners, Nikolce Murgovski and Morteza Haghir Chehreghani, for their valuable comments, academic guidance, and constructive feedback, which helped improve the quality of this work.
Special thanks are extended to Mohamed Abrash for his helpful advices, insightful discussions, and support during different stages of the thesis.
In addition, we would like to thank all members of the Efficient Driving team at Volvo Group, as well as their manager, Dania Badeie, for creating a supportive and collaborative working environment and for sharing their knowledge and experience throughout this project.
Finally, and most importantly, we would like to express our deepest gratitude to our families for their endless support, encouragement, patience, and understanding throughout our studies and during the completion of this thesis. Their constant support has been invaluable and made this achievement possible.

HOOMAN HAMZE, Gothenburg, June 2026
BENJAMIN HOUSHMAND, Gothenburg, June 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AC^{2}MPC	Actor-Critic Reinforcement Learning Compensated Model Predictive Control
ADP	Approximate Dynamic Programming
CCP	Chance-Constrained Programming
CDF	Cumulative Distribution Function
DP	Dynamic Programming
DQN	Deep Q-Networks
EKF	Extended Kalman Filter
erf	Error Function
FHOCP	Finite Horizon Optimal Control Problem
I.I.D	Independent and Identically Distributed
LTI	Linear Time-Invariant
MDP	Markov Decision Process
MIMO	Multi-Input Multi-Output
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
OCP	Optimal Control Problem
PDF	Probability Density Function
PPO	Proximal Policy Optimization
QP	Quadratic Programming
RL	Reinforcement Learning
SMPC	Stochastic Model Predictive Control
TCOP	Total Cost of Operation
V2X	Vehicle-to-Everything
WMPC	Weights-varying Model Predictive Control

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

h	Horizon step index
H	Prediction horizon length
H_{roll}	Rollout horizon length
H_{total}	Total rollout horizon length
i	Index for surrounding vehicles
k	Discrete time step index
t	Time index

Sets

$\mathcal{A}$	Action space in the Markov Decision Process
$\mathcal{D}$	Replay buffer of stored transitions
$\mathcal{M}$	Markov Decision Process tuple
$\mathcal{P}$	State-transition probability distribution
$\mathcal{R}$	Reward function
$\mathcal{S}$	State space in the Markov Decision Process
$\mathcal{X}_h$	Collective environment snapshot at step h
$\mathcal{T}_H$	Set of predicted traffic states at horizon step H
$\mathbb{U}$	Set of admissible control inputs
$\mathbb{X}$	Set of admissible states
$\mathcal{X}_{reach}$	Reachable state space

Parameters

α	Terminal cost weighting factor
α_{lr}	Learning rate in Q-learning
γ	Discount factor
Δt	Sampling time
ϵ	Risk tolerance / exploration probability
ϵ_{start}	Initial exploration rate
ε_{end}	Final exploration rate for reinforcement learning training
θ	Parameters of the Q-network
θ^{-}	Target network parameters
μ_k	Mean of the state estimate at time step k
σ_a	Measurement noise standard deviation for acceleration
σ_s	Measurement noise standard deviation for position
$\sigma_{s,h}$	Empirical standard deviation of longitudinal position error at step h
$\sigma_{v,h}$	Empirical standard deviation of velocity error at step h
σ_v	Measurement noise standard deviation for speed
Σ_v	Measurement noise covariance matrix
Σ_w	Process noise covariance matrix
τ	Driver reaction time in SUMO's Krauss model
τ_{target}	Target network update interval
A_k	Jacobian of the system dynamics
a_{break}	Maximum deceleration of the following vehicle
a_{max}	Maximum longitudinal acceleration
a_{min}	Minimum longitudinal acceleration (maximum deceleration)
B	Mini-batch size
C_{start}	Counter threshold for active transition lock
C_{steps}	Total counter steps for lane-change and cooldown
D_0	Standstill safe distance
d_{safe}	Safe distance to surrounding vehicles
$f_{explore}$	Exploration fraction of total training steps
f_{train}	Policy network training frequency
H_k	Observation Jacobian matrix

K_k	Kalman gain matrix
N_{lanes}	Number of available lanes
N_{total}	Total training steps executed for the DQN
P_k	State covariance matrix at time step k
P_{CD}	Penalty for initiating a lane change during cooldown period
P_{col}	Collision penalty
P_{LC}	Lane changing penalty
r_{scale}	Reward scaling factor
S_{STEP}	Longitudinal position resolution discretization parameter
T_{CD}	Effective cooldown period
T_{head}	Time headway
T_{LC}	Lane change fixed duration
T_{TD}	Total lane-change and cooldown time
v_{max}	Maximum allowable velocity
v_{min}	Minimum allowable velocity
v_{ref}	Target (reference) speed
V_{STEP}	Longitudinal velocity resolution discretization parameter
w_a	Weight for acceleration cost
w_{back}	Weight for back vehicle safety cost
w_{front}	Weight for front vehicle safety cost
w_{LL}	Weight for left lane occupancy cost
w_v	Weight for speed deviation cost

Variables

$\bar{e}_s$	Mean longitudinal position prediction prediction error (bias)
$\bar{e}_{s,h}$	Mean longitudinal position prediction error (bias) at horizon step h
$\bar{e}_v$	Mean longitudinal velocity prediction prediction error
$\bar{e}_{v,h}$	Mean longitudinal velocity prediction error at horizon step h
$\pi(a\|s)$	Policy function
π^R	Rollout policy
$\tilde{\pi}$	Base policy
σ_{e_s}	Standard deviation of longitudinal position prediction error

σ_{e_v}	Standard deviation of longitudinal velocity prediction error
a_k	Longitudinal acceleration at time step k
c_k	Lane changing counter state at time step k
Δl_k	Lateral lane change command at time step k
$g(x_k, u_k)$	Stage cost function
g_{back}	Gap to the following vehicle
g_{front}	Gap to the lead vehicle
G_t	Discounted return
$J_h(x_h)$	Cost-to-go function at horizon step h
l_k	Lane position state at time step k
$l_{i,H}$	Predicted lateral lane index of surrounding vehicle i at horizon step H
len_i	Physical length of surrounding vehicle i
l_{reg}	Logical lane tracking variable
$p_{CD,H}$	Lane-change cooldown status flag at terminal horizon step H
$p_{LC,H}$	Lane-change activity status flag at terminal horizon step H
$q(x_H)$	Terminal cost function
$Q^{\pi}(s, a)$	Action-value function under policy π
$Q^{*}(s, a)$	Optimal action-value function
$Q_{\theta}(s, a)$	Approximated action-value function
r_t	Reward at time t
s_k	Front-end longitudinal position at time step k
$s_{i,H}$	Predicted longitudinal position of surrounding vehicle i at horizon step H
u_k	Control input at time step k
v_k	Measurement noise vector
v_k	Vehicle velocity state at time step k
$v_{i,H}$	Predicted longitudinal velocity of surrounding vehicle i at horizon step H
$\hat{V}(x_H)$	Approximate terminal value function
$V^{\pi}(s)$	State-value function under policy π
$V^{*}(s)$	Optimal state-value function
w_k	Process noise vector
x_k	System state vector at time step k
y_k	Observation vector at time step k

z_k	Agent observation state vector for reinforcement learning
z_H	Reinforcement learning observation state vector at horizon step H

Contents

List of Figures

List of Tables

1
Introduction

1.1 Motivation

Autonomous driving has experienced rapid progress in recent years, driven by advances in sensing technologies, embedded computing, and machine learning. Despite these developments, achieving safe, efficient, and comfortable vehicle motion in complex traffic environments remains a challenging control problem. This challenge is particularly prominent for autonomous trucks, which impose stricter operational constraints due to their larger mass, longer braking distances, and the increased severity of potential accidents. Beyond safety, the Total Cost of Operation (TCOP) incorporating factors such as energy consumption, break wear, and even passenger comfort remains a critical metric for the industrial viability of heavy duty transport [1, 2]. As a result, truck motion control systems must exhibit both short-term responsiveness and long-term predictive capability.

Model Predictive Control (MPC) has emerged as one of the most successful frameworks for autonomous vehicle motion control due to its ability to explicitly incorporate system dynamics, safety constraints, and actuator limitations within an optimization-based formulation. MPC has been widely applied to car following, lane keeping, lane changing, and trajectory tracking tasks in autonomous driving [3]. By solving a constrained finite-horizon optimal control problem in a receding horizon manner, MPC generates control actions that optimize objectives in near future while guaranteeing constraint satisfaction.

However, a fundamental limitation of conventional MPC lies in its reliance on relatively short horizon prediction, which restricts its ability to reason about long-term consequences. In dense and dynamic highway environments, this often leads to conservative, suboptimal, or even unsafe behavior, as the controller cannot fully account for the strategic interactions with other vehicles or distal goals [4]. While increasing the MPC prediction horizon can partially mitigate this limitation, doing so significantly increases the computational burden, and often makes real-time implementation on embedded hardware infeasible [5]. Furthermore, long horizon optimization alone cannot fully capture the stochastic nature of traffic, interactions with human-driven vehicles, and generally modeling uncertainties [6, 7].

These challenges motivate the integration of learning-based techniques with MPC to improve long term performance while preserving safety and computational feasibility. By utilizing Reinforcement Learning (RL) to provide a terminal value estimate, the effective planning horizon of MPC can be extended, enabling low-level motion control to better account for long-term tactical objectives [8, 9, 10]. This hybrid

approach allows the control architecture to inherit the interpretability and robustness of predictive control while gaining the adaptive, long-term foresight of deep learning.

1.2 Related Works

1.2.1 Model Predictive Control for Autonomous Driving

MPC has become the key methodology of autonomous vehicle motion planning and control due to its ability to explicitly handle multi-variable system dynamics and state/input constraints [11]. In the context of highway driving, MPC is frequently used for trajectory generation and tracking, as it can be optimized for safety, passenger comfort, and fuel efficiency within a receding horizon framework.
Early applications of MPC in this domain often relied on linear time-invariant (LTI) models to ensure real-time feasibility. However, to capture the highly nonlinear nature of vehicle dynamics, Nonlinear MPC (NMPC) has been increasingly adopted. For instance, recent work by Börve et al. [4] utilizes NMPC within a distributed framework to solve interaction-aware trajectory planning in dense highway traffic. Their approach couples prediction and planning, allowing the ego-vehicle to account for the future trajectories of surrounding agents, thereby improving safety and coordination in tight merging or lane-changing scenarios.
Despite its strengths, the performance of MPC is heavily dependent on the accuracy of the underlying vehicle model, prediction model, optimization horizon, and the tuning of its cost function parameters. Recent work has explored combining learning-based and optimization-based approaches to leverage the strengths of both paradigms. For example, Brito et al. [12] proposed a hybrid framework in which a reinforcement-learning policy provides interaction-aware guidance to an MPC-based trajectory optimizer, achieving improved performance in dense traffic scenarios while preserving safety constraints. Furthermore, the selection of the prediction horizon remains a critical trade-off, while a longer horizon provides better foresight, it significantly increases the computational burden, often requiring specialized solvers or architectural simplifications to remain viable for real-time deployment.

1.2.2 Reinforcement Learning for Autonomous Driving

RL has emerged as a powerful paradigm for autonomous driving, offering the ability to learn complex driving strategies directly from data and environmental interaction [13]. Early research in this area often focused on end-to-end learning, where deep neural networks map raw sensor inputs, such as images, directly to low-level control commands like steering and acceleration [14]. While these approaches demonstrate the potential for capturing human-like driving behavior, they often struggle with safety guarantees and the massive data requirements needed to handle special cases in real-world environments.
To mitigate these challenges, recent literature has shifted toward utilizing RL for high-level tactical decision-making, while leaving low-level execution to traditional controllers. Pathare et al. [2] demonstrated such an integration, pairing RL-based

tactical decision-making with low-level controllers for execution, and showed that separating high-level tactical processes from low-level physical models improves both safety and overall driving efficiency. Within the same paradigm, Hoel et al. [15] demonstrated the effectiveness of Deep Q-Networks (DQN) in managing speed and lane-change decisions for heavy-duty vehicles, showing that RL agents can match or outperform heuristic-based reference models in highway scenarios. Further advancements have addressed the "black-box" nature of RL by incorporating uncertainty estimation. By employing a set of neural networks with randomized prior functions, agents can quantify their confidence in specific actions, allowing the system to fall back to a safe state when encountering out-of-distribution scenarios [6]. Despite these successes, the transition from discrete high-level actions to continuous, safe control remains a primary area of investigation.

1.2.3 Hybrid Model Predictive Control and Reinforcement Learning Approaches

The integration of MPC and RL has emerged as a dominant research direction to overcome the individual limitations of each method. While MPC provides safety guarantees through explicit constraint handling and a physics-based model, it often suffers from model inaccuracies and short-sightedness due to limited horizons. Conversely, RL can learn complex behaviors and account for long-term rewards but lacks inherent safety mechanisms and transparency. Recent research explores various hybrid architectures to create "Safe RL" or "Learning-based MPC" frameworks.

One prominent hybrid approach involves using RL to dynamically adapt the parameters of an MPC controller. Zarrouki et al. [10] proposed a Weights-varying MPC (WMPC) where an RL agent selects optimal cost function weights from a pre-optimized Pareto set, allowing the controller to adapt to changing environments while remaining in a stable operating space. Similarly, Chen et al. [9] utilized Proximal Policy Optimization (PPO) to adjust the MPC prediction horizon online, significantly improving trajectory tracking performance across varying speeds and curvatures. In car-following scenarios, Wang et al. [16] demonstrated that RL can be used to extract features from dynamic traffic scenes to adjust weight coefficients, improving adaptability to real-world risk levels.

Another research direction uses RL to compensate for unmodeled dynamics within the MPC framework. Gupta et al. [7] introduced the Actor-Critic Reinforcement Learning Compensated MPC (AC^2MPC) for off-road driving, where the MPC handles the nominal vehicle model and the RL agent learns to compensate for complex tire-terrain interactions. Similarly, Li et al. [17] proposed using a low-dimensional residual model to learn deviations in vehicle behavior and incorporate these learned residuals directly back into the MPC constraints.

Hierarchical structures have also emerged as an effective way to separate tactical decision-making from low-level motion control. Reiter et al. [18] proposed a hierarchical motion planning framework in which a high-level policy generates reward specifications for a low-level NMPC controller while restricting exploration to safe and feasible trajectories. Similarly, Albarella et al. [19] developed a hybrid architecture for autonomous highway driving that combines deep reinforcement learning

for behavioral decision-making with nonlinear MPC for constrained vehicle control in dynamic traffic scenarios.
A particularly promising class of hybrid methods focuses on using RL to provide terminal value function approximations for MPC. The theoretical basis for using learned terminal costs in MPC can be interpreted through the framework of Dynamic Programming (DP). Bertsekas [20] formalizes the relationship between MPC and RL by framing both as forms of approximation in value space. Within this context, the MPC optimization serves as a finite-horizon "lookahead" policy, while RL provides approximations of the long-term cost-to-go. This theoretical connection is further expanded in [8], where hybrid MPC-RL architectures are interpreted as rollout algorithms that combine finite-horizon optimization with approximate value functions to recover infinite-horizon behavior. These approaches aim to extend the effective planning horizon without significantly increasing online computational complexity. A good estimation of terminal cost can contribute to closed-loop stability, recursive feasibility, and safe operation by accounting for the future consequences of decisions beyond the explicit prediction horizon. Therefore, learning accurate terminal value approximations may not only improve performance but also support the robustness and reliability of autonomous driving systems. Morita et al. [21] and Chintalapudi et al. [22] demonstrated that augmenting the MPC cost-to-go with a learned terminal value function allows the controller to operate with shorter real-time horizons while preserving long-term performance and decision quality. Such formulations are especially attractive for autonomous driving applications, where computational limitations often restrict the feasible prediction horizon despite the need for long-term reasoning.
Recent survey works further emphasize the broad unification between MPC and RL across these hybrid formulations. In particular, Reiter et al. [3] describe MPC as a deterministic policy derived from an online optimization problem and discuss how RL components can complement MPC through value approximation, policy adaptation, residual learning, and hierarchical decision-making. Their work highlights that learned value functions, adaptive policies, and model compensation strategies can all be interpreted within a common DP-based framework, where MPC provides structured safety and constraint handling while RL contributes long-term adaptation and learning capability. Consequently, hybrid RL-MPC methodologies represent one of the most promising directions toward robust, interpretable, and computationally efficient autonomous driving systems.

1.3 Research Gap and Problem Statement

While MPC provides safety, interpretability, and explicit constraint handling, its performance is fundamentally limited by the length of the prediction horizon. RL, conversely, can capture long-term effects but often lacks the safety guarantees and transparency required for safety-critical applications. These limitations are complementary: the long-horizon reasoning that MPC lacks is precisely what RL provides, while the safety and constraint handling that RL lacks are inherent to MPC. Hybrid RL-MPC frameworks exploit this complementarity by incorporating a value function learned through RL into the MPC optimization problem, where it serves

as a terminal cost that approximates the long-term consequences of actions beyond the prediction horizon. Recent studies have reported encouraging results for such frameworks. Nevertheless, a clear research gap remains in bringing these elements together for autonomous truck control: no existing framework combines safety, interpretability, real-time feasibility, and long-horizon decision-making within a unified architecture.

1.4 Research Questions

This thesis aims to investigate the following research questions:

1. How can RL value functions be designed to capture long-horizon cost in truck driving while ensuring compatibility with MPC's constraint handling?
2. In realistic scenarios, how does an RL-MPC framework perform compared to a standard MPC controller and to RL-only methods in terms of safety, coordination, and computational efficiency?
3. To what extent do uncertainties, such as the behavior of surrounding vehicles and the occurrence of unforeseen events, affect the performance of an RL-MPC framework, and can these effects be mitigated?
4. How should the MPC horizons and RL discount factor be balanced so that the controller is both computationally and operationally efficient?

1.5 Objectives and Contributions

The objective of this thesis is to develop, implement, and evaluate a learning-based Model Predictive Control framework for autonomous truck control in highway traffic scenarios. The proposed approach integrates a learned terminal value function into a MPC formulation, thereby extending the effective planning horizon while preserving safety, interpretability, and computational tractability. Furthermore, the effect of uncertainty on the performance of the controller is investigated by considering both process and measurement noises.

2

Theory

This chapter provides the mathematical foundation for the hybrid control framework proposed in this thesis. We begin by defining the principles of MPC and its formulation as a constrained optimization problem. Subsequently, we introduce RL and Markov Decision Process (MDP), focusing on value function approximation as a means to capture long-term system behavior.

2.1 Model Predictive Control

Model Predictive Control (MPC) represents a family of control algorithms that explicitly use a process model to predict the future evolution of a system and calculate an optimal control signal. Since its inception in the late 1970s, MPC has evolved from a tool primarily used in the chemical process industries to a dominant strategy for high-performance applications, including autonomous vehicle guidance and aerospace systems [23].

2.1.1 Motivation and Advantages

The widespread adoption of MPC is driven by several key advantages that distinguish it from classical control methods like Proportional-Integral-Derivative (PID) control or Linear Quadratic Regulators (LQR):

Perhaps the most significant advantage of MPC is its ability to incorporate constraints on both system states and control inputs. In the context of autonomous driving, this allows the controller to respect physical limits, such as maximum engine acceleration (a_{max}), while simultaneously maintaining safety-critical boundaries, such as minimum following distances [24].

MPC naturally supports the formulation of multi-objective optimization problems, similarly to other optimal control approaches such as LQR. However, MPC extends this capability by explicitly handling state and input constraints within the optimization problem.

Furthermore, the controller possesses foresight by utilizing a prediction horizon. It can begin adjusting control actions (e.g., decelerating) well before a constraint is reached or a future reference change occurs, rather than merely reacting to an error after it has appeared.

2.1.2 Receding Horizon Concept

The defining characteristic of MPC is the receding horizon principle. At each sampling instant, the controller solves an open-loop optimal control problem over a finite time horizon. Although an entire sequence of future control actions is calculated, only the first element is applied to the system. At the next time step, the horizon recedes or shifts forward, and the optimization is repeated using the most recent state measurements or estimates [23].
This feedback mechanism provides the system with robustness against model mismatch and external disturbances. By recalculating the trajectory at every step, the controller can correct itself for errors that occurred between the predicted behavior and the actual system response.

2.1.3 Mathematical Formulation

The operational logic of MPC is centered on solving a Finite Horizon Optimal Control Problem (FHOCP) at each discrete time step k. To do so, the system dynamics are represented by a discrete-time model:

$$x_{k+1} = f(x_k, u_k) \tag{2.1}$$

where $x_k \in \mathbb{R}^n$ is the state vector and $u_k \in \mathbb{R}^m$ is the control input.
The controller seeks to minimize a cost function J over a prediction horizon H. This cost function is generally structured as a summation of stage costs $g(x, u)$ and a terminal cost $q(x_H)$ [11]:

$$\min_{u_{0\dots H-1}} J = \sum_{h=0}^{H-1} g(x_{k+h}, u_{k+h}) + q(x_{k+H}) \tag{2.2}$$

subject to:

$$x_{k+h+1} = f(x_{k+h}, u_{k+h}) \tag{2.3}$$
$$u_{k+h} \in \mathbb{U} \tag{2.4}$$
$$x_{k+h} \in \mathbb{X} \tag{2.5}$$

Here, $\mathbb{U}$ and $\mathbb{X}$ represent the sets of admissible control inputs and states, respectively. These sets define the physical and safety constraints of the system.

2.1.4 The effect of Terminal Cost

The performance of the MPC controller is critically sensitive to the choice of terminal cost $q(x_H)$. Although infinite-horizon optimal control provides the theoretically ideal solution, solving such problems online is generally computationally intractable for autonomous driving applications. MPC therefore approximates the infinite-horizon problem using a finite prediction horizon.
The terminal cost serves as a surrogate representation of all costs occurring beyond the prediction horizon. Without a terminal cost, the controller implicitly assumes that no future cost exists after the horizon. Setting $q(x_H) = 0$ is the simplest choice

and implies no cost beyond the horizon, an assumption that causes myopic behavior in dynamic environments.
From control theory perspective, terminal costs are not only introduced to improve performance. Properly designed terminal costs contribute to closed-loop stability, recursive feasibility, and safe operation by encouraging the optimizer to steer the system toward desirable terminal states. Choosing $q(x_H)$ as an approximation of the infinite-horizon cost-to-go, denoted $\hat{V}(x_H) \approx V^*(x_H)$, extends the effective planning horizon without increasing the computational cost of the optimization [8, 11, 20, 21, 22]. This observation motivates the hybrid MPC-RL architecture proposed in this thesis, where a learned value function provides the terminal cost. The central idea of this thesis is that reinforcement learning can be used to approximate this terminal value function, thereby extending the effective planning horizon without increasing the online optimization burden.

2.1.5 Receding Horizon Implementation

The actual implementation of the controller follows the receding horizon strategy. Although the optimization yields an optimal sequence of future control actions $U^* = \{u_k^*, u_{k+1}^*, \ldots, u_{k+H-1}^*\}$, only the first action u_k^* is applied to the system.
At the next sampling interval $k+1$, a new measurement of the state x_{k+1} is obtained, and the entire optimization process is repeated over a shifted horizon. This repetitive optimization introduces a feedback loop into the system, which allows the MPC to compensate for disturbances and modeling inaccuracies that were not accounted for in the initial prediction.
This approach transforms the control problem into a sequence of optimization problems that can be solved using various numerical techniques, from quadratic programming (QP) for linear systems to dynamic programming (DP) or nonlinear optimization for more complex, high-dimensional environments.

2.1.6 Dynamic Programming and the Backward Sweep

The finite-horizon optimal control problem defined in Section 2.1 can be solved exactly via Dynamic Programming (DP), a general-purpose method for solving sequential decision problems introduced by Bellman [25]. DP exploits the Principle of Optimality, which means an optimal control sequence has the property that, regardless of the initial state and the first decision, the remaining decisions must constitute an optimal policy for the sub-problem starting from the state that results from the first decision. This recursive structure converts the joint minimization over the full sequence $u_0, \ldots, u_{H-1}$ into a sequence of single-step minimizations that can be solved efficiently [26].

2.1.6.1 Backward Induction

For a finite horizon H, DP proceeds backward in time from the terminal step. The cost-to-go at each step h is defined as the optimal cumulative cost from step h to the end of the horizon:

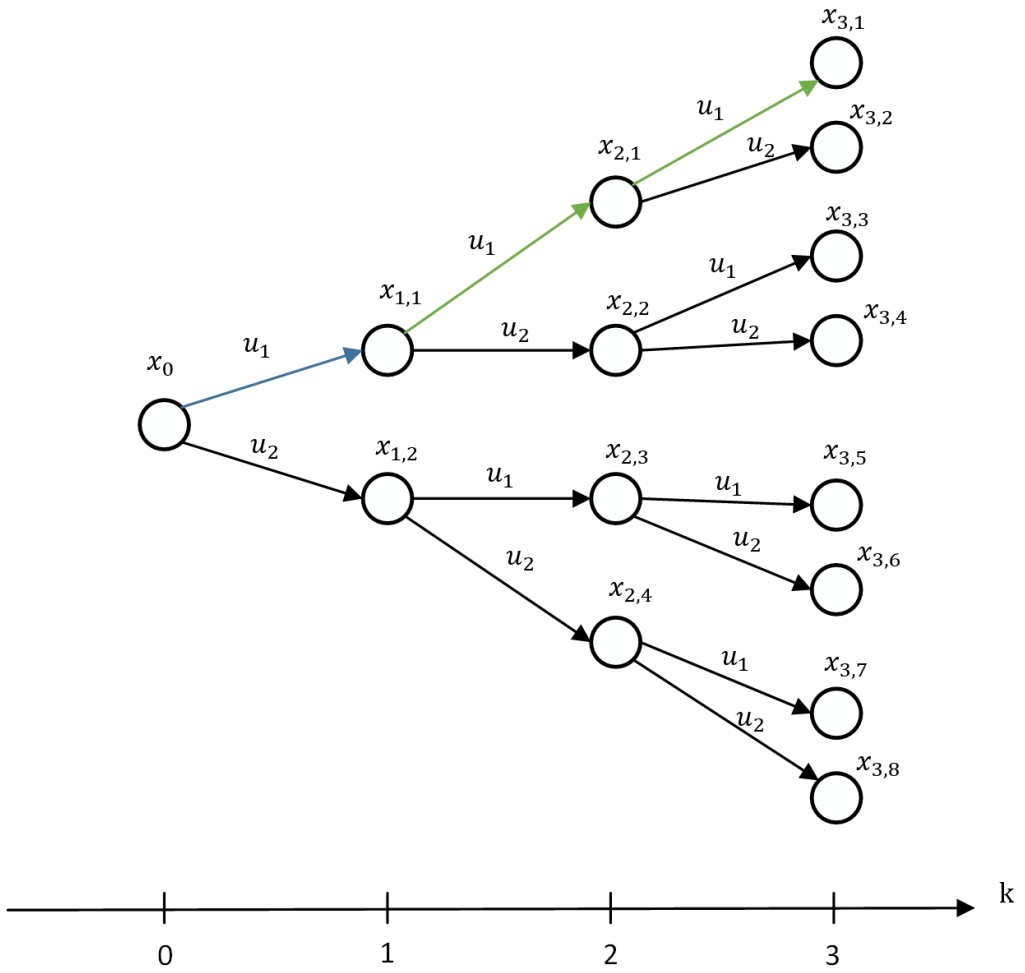


Figure 2.1: A concept of how DP solves MPC problem.

$$J_h(x_h) = \min_{u_h,\ldots,u_{H-1}} \left[\sum_{k=h}^{H-1} g(x_k, u_k) + q(x_H) \right] \tag{2.6}$$

The boundary condition initializes the value at the terminal step with the terminal cost:

$$J_H(x_H) = q(x_H) \tag{2.7}$$

The backward recursion then computes J_h from J_{h+1} by solving a one-step minimization at each step:

$$J_h(x_h) = \min_{u_h \in \mathcal{U}} \left[g(x_h, u_h) + J_{h+1}(f(x_h, u_h)) \right] \tag{2.8}$$

This is the Bellman equation for the finite-horizon problem [25, 26]. By sweeping from $h = H - 1$ back to $h = 0$, the algorithm constructs the optimal value function $J_0(x_0)$ and the associated optimal policy $u_h^* = \arg\min_{u_h} \left[g(x_h, u_h) + J_{h+1}(f(x_h, u_h)) \right]$ at every step.

2.1.6.2 Discrete State and Action Space

In the continuous-state setting, solving the Bellman recursion exactly is generally intractable. However, when the state and action spaces are discretized to finite grids, the backward sweep reduces to a table lookup over all reachable state-action pairs [26, 27]. For each grid point x_k in the discrete state space and each candidate action u_k in the discrete action set, the controller evaluates the stage cost $g(x_k, u_k)$ and looks up the next state value $J_{k+1}(f(x_k, u_k))$. The minimizing action is stored as the optimal policy at that state and step.
Once the backward sweep is complete, the optimal action at the current state x_0 is recovered by a single forward lookup.

2.2 Approximate Dynamic Programming (ADP)

While the mathematical framework of MPC is optimal, its practical application in real-time environments is often limited by the computational cost. This is especially true in highway driving, where the controller must evaluate a vast sequence of potential maneuvers within milliseconds.

2.2.1 The Curse of Dimensionality

The primary challenge in solving the FHOCP lies in the exponential growth of the search space as the prediction horizon H increases. Even when the state vector in this project is limited to the ego vehicle and does not include surrounding vehicles , the number of possible control sequences is determined by the branching factor.
For a set of discrete inputs, the total number of possible paths through the decision tree is N^H, where N is the number of available control actions. For a high-resolution control task with a long horizon, this leads to a "Curse of Dimensionality" [26]. Exhaustive search methods or standard nonlinear solvers struggle to find the global optimum for long horizons within the strict sampling times required for safe autonomous navigation.

2.2.2 Rollout Approximation

To overcome these computational difficulties, approximate dynamic programming can be used. The core idea of ADP is to solve the Bellman equation approximately rather than exactly. In a traditional DP setup, the Cost-to-Go must be calculated for every possible reachable state.
ADP replaces the exact calculation of the future cost with an approximation, often referred to as a Value Function Approximation [27]. This allows the controller to focus computational resources on the immediate lookahead horizon while using a simplified estimate for the remaining steps.
One of the most effective ADP techniques for real-time control is the Rollout Algorithm. Rollout is based on the principle of policy improvement. It uses a base policy, a heuristic or simplified control law, to estimate the future cost-to-go beyond a certain point [26].

In this implementation, the controller:

1. Performs an exact optimization (such as DP) over a short-term lookahead horizon (H).
2. At the end of this horizon, it rolls out the trajectory for a much longer total horizon (H_{total}) using a greedy base policy.

This ensures that the system makes precise tactical actions in the immediate future while still accounting for the long-term consequences without the burden of an exhaustive search over the long window.

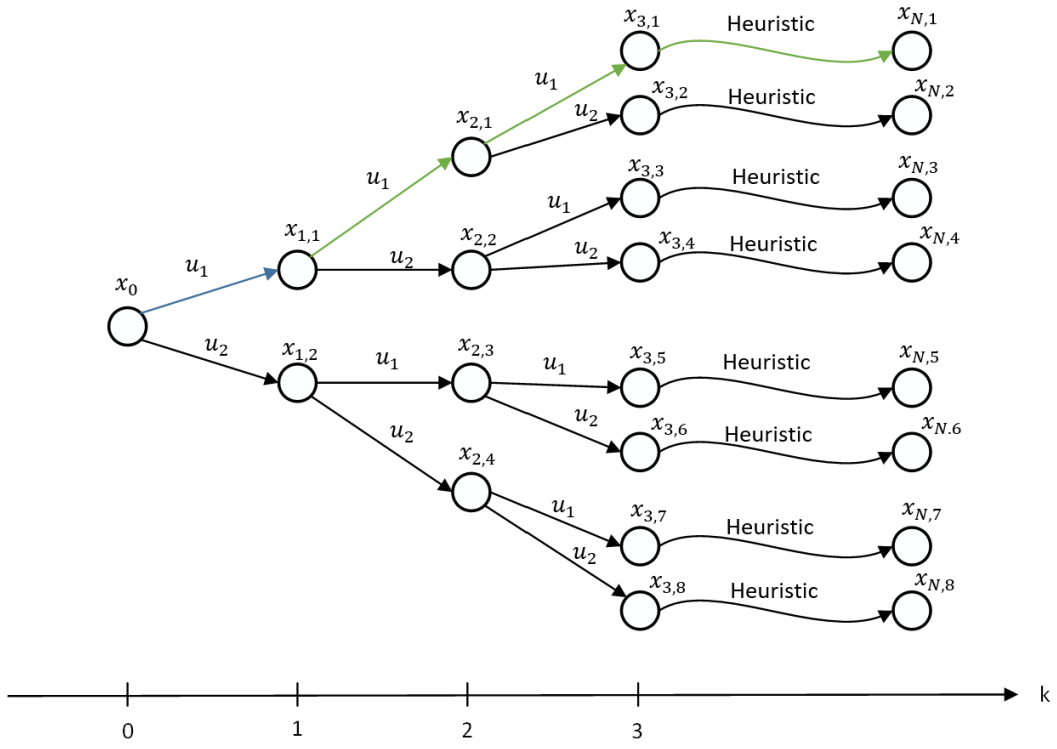


Figure 2.2: A concept of how Rollout works.

The theoretical justification for rollout is provided by Bertsekas [8, 20, 26], who shows that for any base policy $\tilde{\pi}$, the rollout policy π^R satisfies:

$$J^{\pi^R}(x) \leq J^{\tilde{\pi}}(x) \quad \forall x \tag{2.9}$$

This means that the rollout policy, which is the combination of short-term lookahead horizon and base policy, achieves a cost no worse than the base policy from every initial state. This one-step policy improvement property is the key property that justifies using a greedy rollout over a finite horizon H_{roll} as a terminal cost approximation.

By combining a short-term DP with a long-term Rollout, the controller achieves a balance between foresight and feasibility. The short-term DP ensures safety and reacts to the immediate changes, while the Rollout ensures the system follows a globally sensible strategy. This prevents the controller from becoming short-sighted

and making decisions that look optimal in the next few steps but lead to worse states or poor performance in the longer period.

2.3 Stochastic Optimal Control

The deterministic MPC formulation presented in Section 2.1 assumes perfect knowledge of the system state and exact prediction of surrounding vehicle behavior over the horizon H. In autonomous driving, neither assumption holds. Sensor measurements carry noise, and the future trajectories of other vehicles are inherently uncertain. When these uncertainties are significant relative to the safety margins, a deterministic controller may produce plans that appear optimal in the model but are unsafe in reality.
Stochastic Model Predictive Control (SMPC) extends the MPC framework to explicitly account for such uncertainties. Rather than optimizing over a single predicted trajectory, SMPC optimizes over the distribution of possible future states, incorporating uncertainty directly into the cost function and constraints [28, 29]. In the context of collision avoidance for autonomous vehicles, this leads to chance-constrained formulations that provide probabilistic safety guarantees, ensuring that the probability of a safety constraint violation remains below a desired risk tolerance [30]. The following subsections establish the mathematical foundation for this framework.

2.3.1 Stochastic Discrete-Time Systems

The foundation of modern control theory for systems operating under uncertainty is the discrete-time stochastic dynamical system. As established by [26], a nonlinear system with additive zero-mean Gaussian process and measurement noise can be written as:

$$x_{k+1} = f(x_k, u_k) + w_k, \quad w_k \sim \mathcal{N}(0, \Sigma_w) \tag{2.10}$$

$$y_k = h(x_k) + v_k, \quad v_k \sim \mathcal{N}(0, \Sigma_v) \tag{2.11}$$

where $x_k \in \mathbb{R}^n$ is the true state vector, $u_k \in \mathbb{R}^m$ is the control input, w_k is additive Gaussian process noise with covariance Σ_w representing uncertainty in the system's model, $y_k \in \mathbb{R}^p$ is the observation vector, $h(\cdot)$ is the observation model, and v_k is additive Gaussian measurement noise with covariance Σ_v representing sensor imperfections. The two noise sources are assumed to be independent (i.i.d.) of each other across time steps.
In the highway driving context, process noise w_k captures prediction errors arising from unmodeled dynamics of surrounding vehicles, for example, a vehicle braking unexpectedly or changing lanes during the prediction horizon. Measurement noise v_k captures sensor uncertainty in the observed states of surrounding vehicles, such as errors in their reported position, speed, and acceleration. Both sources contribute to the total uncertainty that the controller must account for when planning over the horizon H.

2.3.2 Stochastic Evolution of State Uncertainty

In the presence of both process and measurement noise, the controller does not have access to the true state x_k but instead maintains a belief over it based on the history of observations $\{y_0, \ldots, y_k\}$. For Gaussian noise and linear dynamics, this belief is exactly a Gaussian distribution $x_k \sim \mathcal{N}(\mu_k, P_k)$, maintained by a Kalman filter. For nonlinear systems, an Extended Kalman Filter (EKF) is used [31].
Assuming that the control input u_k is known exactly and deterministic at time step k, the uncertainty propagation is solely determined by the state uncertainty and the process noise. Under this assumption, the covariance prediction step of the EKF is given by:

$$P_{k+1|k} = A_k P_k A_k^T + \Sigma_w \tag{2.12}$$

where $A_k = \partial f / \partial x|_{x=\mu_k}$ is the Jacobian of f evaluated at the predicted mean. Second, the update step incorporates the new measurement y_{k+1}, reducing uncertainty:

$$P_{k+1} = (I - K_{k+1} H_{k+1}) P_{k+1|k} \tag{2.13}$$

where $H_{k+1} = \partial h / \partial x|_{x=\mu_{k+1|k}}$ is the observation Jacobian and:

$$K_{k+1} = P_{k+1|k} H_{k+1}^T (H_{k+1} P_{k+1|k} H_{k+1}^T + \Sigma_v)^{-1} \tag{2.14}$$

is the Kalman gain. The prediction step causes uncertainty to grow due to process noise, while the update step reduces it by incorporating new measurements. The balance between these two effects determines the steady-state uncertainty level.
In the MPC prediction horizon, no new measurements arrive after $h = 0$, so only the prediction step applies for $h = 1, \ldots, H$. Starting from the initial covariance P_0 at observation time, the predicted covariance grows as:

$$P_{h+1} = A_h P_h A_h^T + \Sigma_w, \quad h = 0, 1, \ldots, H-1 \tag{2.15}$$

Even if the initial state is known exactly ($P_0 = 0$), process noise causes the uncertainty to grow with each prediction step. When measurement noise also contributes to P_0 (because the initial observation is itself uncertain), both sources compound over the horizon.
In this work, empirical analysis of the trajectory prediction error (Section 4.3) shows that process noise is negligible over the short horizon $H \leq 5$ steps used in the DP. The dominant source of uncertainty is therefore measurement noise at $h = 0$, which propagates deterministically through the prediction model. The explicit derivation of this propagation for the constant-acceleration prediction model is given in Section 3.6.

2.3.3 The Principle of Optimality in Stochastic Environments

In deterministic optimal control, the objective is to find a sequence of inputs that minimizes a cost function. However, in the presence of both process noise w_k and measurement noise v_k, the true state x_k and the observed state y_k are both random

variables. The controller therefore optimizes over the expected cost with respect to both noise sources [25]:

$$J(\mu_0) = \min_{u_0,\ldots,u_{H-1}} \mathbb{E}_{\{w_k,v_k\}_{k=0}^{H-1}} \left[\sum_{k=0}^{H-1} g(x_k, u_k) + q(x_H) \right] \tag{2.16}$$

where μ_0 is the mean of the initial state estimate and the expectation is taken with respect to the joint distribution of the noise sequences $\{w_k\}$ and $\{v_k\}$. The stage costs $g(x_k, u_k)$ are functions of the true state x_k, which is itself a random variable because it depends on the accumulated process noise up to step k and cannot be observed exactly due to measurement noise. The terminal cost $q(x_H)$ is similarly stochastic.

According to the Stochastic Dynamic Programming framework, the optimal value function $V^*(\mu_k, P_k)$, which is now a function of (μ_k, P_k) rather than the true state, satisfies the recursive Bellman equation [26, 27]:

$$V^*(\mu_k, P_k) = \min_{u_k} \left\{ \mathbb{E}_{w_k,v_k} \left[g(x_k, u_k) \right] + \mathbb{E}_{w_{k+1},v_{k+1}} \left[V^*(\mu_{k+1}, P_{k+1}) \right] \right\} \tag{2.17}$$

Both the stage cost and the future value function are inside the expectation because x_k is uncertain (measurement noise), and μ_{k+1} and P_{k+1} depend on the next observation which is also uncertain (both noise sources). This formulation ensures that the controller accounts for all uncertainties, and weights them by their probability of occurrence.

In practice, solving this belief-space Bellman equation exactly is intractable for general nonlinear systems. The chance-constrained approach described in the following sections provides a tractable approximation. In it, the controller optimizes the cost evaluated at the mean state μ_k while enforcing probabilistic safety constraints that account for the P_k.

2.3.4 Chance-Constrained Optimization

A primary challenge in stochastic control is the satisfaction of state constraints, such as safety boundaries. Since process and measurement noise both are Gaussian noise, it is impossible to guarantee that a constraint $c(x_k, u_k) \leq 0$ will hold for all possible noise realizations. To address this, Charnes et al. [32] introduced Chance-Constrained Programming (CCP). Instead of hard constraints, CCP requires that the probability of violation remains below a risk tolerance $\epsilon \in (0, 1)$:

$$\mathbb{P}\{c(x_k, u_k) \leq 0\} \geq 1 - \epsilon \tag{2.18}$$

where $1 - \epsilon$ is the confidence level. This approach allows the designer to explicitly tune the trade-off between system performance and safety risk. The total uncertainty entering the constraint is the sum of contributions from process noise accumulated over the horizon and measurement noise at the initial observation, both of which are captured in the covariance P_k.

To solve the optimization problem numerically, the chance constraint must be converted into a Deterministic Equivalent. Following the derivation in [33], assume a linear constraint $c(x_k) = a^T x_k - b \leq 0$ where x_k is a Gaussian random variable

$x_k \sim \mathcal{N}(\mu_x, \Sigma_x)$, with Σ_x reflecting the combined effect of process and measurement noise propagated to step k. The constraint is:

$$\mathbb{P}\{a^T x_k \leq b\} \geq 1 - \epsilon \tag{2.19}$$

By standardizing the variable, we let $z = \frac{a^T x_k - a^T \mu_x}{\sqrt{a^T \Sigma_x a}}$, where $z \sim \mathcal{N}(0, 1)$. The constraint becomes:

$$\mathbb{P}\left\{z \leq \frac{b - a^T \mu_x}{\sqrt{a^T \Sigma_x a}}\right\} \geq 1 - \epsilon \tag{2.20}$$

After using the standard normal cumulative distribution function (CDF), $\Phi(\cdot)$, it can be obtained:

$$\Phi\left(\frac{b - a^T \mu_x}{\sqrt{a^T \Sigma_x a}}\right) \geq 1 - \epsilon \tag{2.21}$$

By applying the inverse CDF, the deterministic equivalent is:

$$a^T \mu_x + \Phi^{-1}(1 - \epsilon)\sqrt{a^T \Sigma_x a} \leq b \tag{2.22}$$

This results in a tightened constraint. The term $\Phi^{-1}(1 - \epsilon)\sigma$ is a stochastic safety buffer that scales with the total standard deviation $\sigma = \sqrt{a^T \Sigma_x a}$, which itself grows with both process and measurement noise contributions. For a 99% confidence level ($\epsilon = 0.01$), $\Phi^{-1}(0.99) \approx 2.33$, meaning the mean state must remain 2.33σ away from the boundary to satisfy the chance constraint.
The integration of these probabilistic constraints into a receding horizon framework results in Stochastic Model Predictive Control (SMPC). At each time step, the controller solves an MPC problem where the deterministic safety constraints are replaced by their chance-constrained equivalents, with the safety buffer $\Phi^{-1}(1 - \epsilon)\sigma_h$ expanding with the horizon step h as uncertainty accumulates from both process and measurement noise [33]. When uncertainty is small (high certainty), the tightened constraint is close to the deterministic one and the controller behaves nearly identically to its deterministic counterpart. As uncertainty grows, whether from a long prediction horizon, large process noise, or imprecise initial measurements, the expanding safety buffer forces the controller to adopt a more conservative stance, maintaining a larger margin from potential constraint violations.

2.4 Reinforcement Learning and Value Function Approximation

Reinforcement Learning provides a complementary perspective to MPC for sequential decision making under uncertainty. In contrast to MPC, which solves a constrained optimization problem at every time step using an explicit prediction model, RL learns a control policy by repeated interaction with a stochastic environment and the maximization of a cumulative reward signal [13]. The two frameworks are connected through the value function, meaning the optimal cost-to-go in MPC can be

interpreted as the optimal value in RL, and this equivalence is utilized in the hybrid architecture proposed in this thesis.

2.4.1 The Markov Decision Process

A reinforcement learning problem is commonly formalized as a Markov Decision Process (MDP) [34], which provides a mathematical framework for sequential decision-making under uncertainty. An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$:

- $\mathcal{S}$ is the state space, i.e. the set of states that describe the environment.
- $\mathcal{A}$ is the set of actions available to the agent.
- $\mathcal{P}(s' \mid s, a)$ is the state-transition probability, giving the probability of reaching state s' after taking action a in state s.
- $\mathcal{R}(s, a)$ is the reward function. The scalar reward received at time step t is denoted $r_t = \mathcal{R}(s_t, a_t)$ and quantifies the immediate desirability of taking action a_t in state s_t.
- $\gamma \in [0, 1)$ is a discount factor that ensures the convergence of infinite-horizon returns and emphasizes near-term outcomes.

The agent interacts with the environment over a sequence of discrete time steps, as illustrated in Figure 2.3. At each step t, the agent observes the current state s_t and selects an action a_t according to its policy $\pi(a \mid s)$, which specifies a probability distribution over actions in each state. The environment then transitions to a new state s_{t+1} according to $\mathcal{P}$ and returns a reward r_t, and the cycle repeats.

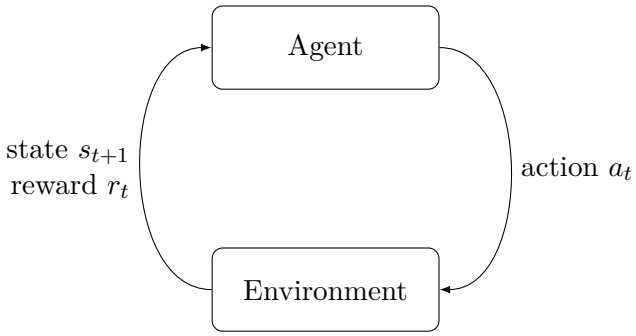


Figure 2.3: The agent-environment interaction loop in reinforcement learning.

The objective is to find a policy that maximizes the expected discounted return, defined as the cumulative sum of future discounted rewards:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}. \tag{2.23}$$

2.4.2 State-Value and Action-Value Functions

Two value functions characterize the long-term consequences of a policy:

1. The state-value function:

$$V^\pi(s) = \mathbb{E}_\pi \left[G_t \mid s_t = s \right] \tag{2.24}$$

which gives the expected return obtained when the system is initialized in state s and then follows policy π.

2. The action-value function, also called the Q-function:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[G_t \mid s_t = s, a_t = a\right] \tag{2.25}$$

which quantifies the expected return when the agent takes a specific action a in state s and follows π from the next time step onwards.

The two functions are related by $V^{\pi}(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^{\pi}(s,a)]$.
The goal of reinforcement learning is to find an optimal policy π^* that maximizes the expected return from every state. The corresponding optimal value functions are defined as the maxima over all policies,

$$V^*(s) = \max_{\pi} V^{\pi}(s), \qquad Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a), \tag{2.26}$$

and represent the best achievable return from a given state, or state–action pair, respectively. Under the optimal policy π^*, the relation between the two functions reduces to the deterministic identity

$$V^*(s) = \max_{a\in\mathcal{A}} Q^*(s,a), \tag{2.27}$$

since an optimal agent selects the action with the highest expected return. Identity (Eq. 2.27) is the central equation linking RL to MPC in the proposed hybrid framework, where the MPC requires a state-value function as a terminal cost, and DQN naturally produces an estimate of Q^*.

2.4.3 The Bellman Equations

Both value functions satisfy recursive Bellman equations, which decompose the long-term return into an immediate reward plus the discounted value of the successor state [25, 26]. For the optimal action-value function, the Bellman optimality equation is:

$$Q^*(s,a) = \mathbb{E}_{s'\sim\mathcal{P}}\left[\mathcal{R}(s,a) + \gamma \max_{a'\in\mathcal{A}} Q^*(s',a')\right]. \tag{2.28}$$

This provides the basis for value-iteration methods. Any function Q for which the right-hand side equals the left-hand side at every (s,a) is necessarily Q^*, and iterative application of the Bellman operator drives any initial estimate towards the unique fixed point under standard contraction arguments.

2.4.4 Tabular Q-Learning

When the state space is small enough to be enumerated, Eq. 2.28 can be solved iteratively using sampled transitions. Q-learning [35] maintains an explicit table of values $Q(s,a)$, and updates them after each observed transition (s_t, a_t, r_t, s_{t+1}):

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha_{\mathrm{lr}}\left[r_t + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)\right], \tag{2.29}$$

where α_{lr} is a learning rate and the expression inside the brackets is known as the temporal-difference error. Under standard conditions, including bounded rewards,

sufficient exploration, and a decaying learning rate, the iterations converge to Q^* [13, 35].
Tabular Q-learning is not directly applicable to the highway driving problem considered in this thesis. If the state vector is considered to contain continuous quantities such as the longitudinal velocity and the gaps to surrounding vehicles, any enumeration of $\mathcal{S}$ would be infeasible. This motivates the use of function approximation.

2.4.5 Function Approximation and Deep Q-Networks

When $\mathcal{S}$ is continuous or high-dimensional, the action-value function is parameterized as $Q(s, a; \theta)$, where θ denotes the parameters of a function approximator [13, 27]. Deep Q-Networks (DQN) [36] use a deep neural network as the approximator and learn θ by minimizing a mean-squared temporal-difference loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(y - Q(s, a; \theta)\right)^2\right], \quad y = r + \gamma \max_{a'} Q(s', a' : \theta^-), \tag{2.30}$$

where the expectation is taken over a replay buffer $\mathcal{D}$ of past transitions and θ^- is a periodically updated copy of the parameters used to compute the bootstrap target. Two modifications distinguish DQN from naive Q-learning with a neural network and are essential for stable training:

1. **Experience replay** stores transitions in $\mathcal{D}$ and trains on uniformly sampled minibatches. This prevents correlations between consecutive updates and improves sample efficiency [36, 37].
2. **A target network** with parameters θ^- supplies the bootstrap target in Eq. 2.30 and is held fixed for a number of updates before being synchronized with θ. This prevents the moving target problem that arises when the same network is used to both define and optimize the loss [36].

DQN is restricted to discrete action spaces, since the greedy action $\arg\max_a Q(s, a : \theta)$ requires enumeration over $\mathcal{A}$. This is consistent with the application in this thesis. The control of the truck is naturally decomposed into a finite set of acceleration levels combined with three lateral commands (left, stay, right), and a discrete action space is therefore well suited to the problem. Further details about the dynamic of system is presented in Chapter 3. Exploration during training is performed using an ε-greedy policy that selects the greedy action with probability $1 - \varepsilon$ and a uniformly random action otherwise, with ε transformed from a high initial value to a small final value over the course of training [13, 36].

2.4.6 The Bridge to Model Predictive Control

As discussed in Section 2.2, the backward DP requires a terminal cost $q(x_H)$ to initialize the recursion, and the quality of the resulting policy depends directly on how well this terminal cost approximates the true infinite-horizon cost-to-go $V^*(x_H)$. Section 2.2.2 described one approximation approach, rolling out a base policy for H_{roll} additional steps and using the accumulated stage cost as a approximation for $V^*(x_H)$. While rollout provides a policy improvement guarantee over the base policy [8, 26], its quality is fundamentally bounded by the quality of the base policy itself. If the base policy is myopic or poorly tuned for a particular traffic scenario, the

rollout estimate will be incorrect. Furthermore, the rollout horizon H_{roll} must be kept finite for computational reasons, meaning the true infinite-horizon tail beyond $H + H_{roll}$ is still ignored.

A more powerful alternative is to learn the terminal cost directly from experience. Rather than constructing $q(x_H)$ analytically from a known base policy, a trained function approximator on interaction data from the same environment can yield a better estimate. DQN provides such approximation, $Q(s, a : \theta)$ of the optimal action-value function Q^*. These two quantities are connected by identity Eq. 2.27, which yields:

$$\hat{V}(x_H) = \max_{a \in \mathcal{A}} Q(x_H, a : \theta) \approx V^*(x_H). \tag{2.31}$$

The terminal cost used in the hybrid controller is therefore obtained by a single forward pass of the trained Q-network at the terminal state, followed by a maximization over the discrete action set. Because the rewards used during DQN training are the negatives of the stage costs in the MPC, the sign of Eq. 2.31 is flipped before being inserted into the FHOCP, and the resulting terminal cost is on the same scale as the stage costs by construction.

This structure has two good properties. First, the computational cost of the terminal evaluation is independent of H and consists of a single neural-network operation per terminal state, which is compatible with the real-time requirements of receding-horizon control. Second, because Q function is trained on long trajectories from the same simulator, the resulting terminal cost captures information that lies beyond any practical MPC horizon, including the consequences of multi-step lane changes and traffic that becomes visible only after the planning window has elapsed.

2.5 Approximation Error, Terminal Cost Weighting and Discount Factor

Whether the terminal cost is obtained from a rollout or from a learned value function, it is in general only an approximation of the infinite-horizon cost-to-go. As noted in [11], an approximation of the infinite horizon cost has to be made whenever the prediction horizon is finite, and the resulting mismatch might cause the loss of closed-loop stability. This observation has motivated a substantial literature on the design and tuning of terminal ingredients [38, 39, 40].

A practical consequence is that the relative magnitude of $q(x_H)$ and the stage costs $g(x_h, u_h)$ has a direct effect on closed-loop behavior. If $q(x_H)$ is too large, the controller is dominated by an approximate quantity and may take actions that are good according to the approximation but suboptimal with respect to the true objective. If $q(x_H)$ is too small, the controller becomes effectively myopic and the benefit of the extended horizon is lost. As discussed in [41], a good terminal cost is often tuned empirically by observing performance, which reflects the absence of closed-form rules for selecting its scale in nonlinear settings. To make this trade-off explicit, one approach is that the terminal cost is multiplied by a non-negative

weight α,

$$J = \sum_{h=0}^{H-1} g(x_{k+h}, u_{k+h}) + \alpha \; q(x_{k+H}) \tag{2.32}$$

which acts as a scalar tuning knob between purely stage-cost-driven behavior ($\alpha = 0$) and full reliance on the approximate cost-to-go ($\alpha = 1$). The role of α is particularly important when the prediction horizon H is short relative to the dynamics of the surrounding traffic, since in that regime stability and performance must be enforced primarily through the terminal cost rather than through horizon length.
Another approach is the use of a discount factor. Discounting is most commonly associated with RL and DP, where future rewards or costs are weighted less than immediate ones in order to improve convergence and represent long-term uncertainty. In infinite-horizon RL problems, the discounted return is typically written as:

$$J = \sum_{h=0}^{\infty} \gamma^h g(x_{k+h}, u_{k+h}), \tag{2.33}$$

where $0 \leq \gamma < 1$ is the discount factor. A smaller γ places more emphasis on short-term behavior, while values close to one increase the influence of long-term predictions, as described in Section 2.4.1.
Although discounting originates primarily from RL, similar ideas have also been investigated in MPC and optimal control. In discounted MPC, future stage costs are weighted according to their prediction step, which can improve robustness to approximation errors and reduce the sensitivity of the controller to inaccurate long-horizon predictions [42, 43]. In the context of rollout-based terminal costs, discounting can also reduce the influence of uncertain trajectory predictions far into the future, where modeling errors and interaction uncertainties become more significant.
The discounted finite-horizon cost can therefore be written as

$$J = \sum_{h=0}^{H-1} \gamma^h g(x_{k+h}, u_{k+h}) + q(x_{k+H}), \tag{2.34}$$

where the terminal cost also has its own discounting. Further discussion regarding the use of discounting in reinforcement learning and value-function approximation is provided in Section 3.5.2.

3
Methods

This chapter presents the complete technical formulation of the proposed control framework. It begins by defining the Pure MPC method in Section 3.3. Section 3.5 describes the rollout-based terminal cost approximation, which extends the effective planning horizon using a computationally efficient base policy. In section 3.6, the stochastic MPC method is explained to handle traffic's uncertainty. Section 3.7 formulates the reinforcement learning method, and finally, Section 3.8 combines these components into the hybrid MPC-RL architecture and derives the terminal cost integration.

3.1 SUMO Simulation Environment

The proposed control framework is developed and evaluated using the Simulation of Urban MObility (SUMO) platform, an open-source microscopic traffic simulator developed by the German Aerospace Center (DLR) [44]. SUMO provides a realistic multi-vehicle traffic environment in which each vehicle evolves according to its own motion and lane-changing models. Through the Traffic Control Interface (TraCI), the controller can access vehicle states such as position, velocity, acceleration, and lane index, while also issuing control commands to the ego vehicle.
The simulator serves as the environment in which the MPC, rollout, and reinforcement-learning components operate. In particular, the traffic prediction models, state representation, observation design, and controller evaluation procedures presented throughout this chapter are all formulated with respect to the information available from SUMO. Consequently, several aspects of the proposed framework, including vehicle interactions and traffic evolution, are naturally tied to the simulation environment.
A detailed description of the highway scenario, traffic generation process, vehicle parameters, and simulation settings used for evaluation is provided in Chapter 4.1.1.

3.2 Proposed Framework

Figure 3.1 illustrates the overall architecture of the proposed method.
The control loop operates within the SUMO traffic simulation environment. At each decision step, the simulator provides the current traffic state, including the ego vehicle state and information about surrounding vehicles. This information is passed to the MPC controller, which performs a finite-horizon optimization and evaluates candidate trajectories.

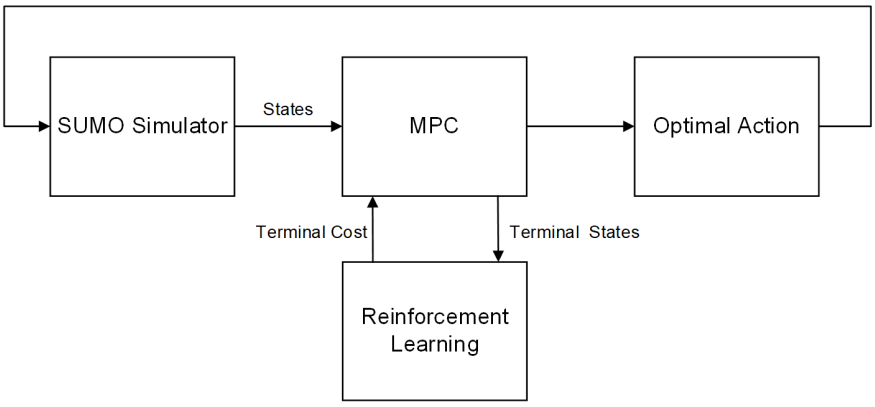


Figure 3.1: Overview of the proposed hybrid MPC-RL framework.

Then the terminal states generated by the MPC are converted into RL observations and evaluated by a reinforcement learning agent. The RL agent estimates the expected long-term return associated with each terminal state and returns a corresponding terminal cost approximation to the MPC optimizer.
The MPC combines this terminal cost estimate with the accumulated stage costs along the prediction horizon and selects the action sequence that minimizes the resulting objective function. Only the first action of the optimal sequence is applied to the simulator, after which the process repeats at the next control step according to the receding-horizon principle.

3.3 MPC Mathematical Formulation

The MPC controller solves a finite-horizon optimal control problem at each time step k, selecting a sequence of longitudinal accelerations and lateral lane-change commands that minimize a cumulative cost over a prediction horizon H. The formulation operates on a discrete-time kinematic model of the ego truck. The receding-horizon principle then applies only the first element of the optimal sequence before re-solving at the next time step.

3.3.1 States and Inputs

The state of the ego vehicle at time step k is defined by the vector:

$$x_k = [s_k, l_k, v_k, c_k]^T \tag{3.1}$$

where:

- $s_k \in \mathbb{R}$: Front-end longitudinal position (m).
- $l_k \in \{0, 1, \ldots, N_{lanes} - 1\}$: Lane index.
- $v_k \in [v_{min}, v_{max}]$: Longitudinal velocity (m/s).
- $c_k \in \{0, 1, \ldots, C_{steps} - 1\}$: Counter managing the lane-change transition and the cooldown period.

The control input is defined as:

$$u_k = [a_k, \Delta l_k]^T \tag{3.2}$$

where $a_k \in [a_{min}, a_{max}]$ is the discrete longitudinal acceleration with the step of $1m/s^2$ and $\Delta l_k \in \{-1, 0, 1\}$ is the lateral lane change command.

3.3.2 System Dynamics

The state transitions follow a discrete-time kinematic model with a sampling time Δt. The longitudinal position is modeled using a double-integrator kinematic model, where the position is updated using the current velocity and acceleration:

$$s_{k+1} = s_k + v_k \Delta t + \frac{1}{2} a_k \Delta t^2 \tag{3.3}$$

The velocity evolves according to the applied acceleration. To remain consistent with the discretized state space used by the DP solver, the velocity is clipped to the admissible range $[v_{min}, v_{max}]$:

$$v_{k+1} = \begin{cases} v_{max} & \text{if } v_k + a_k \Delta t \geq v_{max} \\ v_{min} & \text{if } v_k + a_k \Delta t \leq v_{min} \\ v_k + a_k \Delta t & \text{otherwise} \end{cases} \tag{3.4}$$

The lane state is governed by both the lane-change command Δl_k and the lane-change counter c_k. A lane-change command can only be executed when the vehicle is not actively performing a lane change. Otherwise, the lane index remains unchanged:

$$l_{k+1} = \begin{cases} l_k + \Delta l_k & \text{if } 0 \leq c_k \leq C_{start} \\ l_k & \text{if } c_k > C_{start} \quad (\text{Constraint: } \Delta l_k = 0) \end{cases} \tag{3.5}$$

$$c_{k+1} = \begin{cases} C_{steps} - 1 & \text{if } 0 \leq c_k \leq C_{start} \ \& \ \Delta l_k \neq 0 \\ c_k - 1 & \text{if } c_k > C_{start} \text{ or } (0 < c_k \leq C_{start} \text{ and } \Delta l = 0) \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

The counter c_k serves two purposes:

1. **Transition Lock:** When $c_k > C_{start}$, the vehicle is actively performing a lane change in the simulator. During this period, Δl_k is forced to 0 to ensure the maneuver is completed and there are no multiple lane changing penalties during transition.
2. **Behavioral Cooldown:** When $0 < c_k \leq C_{start}$, the physical maneuver is finished, but a high penalty (P_{CD}) is applied to any new lane-change command. If a change is initiated anyway, the counter resets to $C_{steps} - 1$, restarting the full cycle.

This state-dependent restriction on the action space ensures the optimization only explores executable trajectories, preventing command conflicts while the simulator's lateral actuator is active.

3.4 MPC Solver

Although MPC is often implemented using quadratic or nonlinear programming solvers, the present work adopts a Dynamic Programming formulation. This choice is motivated by two factors. First, the state and action spaces are discretized, making backward DP a natural solution method. Second, the DP formulation allows direct incorporation of alternative terminal-cost approximations, including rollout-based and reinforcement-learning-based value functions, while maintaining a common optimization framework across all controller variants.

3.4.1 Optimization Objective

The controller minimizes the cumulative cost function J over the horizon H:

$$J^* = \min_{u_0,\ldots,u_{H-1}} \sum_{h=0}^{H-1} g(x_h, u_h) + q(x_H) \tag{3.7}$$

The stage cost $g(x, u)$ is:

$$\begin{aligned} g(x,u) = & Cost_{speed} + Cost_{accel} + Cost_{LC} + Cost_{Safety} + \\ & Cost_{CD} + Cost_{LL} + Cost_{col} \end{aligned} \tag{3.8}$$

where the terminal cost is set to zero ($q(x_H) = 0$) for pure MPC baseline and is replaced by a learned or rollout-based approximation of the infinite-horizon cost-to-go for hybrid variants.
The speed cost shown in Eq.3.9 penalizes deviations from the target speed v_{ref} to ensure that the vehicle maintains a steady traffic flow.

$$Cost_{speed} = w_v(v - v_{ref})^2 \tag{3.9}$$

The acceleration cost in Eq.3.10 penalizes high-magnitude control actions to prioritize passenger comfort and ensure the longitudinal movement of the truck remains smooth. In the next section, the other constraints are described in detail in Sections 3.4.2.1 through 3.4.2.5.

$$Cost_{accel} = w_a a^2 \tag{3.10}$$

3.4.2 Constraints

Although MPC is often formulated using explicit hard constraints, most safety and operational requirements in this work are implemented as large penalty terms in the objective function. This formulation is adopted because the backward DP framework evaluates a finite set of discrete states and actions, making soft constraints more practical to integrate while still strongly discouraging unsafe behavior.

3.4.2.1 Lane Change Cost

The lane changing penalty is defined to discourage frequent or unnecessary lateral maneuvers. To make the transition realistic, a lane change is a non-instantaneous process that requires a fixed duration T_{LC} to complete in SUMO (the simulator introduced in Section 4.1.1). To model this, the state vector is augmented with a counter $c \in \{0, 1, \ldots, C_{steps} - 1\}$, where $C_{steps} = T_{TD}/\Delta t$. T_{TD} is the cooldown time, which will be discussed in 3.4.2.3.
The lane changing cost is defined as:

$$Cost_{LC} = \begin{cases} P_{LC} & \text{if } c \leq C_{start} \text{ and } \Delta l \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

The counter c locks the vehicle into a maneuver once initiated. By enforcing $\Delta l = 0$ whenever $c > C_{start}$, the controller is prevented from commanding another lane change during the transition period T_{LC}. This mechanism ensures that the lane changing penalty P_{LC} is only considered once at the start of the action, preventing multiple penalties from being triggered while the vehicle is still in the process of reaching the target lane.

3.4.2.2 Safety Cost

Safety is enforced via a quadratic penalty when the gap g to surrounding vehicles is less than the safe distance d_{safe}. The total safety cost is:

$$Cost_{Safety} = Cost_{front} + Cost_{back} \tag{3.12}$$

- $Cost_{front}$: For the vehicle ahead in the target lane corresponding to action Δl:

$$Cost_{front} = w_{front} \cdot \max(0, d_{safe} - g_{front})^2, \quad d_{safe} = D_0 + T_{head} v \tag{3.13}$$

- $Cost_{back}$: This cost is active only when a lane change is commanded ($\Delta l \neq 0$), and applies to the follower on the target lane:

$$Cost_{back} = \begin{cases} 0 & \text{if } \Delta l = 0 \\ w_{back} \cdot \max(0, d_{safe} - g_{back})^2, \quad d_{safe} = D_0 + T_{head} v_{follow} & \text{if } \Delta l \neq 0 \end{cases} \tag{3.14}$$

Further details for the lane change logic and target lane follower will be provided in 3.4.3.

3.4.2.3 Lane Change Cooldown Penalty

To prevent dangerous behavior of vehicle that could lead to frequent lane changing and having a realistic behavior, a temporal cooldown penalty is implemented. The cooldown cost $Cost_{CD}$ penalizes the initiation of a new maneuver if a minimum time threshold T_{CD} has not passed since the completion of the previous one. While the total cycle duration is T_{TD}, the first T_{LC} seconds are physically locked by the simulator. This leaves an effective cooldown period T_{CD} where a new lane change is

technically possible but heavily penalized by P_{CD}. This reflects human-like driving behavior on the road, prioritizing safety and predictability over marginal gains in target speed.
The reason a separate penalty was considered for this issue from the lane change penalty is that if the lane change penalty were set too high, the controller might fail to take action even when a maneuver is necessary. So by separating them and assigning a higher value to P_{CD}, two consecutive lane changes is avoided, but at the same time, the controller can change the vehicle's lane when necessary and after the required time has passed since the previous action.
The cooldown cost is defined as:

$$Cost_{CD} = \begin{cases} P_{CD} & \text{if } \Delta l \neq 0 \text{ and } 0 < c \leq C_{start} \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

3.4.2.4 Left Lane Occupancy Penalty

To encourage the vehicle to return to the slow-traffic lane after completing an overtaking maneuver, a progressive penalty is applied for occupying the left-most lanes. This reflects standard traffic regulations and social norms for heavy vehicles, prioritizing the availability of high-speed lanes for passenger cars.
The left lane cost is defined as:

$$Cost_{LL} = \begin{cases} w_{LL} \cdot l_{ego} & \text{if } l_{ego} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

where $l \in \{0, 1, \ldots, N_{lanes} - 1\}$ is the current lane index, with $l = 0$ representing the right-most (slowest) lane.
This creates a continuous pressure on the controller, and the vehicle will only occupy the left lanes when the functional benefits, such as maintaining the reference velocity v_{ref} or avoiding a slow-moving leader. This formulation ensures that the truck naturally goes back to the slow lane as soon as it is safe and efficient to do so.

3.4.2.5 Collision Cost Logic

The controller uses a large collision penalty P_{col} to prevent collisions to surrounding vehicles, and fires when the gap falls within a physically unsafe range. Similar to safety cost, it has two parts:

$$Cost_{col} = Cost_{col-front} + Cost_{col-back} \tag{3.17}$$

where:

$$Cost_{col-front} = P_{col} \quad \text{if} \quad g_{front} \leq 0 \tag{3.18}$$

and:

$$Cost_{col-back} = \begin{cases} 0 & \text{if } \Delta l = 0 \\ P_{col} & \text{if } \Delta l \neq 0 \quad \text{and} \quad g_{back} \leq \frac{(v_{follow} - v)^2}{2a_{break}} \end{cases} \tag{3.19}$$

where a_{break} is the maximum deceleration of the following vehicle, which is assumed to be equal to the maximum deceleration value of the ego truck for a more conservative lane change.

The condition $g_{back} \leq (v_{follow} - v)^2/(2a_{break})$ is more conservative than the simple $g_{back} \leq 0$ check that is used for the front gap. This change is intentional. In early evaluations, a repeated collision scenario was observed. In some case, the ego truck faced a slow leader ahead and a fast follower in the side lane with a positive but small rear gap. The DP computed that staying in the current lane would accumulate several steps of front safety cost, whereas executing a lane change would have only a single step back safety cost, making the lane change appear cheaper. After the lane change, the fast follower, now in the same lane as ego, continued to get closer rapidly, resulting in a rear-end collision that the DP had not penalized because penalty for the collision from behind on the same lane is not defined to it. This problem is solved with a physics-based condition. If the follower is traveling faster than ego and the gap is insufficient for it to brake to ego's speed before reaching ego's rear bumper, the collision penalty P_{col} is applied regardless of whether the physical gap is currently positive.

3.4.3 Lane Change Commitment and Model Mismatch

A critical challenge in the proposed controller is the discrepancy between the Discrete State Space of the DP formulation and the Continuous Lateral Physics of the SUMO simulator. While a lane change command is triggered at a single time step, the physical transition requires a duration of T_{LC} (equivalent to N_{steps}) to complete. This creates a divergence between the Backward DP and the Forward step Execution:

1. **Backward DP: Immediate State Assignment**
   In the backward pass, the algorithm treats the lane transition as an instantaneous shift in logical occupancy to evaluate the long-term benefit of the maneuver.

$$l_{h+1}^{DP} = l_h + \Delta l_h \quad \text{for } \Delta l \neq 0 \tag{3.20}$$

   As a result, from the very first step of the transition ($c_h = C_{steps} - 1$), the safety costs constraint in the DP is calculated with respect to the target lane leader. This logic allows the value function J to capture the immediate flow advantages of the new lane.
2. **Forward Simulation: Persistent Physical Occupancy**
   In contrast, the forward simulation recognizes that the vehicle remains physically present on the original lane l_h for a part of the duration of N_{steps}.

$$l_{h+1}^{SIM} = \begin{cases} l_h + \Delta l & \text{only at the step of lane change command} \\ l_h & \text{for the rest of transition, before it physically crosses the line} \end{cases} \tag{3.21}$$

   By using counter c_h, the multiple lane changing cost is avoided during transition. However, the mismatch between DP and simulation remains, which can cause a problem in other constraints, specially safety cost.
3. **Bridging the Gap**
   To make the DP and simulation have the same logic, the controller maintains a separate variable l_{reg} (the logical lane), which tracks the lane the controller

considers itself committed to:

$$l_{reg,k} = \begin{cases} l_{SUMO,k} & \text{if } c_k < C_{start} \quad \text{(No active transition)} \\ l_{reg,k-1} & \text{if } c_k \geq C_{start} \quad \text{(Transition in progress)} \end{cases} \tag{3.22}$$

where $l_{SUMO,k}$ is the lane index reported by SUMO.
This ensures that during active lane change, the DP plans from the target lane, correctly computing the cost of remaining behind the target lane's leader.

3.4.4 Traffic Environment Prediction

To evaluate the stage costs in Eq. 3.7 at each step h of the horizon, the controller requires predicted trajectories $(s_{i,h}, v_{i,h}, l_{i,h})$ for all surrounding vehicles i over $h = 0, \ldots, H$. Two methods were considered for this purpose.

3.4.4.1 Method 1: Reactive Multi-Agent Projection

To achieve a high prediction accuracy based on a realistic policy, this method implements a reactive simulation where surrounding vehicles act as intelligent agents with a heuristic policy.
In this predictive framework, the Ego vehicle is projected forward at a constant velocity and fixed lane to serve as a stable agent for surrounding vehicles. Surrounding vehicles also have fix lane policy:

$$s_{h+1} = s_h + v_0 \cdot \Delta t \tag{3.23}$$

$$v_{h+1} = v_0 \tag{3.24}$$

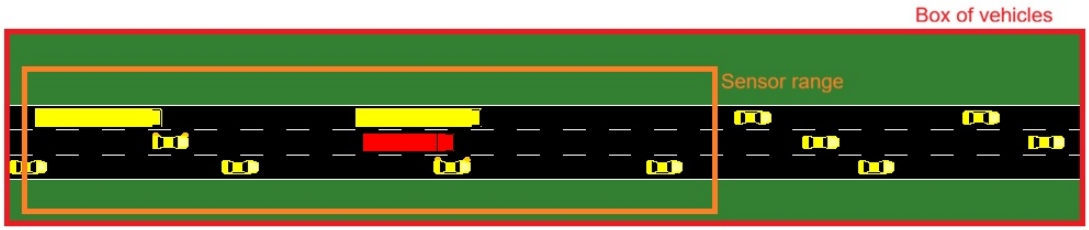
$$l_{ego,h+1} = l_{ego,0} \tag{3.25}$$

Fixing the Ego's state and surrounding vehicles' lane simplifies the prediction by avoiding a loop between the controller's and other vehicles logic, which does not have huge impact over a short lookahead horizon.
Every neighboring agent i updates its state at each discrete step h based on the collective environment snapshot X_h. The state transition follows the same logic as described in Section 3.5.1, but without the lane changing:

$$[s_{i,h+1}, v_{i,h+1}, l_{i,h+1}, c_{i,h+1}] = \pi_{SUMO}(s_{i,h}, v_{i,h}, l_{i,h}, c_{i,h}, X_h) \tag{3.26}$$

The snapshot X_h includes the positions and velocities of all neighboring agents and the Ego vehicle. This allows the prediction to capture reactive behaviors.
However, a practical limitation arises at the boundary of the sensor range (see Section 3.4.4.4). Vehicles at the front of each lane within the sensor range have no visible leader, and their true leader lies beyond the sensing horizon. Without modification, the policy π_{SUMO} treats these boundary vehicles as having a clear road ahead and accelerates them toward their maximum speed. This triggers a chain reaction:

1. The vehicles immediately behind the boundary leaders also accelerate unnecessarily,



(a) Actual traffic

(b) Traffic within sesnor range

Figure 3.2: A simple demonstration of the effect of sensor range in the Method 2 prediction (simulated in SUMO).

2. producing growing position errors over longer horizons that are not representative of true traffic behavior.

To address this, the lead vehicle in each lane at the start of the prediction (the furthest-ahead vehicle within the sensor range) is identified at $h = 0$ and assigned a constant-speed, fixed-lane trajectory for the duration of the horizon. All other vehicles then react to these frozen leaders and to the ego vehicle using the standard π_{SUMO} policy. This ensures that the boundary condition does not corrupt the reactive behavior of the follower vehicles, while avoiding unrealistic acceleration at the sensor edge.

3.4.4.2 Method 2: Constant Acceleration

By using a simple approach, the vehicle forward problem is solved by using its current acceleration $a_{i,0}$:

$$s_{i,h+1} = s_{i,h} + v_{i,h} \cdot \Delta t \tag{3.27}$$

$$v_{i,h+1} = \begin{cases} v_{i,\max} & \text{if } v_{i,h} + a_{i,0} \cdot \Delta t \geq v_{i,\max} \\ 0 & \text{if } v_{i,h} + a_{i,0} \cdot \Delta t \leq 0 \\ v_{i,h} + a_{i,0} \cdot \Delta t & \text{otherwise} \end{cases} \tag{3.28}$$

By comparing the error between predicted and actual states of surrounding vehicles after H steps, Method 2 has the lower error. A comparison of prediction error across two methods confirmed that Method 2 achieves the lowest mean position error at the short horizons ($H \leq 5$ steps, 1 second) used in the DP, because at these timescales the current acceleration is the dominant predictor of near-term position. Tables A.1 - A.10 show the mean and standard deviation of error for each method.

3.4.4.3 Ghost Vehicle Representation for Lane-Changing Traffic

The Method 2 described in Section 3.4.4.2 assumes all surrounding vehicles maintain their current lane throughout the prediction horizon. This assumption breaks down when a vehicle within sensor range is actively executing a lane change at time step k. In this case, the vehicle's true lane occupancy is ambiguous, as it is physically transitioning between its origin lane l_i and its target lane $l_i^* = l_i \pm 1$, and it is uncertain whether, from the perspective of the ego vehicle's safety constraints, the vehicle should be treated as occupying the origin lane, the target lane, or both.
To handle this uncertainty, the prediction model adopts a ghost vehicle representation. Rather than assigning the lane-changing vehicle to a single lane, it is represented by two vehicles in the predicted trajectory set.
The original vehicle i remains assigned to the origin lane l_i. A ghost vehicle $\tilde{i}$ is placed on the target lane, with the same longitudinal position, speed, and acceleration as vehicle i.
This is equivalent to a probability problem, assuming that the lane changing vehicle simultaneously occupies both lanes at the moment of prediction. The ghost ensures that the DP evaluates the safety cost against the merging vehicle in the target lane. Without the ghost, the DP would perceive the target lane as unoccupied and potentially plan a lane change into a gap that is already being closed by the merging vehicle, or accelerate if the target lane of other vehicle is same as ego's current lane. The ghost vehicle thus acts as a worst-case occupancy assumption that makes the controller robust to the uncertainty in the vehicle's lateral position during the transition.
To identify a vehicle is in mid-lane-change, the lateral deviation from lane center is used. If this deviation exceeds a threshold, the controller assumes that that vehicle is changing its lane.

3.4.4.4 Sensor Assumptions

The prediction uses position, velocity, acceleration, length, and maximum speed of all vehicles within the sensor range. Also, the maximum acceleration and maximum deceleration are used for the rollout policy which will be discussed in the Section 3.5.
In a real deployment, position and velocity are directly measurable by sensors such as radar or lidar, and the acceleration can be derived from consecutive velocity measurements or from an inertial measurement unit. These quantities are therefore practically obtainable for all vehicles within the sensor range with sufficient accuracy.
Vehicle length and maximum speed, by contrast, are not directly observable from standard onboard sensors. However, for vehicles in the immediate neighborhood of the ego truck, within close radar or camera range, length can be estimated from point-cloud segmentation or bounding-box detection, and maximum speed may be inferred from vehicle class identification.
For vehicles further away, such estimation becomes less reliable. Making these quantities available for all vehicles within the full sensor range, particularly those not in the immediate neighborhood, may become feasible as Vehicle-to-Everything (V2X) communication infrastructure matures, which may enable direct broadcast of vehicle

type and capability parameters.
Maximum acceleration and maximum deceleration present a similar challenge. In this work they are read directly from the simulator, which has access to the ground-truth vehicle type parameters. In practice, these quantities are not directly measurable. A realistic approximation would be to estimate the vehicle type from its observed length, for example, classifying vehicles into passenger car, van, or heavy truck categories based on length thresholds, and then look up nominal acceleration and deceleration values from a predefined table for each class. Such an approach introduces uncertainty, but provides physically plausible bounds that are sufficient for the Krauss safe-speed computation in the rollout policy (see Section 3.5.1). However, this uncertainty is addressed in 3.6 by considering a gaussian noise in our measurements.

3.4.5 Reachable State Space

To maintain computational efficiency within the DP framework, the algorithm only considers reachable states at each step h of the prediction horizon H. Given the current state x_h and the physical limits of the vehicle, the set of reachable states X_{h+1} is constrained by the longitudinal and lateral dynamics.

3.4.5.1 Longitudinal Position Reachability

The longitudinal search space is bounded by the vehicle's maximum acceleration a_{max} and minimum acceleration (maximum braking) a_{min}. For each horizon step $h \in \{1, \dots, H\}$, the maximum and minimum reachable longitudinal positions are calculated as follows:

$$s_{max,h} = s_0 + h \cdot v_0 \cdot \Delta t + \frac{1}{2} a_{max} (h\Delta t)^2 + 2S_{STEP} \tag{3.29}$$

$$s_{min,h} = s_0 + h \cdot v_0 \cdot \Delta t + \frac{1}{2} a_{min} (h\Delta t)^2 - 2S_{STEP} \tag{3.30}$$

where s_0 and v_0 are the vehicle's position and velocity at the beginning of the planning horizon.
This ensures that the vehicle only evaluates trajectories that are physically achievable, and prevents unrealistic longitudinal shifts. The reason that $2S_{STEP}$ was added is due the rounding problem during grid indexing in Python. This ensures that the conversion from continuous coordinates to discrete indices does not result in out-of-bounds errors or unwanted exclusion of valid trajectories.

3.4.5.2 Velocity Reachability and Initial State

The reachable velocity at each step h is constrained by the vehicle's acceleration limits and the global system boundaries $[v_{min}, v_{max}]$. To account for discretization resolution V_{STEP}, a numerical safety margin is applied:

$$v_{max,h} = \min(v_{max},\ v_0 + a_{max}(h\Delta t) + 2V_{STEP}) \tag{3.31}$$

$$v_{min,h} = \max(v_{min},\ v_0 + a_{min}(h\Delta t) - 2V_{STEP}) \tag{3.32}$$

3.4.5.3 Lateral Reachability

The reachable lane l_{h+1} should also be within available lanes:

$$l_{h+1} \in \{l_h - 1, l_h, l_h + 1\} \cap \{0, \ldots, N_{lanes} - 1\} \tag{3.33}$$

3.4.5.4 Initial State Constraint

To ensure the continuity of the optimized trajectory from the current physical observation, the search space at the initial time step ($h = 0$) is strictly anchored to the vehicle's observed state:

$$x_0 = [s_0, l_0, v_0, c_0]^T \tag{3.34}$$

By enforcing this initial condition, the Dynamic Programming algorithm effectively roots the decision tree in the real-time sensor data provided by the SUMO environment. This ensures that the first control action u_0^* is directly applicable to the current simulation step.
Combined with the longitudinal, velocity, and lateral reachability pruning, this formulation significantly reduces the number of state-transition evaluations required.

3.5 Rollout-Based Policy Implementation

The rollout-based policy implemented for this project approximates the optimal control policy by evaluating a set of candidate actions (acceleration and lane change decision) over a long (but not infinite) horizon H_{roll}.
Unlike a complete ADP approach which utilizes a pre-calculated cost-to-go function $J(s)$ derived from the Bellman equation via backward dynamic programming, this implementation does not use a precomputed value function terminal cost. Instead, the terminal cost is approximated online through rollout simulation. It selects the immediate action with minimum stage cost by a heuristic base policy, and accumulate them over the prediction horizon, without considering a terminal cost for the final state in the horizon.

3.5.1 System Dynamics and Implementation

The rollout policy utilizes SUMO's built-in simulation engine as the base policy to propagate the vehicle state $x = [s, l, v, c]^T$ over the horizon H_{roll}. However, it should be noted that implementing SUMO's exact logic can make the computation much slower. Therefore, in order for our prediction to be close to the behavior of SUMO's built-in dynamics at each step of the horizon, the rollout base policy re-implements the Krauss car-following model [45], matching SUMO's longitudinal dynamics without invoking the SUMO simulator at runtime.
The Krauss model determines the maximum safe speed v_{safe} for a vehicle based on the front gap g and the lead vehicle speed v_l. The safe speed for a surrounding vehicle i following its leader is:

$$v_{safe}(t) = -b\tau + \sqrt{(b\tau)^2 + v_l^2(t) + 2bg(t)} \tag{3.35}$$

where τ is the reaction time and b is the maximum deceleration. In this way, the surrounding vehicles will propagate at each step of rollout.
For the ego vehicle, a modified safe-speed gap using the MPC headway parameters D_0 and T_{head} is used instead of the SUMO's default minimum gap, ensuring consistency with the DP safety cost:

$$g(t) = g_{actual} - D_0 \tag{3.36}$$

$$\tau = T_{head} \tag{3.37}$$

Surrounding vehicles in the rollout are assumed to maintain their current lane throughout the rollout horizon. This is a simplifying assumption, in the actual simulation surrounding vehicles may change lanes, and it causes the rollout to be optimistic about situations where a vehicle cuts in front of or behind ego during the rollout extension. Addressing this limitation by adding a simplified lane-change model to the rollout is left for future work.
To simplify the rollout policy, it utilizes an Immediate Lane Assignment logic, similar to the backward DP formulation (Section 3.4.3), where it updates the vehicle's lane state to the target lane ($l_{h+1} = l_h + \Delta l_h$) at the exact moment a lane change command is triggered. However, lane change is allowed only if three conditions are met:

1. The gap from the leader on the target lane is safe.
2. The gap from the follower on the target lane is safe.
3. The allowed speed on the target lane is better (closer to v_{ref}).

The allowable velocity v_{allow} at each step of the rollout is calculated by computing the safe speed for the current lane and also, if a lane change is considered, the safe speed for the target lane. The target velocity is taken as the minimum of the reference velocity v_{ref} and the safe speeds across considered lanes:

$$v_{target} = \min(v_{ref}, v_{safe\ current}, v_{safe\ target}) \tag{3.38}$$

The acceleration a_k is then calculated to reach this target velocity, bounded by maximum acceleration limits. Finally, the state is updated using kinetic equations similar to SUMO's internal state update mechanisms, Euler's method, which is different from equations 3.3-3.6:

$$v_{k+1} = \begin{cases} v_{max} & \text{if } \ v_k + a_k \Delta t \geq v_{max} \\ v_{min} & \text{if } \ v_k + a_k \Delta t \leq v_{min} \\ v_k + a_k \Delta t & \text{otherwise} \end{cases} \tag{3.39}$$

$$s_{k+1} = s_k + v_{k+1} \Delta t \tag{3.40}$$

$$l_{k+1} = l_k + \Delta l_k \tag{3.41}$$

$$c_{k+1} = \begin{cases} C_{steps} - 1 & \text{if } c_k = 0 \ \& \ \Delta l_k \neq 0 \\ c_k - 1 & \text{if } c_k > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.42}$$

At each step, the rollout stage cost is calculated, using the same stage cost formulation in Eq. 3.8. Then, the approximated terminal cost $\hat{q}_{roll}$ is computed by accumulating all stage costs in the rollout horizon.

3.5.2 Discount Factor

Both the backward DP and the rollout base policy apply an exponential discount factor γ to stage costs, weighting immediate costs more heavily than costs further into the future. The discounted MPC objective becomes:

$$J^* = \min_{u_0,\ldots,u_{H-1}} \sum_{h=0}^{H-1} \gamma^h \cdot g(x_h, u_h) + q(x_H) \tag{3.43}$$

and the discounted rollout cumulative cost starting from the terminal state x_H is:

$$\hat{q}_{roll}(x_H) = \sum_{j=0}^{H_{roll}-1} \gamma^j \cdot g(x_{H+j},\, \pi_{roll}(x_{H+j})) \tag{3.44}$$

where π_{roll} is the rollout base policy. Applying the same γ to both MPC and rollout has two effects. First, it provides a principled connection between the finite-horizon DP and the rollout extension, where the cost at step h within the horizon carries weight γ^h, and a cost incurred at rollout step j beyond the horizon carries total weight $\gamma^H \cdot \gamma^j = \gamma^{H+j}$. The discounting is therefore continuous and consistent through the whole horizon $H + H_{roll}$. Second, it aligns the MPC cost formulation with the RL value function used in the hybrid controller, where the DQN agent was trained with discount factor γ, so its Q-values represent a γ-discounted infinite-horizon cost.

3.6 Stochastic Extension

3.6.1 Noise Model

The trajectory prediction method described in Section 3.4.4 (Constant-acceleration extrapolation) was evaluated against ground-truth vehicle positions across all prediction steps $h = 1, \ldots, H$. The results, summarized in Tables A.6 - A.10, show that the mean prediction error is near zero with a very small standard deviation at all horizon steps. This indicates that the system does not have a large process noise in the vehicle dynamics and vehicles largely follow constant-acceleration trajectories over the short horizon, e.g. $H \leq 5$ steps (1 second). Therefore, to make our method more realistic, we introduced measurement uncertainty in the sensor observations at time step k.
This uncertainty is modeled as additive Gaussian measurement noise applied once at $h = 0$ to the observed position, speed, and acceleration of each surrounding vehicle i within the sensor range:

$$\tilde{s}_{i,0} = s_{i,0} + \epsilon_s^i, \quad \epsilon_s^i \sim \mathcal{N}(0, \sigma_s^2) \tag{3.45}$$

$$\tilde{v}_{i,0} = v_{i,0} + \epsilon_v^i, \quad \epsilon_v^i \sim \mathcal{N}(0, \sigma_v^2) \tag{3.46}$$

$$\tilde{a}_{i,0} = a_{i,0} + \epsilon_a^i, \quad \epsilon_a^i \sim \mathcal{N}(0, \sigma_a^2) \tag{3.47}$$

where σ_s, σ_v, and σ_a are the measurement noise standard deviations for position, speed, and acceleration respectively.

The noises are independent across vehicles and across the three quantities. The noisy observations $(\tilde{s}_{i,0}, \tilde{v}_{i,0}, \tilde{a}_{i,0})$ replace the ground-truth values as inputs to the constant-acceleration prediction model, after which the trajectory is propagated deterministically forward for $h = 1, \ldots, H$ steps using Equations 3.27 and 3.28.

3.6.2 Propagation of Uncertainty Through the Prediction Horizon

According to Equations 3.27 and 3.28, the prediction model uses a discrete Euler integration where position at each step is updated using the speed from the current step, and speed is then updated using the fixed initial acceleration $a_{i,0}$.
By expanding the position recursive function, the predicted position at step h is:

$$s_{i,h} = s_{i,0} + h \cdot v_{i,0} \cdot \Delta t + \frac{h(h-1)}{2} \cdot a_{i,0} \cdot (\Delta t)^2 \tag{3.48}$$

By substituting the noisy initial observations ($\tilde{s}_{i,0}$, $\tilde{v}_{i,0}$, $\tilde{a}_{i,0}$) from Eqs. 3.45 - 3.47, the predicted position at step h is:

$$\tilde{s}_{i,h} = \tilde{s}_{i,0} + h \cdot \tilde{v}_{i,0} \cdot \Delta t + \frac{h(h-1)}{2} \cdot \tilde{a}_{i,0} \cdot (\Delta t)^2 \tag{3.49}$$

Since $\tilde{s}_{i,0}$, $\tilde{v}_{i,0}$, and $\tilde{a}_{i,0}$ are mutually independent Gaussian random variables, $\tilde{s}_{i,h}$ is also Gaussian. Its mean and variance at step h are:

$$\mu^i_{s,h} = s_{i,0} + h \cdot v_{i,0} \cdot \Delta t + \frac{h(h-1)}{2} \cdot a_{i,0} \cdot (\Delta t)^2 \tag{3.50}$$

$$(\sigma^i_{s,h})^2 = \sigma^2_{s,h} = \sigma^2_s + (h \cdot \Delta t)^2 \cdot \sigma^2_v + \left(\frac{h(h-1)}{2} \cdot (\Delta t)^2 \right)^2 \cdot \sigma^2_a \tag{3.51}$$

The standard deviation $\sigma_{s,h}$ grows with h, reflecting increasing positional uncertainty as the prediction horizon extends. For the specific values $\Delta t = 0.2\,s$ and $h = 1, \ldots, 5$, the acceleration contributions $\frac{h(h-1)}{2}(\Delta t)^2$ are 0, 0.04, 0.12, 0.24, 0.400 meters per unit of σ_a, showing that acceleration uncertainty becomes non-negligible at ≥ 3 steps.
The front bumper-to-bumper gap between ego at deterministic position s_h and vehicle i ahead is:

$$\tilde{g}^i_{front,h} = \tilde{s}_{i,h} - s_h - \ell_i \tag{3.52}$$

Since $s_{ego,h}$ and ℓ_i are deterministic, the gap $\tilde{g}_{i,h}$ is Gaussian with:

$$\mu^i_{g_{front},h} = \mu^i_{s,h} - s_h - \ell_i, \qquad \sigma^i_{g_{front},h} = \sigma_{s,h} \tag{3.53}$$

The velocity of vehicle i at step h under constant acceleration extrapolation is:

$$\tilde{v}_{i,h} = \tilde{v}_{i,0} + h \cdot \tilde{a}_{i,0} \cdot \Delta t \tag{3.54}$$

Since $\tilde{v}_{i,0}$ and $\tilde{a}_{i,0}$ are independent Gaussian random variables, $\tilde{v}_{i,h}$ is also Gaussian with:

$$\mu^i_{v,h} = v_{i,0} + h \cdot a_{i,0} \cdot \Delta t \tag{3.55}$$

$$(\sigma^i_{v,h})^2 = \sigma^2_{v,h} = \sigma^2_v + (h \cdot \Delta t)^2 \cdot \sigma^2_a \tag{3.56}$$

The velocity uncertainty $\sigma_{v,h}$ is used in the robust rear safety and collision costs, as described in Sections 3.4.2.2 and 3.4.2.5. For the rear bumper-to-bumper gap to a following vehicle i behind ego on the target lane:

$$\tilde{g}^i_{back,h} = s_{ego,h} - \ell_{ego} - \tilde{s}_{i,h} \tag{3.57}$$

which is Gaussian with:

$$\mu^i_{g_{back},h} = s_{ego,h} - \ell_{ego} - \mu^i_{s,h}, \qquad \sigma^i_{g_{back},h} = \sigma_{s,h} \tag{3.58}$$

3.6.3 Chance-Constrained Safety

The deterministic safety constraint $g_{front} \geq d_{safe,front}$ from Section 3.4.2.2 is replaced by a probabilistic requirement. It requires that the probability of the gap remaining above the safe distance exceeds a confidence level $1 - \epsilon$:

$$\mathbb{P}\left(\tilde{g}^i_{front,h} \geq d_{safe,front}\right) \geq 1 - \epsilon \tag{3.59}$$

where $\epsilon \in (0, 1)$ is the risk tolerance. Following the deterministic equivalent derivation in Section 2.3.4, this probabilistic constraint is converted to a tightened deterministic constraint on the mean gap. Standardizing the gap:

$$z = \frac{\tilde{g}^i_{front,h} - \mu^i_{g_{front},h}}{\sigma_{s,h}} \sim \mathcal{N}(0, 1) \tag{3.60}$$

the chance constraint becomes:

$$\Phi\left(\frac{\mu^i_{g_{front},h} - d_{safe,front}}{\sigma_{s,h}}\right) \geq 1 - \epsilon \tag{3.61}$$

Applying the inverse CDF Φ^{-1} yields the deterministic equivalent:

$$\mu^i_{g_{front},h} \geq d_{safe,front} + \Phi^{-1}(1 - \epsilon) \cdot \sigma_{s,h} \tag{3.62}$$

This is equivalent to replacing the nominal safe distance with a tightened safe distance that grows with horizon step h to account for the increasing positional uncertainty:

$$d^{robust}_{safe,front}(h) = D_0 + T_{head} \cdot v_h + \Phi^{-1}(1 - \epsilon) \cdot \sigma_{s,h} \tag{3.63}$$

where the stochastic safety buffer $\Phi^{-1}(1 - \epsilon) \cdot \sigma_{s,h}$ expands with h as positional uncertainty accumulates.
For a confidence level of 99% ($\epsilon = 0.01$), $\Phi^{-1}(0.99) \approx 2.33$, meaning the mean gap must remain 2.33 times the standard deviations above the nominal safe distance. The quadratic safety cost in Eq. 3.13 is then evaluated against $d^{robust}_{safe,front}(h)$ rather than $d_{safe,front}$:

$$Cost^{robust}_{safe,front}(h) = w_{front} \cdot \max\left(0, d^{robust}_{safe,front}(h) - \mu^i_{g_{front},h}\right)^2 \tag{3.64}$$

The same logic applies to the rear gap during a lane change. The robust tightened rear safe distance and cost are:

$$d^{robust}_{safe,back}(h) = D_0 + T_{head} \cdot \mu^i_{v,h} + \Phi^{-1}(1-\epsilon) \cdot \sigma_{s,h} \tag{3.65}$$

$$Cost^{robust}_{safe,back}(h) = w_{back} \cdot \max\Big(0,\ d^{robust}_{safe,back}(h) - \mu^i_{g_{back},h}\Big)^2 \tag{3.66}$$

Note that $\mu^i_{v,h}$ replaces the deterministic v_{follow} in the safe distance computation, reflecting that the follower's speed is itself uncertain.
The collision penalty condition from Eq. 3.17 follows the same probabilistic logic, too. In the deterministic formulation, the collision penalty P_{col} fires when the front gap is non-positive ($g^i_{front,h} \leq 0$). Under the sensor noise model, the gap $\tilde{g}_{i,h}$ is a Gaussian random variable, so with the same logic:

$$Cost^{robust}_{col,front}(h) = \begin{cases} P_{col} & \text{if } \mu^i_{g_{front},h} < \Phi^{-1}(1-\epsilon) \cdot \sigma_{s,h} \\ 0 & \text{otherwise} \end{cases} \tag{3.67}$$

The robust rear collision condition combines the physics-based braking distance threshold with the stochastic buffer:

$$Cost^{robust}_{back,col}(h) = \begin{cases} P_{col} & \text{if } \mu^i_{g_{back},h} < \dfrac{(\mu^i_{v,h} - v_h)^2}{2\,a_{break}} + \Phi^{-1}(1-\epsilon) \cdot \sigma_{s,h} \\ 0 & \text{otherwise} \end{cases} \tag{3.68}$$

The uncertainty associated with the follower vehicle's velocity $\dfrac{(\mu^i_{v,h} - v_h)^2}{2\,a_{break}}$ is neglected. This approximation is adopted to maintain a simple and computationally efficient constraint formulation. Furthermore, $\dfrac{(\mu^i_{v,h} - v_h)^2}{2\,a_{break}}$ is not derived from a probabilistic collision model but serves as an additional conservative safety margin. The stochastic treatment is therefore restricted to the longitudinal position prediction, which constitutes the dominant source of collision risk uncertainty in the considered scenario.
In all cases, the effect of the sensor noise model is to add the same stochastic buffer $\Phi^{-1}(1-\epsilon) \cdot \sigma_{s,h}$ to the corresponding threshold. Since $\sigma_{s,h}$ grows with h according to Eq. 3.51, the controller becomes progressively more conservative at longer prediction steps, naturally reflecting the increasing uncertainty in the predicted gap as the horizon extends.

3.6.4 Model Parameters

Table A.16 in Appendix A.2 summarizes the physical constants, cost weights, and noise parameters that govern the MPC, rollout, and stochastic extension formulations presented in this chapter.
The cost weights and penalty values were selected through empirical tuning. The collision penalty $P_{\text{col}} = 10^8$ is set several orders of magnitude above the maximum accumulated safety cost over a full horizon to ensure that any predicted collision dominates the DP objective regardless of the horizon length or the stage costs at other steps.

3.7 Reinforcement Learning Formulation

3.7.1 Problem Statement

The autonomous highway driving task is formulated as a MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. The transition dynamics $\mathcal{P}$ are implicitly defined by the SUMO simulator and are not available in closed form. The agent learns a policy that maximizes the expected discounted cumulative reward through interaction with the simulated environment.

3.7.2 State Space

At each simulation step k, the agent observes a 14-dimensional state vector $z_k \in \mathcal{S} \subset \mathbb{R}^{14}$:

$$\begin{aligned} z_k = &[v_k,\ l_k,\ g_{F,k},\ \Delta v_{F,k},\ g_{FL,k},\ \Delta v_{FL,k},\ g_{FR,k},\\ &\Delta v_{FR,k},\ g_{BL,k},\ \Delta v_{BL,k},\ g_{BR,k},\ \Delta v_{BR,k},\ p_{LC,k},\ p_{CD,k}]^T \end{aligned} \tag{3.69}$$

where:

- $v_k \in [v_{min}, v_{max}]$: ego longitudinal velocity (m/s).
- $l_k \in \{0, 1, \ldots, N_{lanes} - 1\}$: current lane index, with $l = 0$ the rightmost lane.
- $g_{F,k}$, $g_{FL,k}$, $g_{FR,k} \in [0, d_{sensor}]$: bumper-to-bumper gaps to the nearest leader in the current, left, and right lanes respectively (m). Set to d_{sensor} when no vehicle is in range.
- $g_{BL,k}$, $g_{BR,k} \in [0, d_{sensor}]$: Bumper-to-bumper gaps to the nearest follower in the left and right adjacent lanes respectively (m). Set to d_{sensor} when no vehicle is in range.
- $\Delta v_{F,k}$, $\Delta v_{FL,k}$, $\Delta v_{FR,k}$: Relative velocities to front vehicles $(v_{leader} - v)$, set to 0 when no vehicle is in sensor range (m/s).
- $\Delta v_{BL,k}$, $\Delta v_{BR,k}$: Relative velocities to rear vehicles $(v_{follower} - v)$, set to 0 when no vehicle is in sensor range (m/s).
- $p_{LC,k} \in [0, 1]$: normalized lane-change transition progress, derived from the unified counter c_k as:

$$p_{LC,k} = \frac{\max(0,\ c_k - C_{start})}{N_{steps}} \tag{3.70}$$

  This is nonzero only during active physical transition.
- $p_{CD,k} \in [0, 1]$: normalized cooldown progress, derived from the same unified counter as:

$$p_{CD,k} = \frac{c_k}{\max(1,\ C_{start} - 1)} \tag{3.71}$$

  This is nonzero during both cooldown and transition.

C_{start} and C_{start} are defined in Section 3.3. Although $p_{LC,k}$ and $p_{CD,k}$ are presented as two separate features, they are both derived from the single unified counter c_k defined in Section 3.3. Their separation provides the network with explicit signal

about the current phase of the maneuver cycle. The two features are mutually exclusive in their nonzero regions.
The longitudinal position s_k is not included in the state. This is acceptable because the highway scenario is translation-invariant, since the future cost depends on the local traffic configuration around the ego vehicle rather than its absolute road position.

3.7.3 Action Space

Similar to MPC configuration, the action space is a joint discrete space over longitudinal acceleration and lateral lane change commands:

$$\mathcal{A} = \mathcal{A}_{accel} \times \mathcal{A}_{lateral} = \{a_1, \ldots, a_{N_a}\} \times \{-1, 0, +1\} \tag{3.72}$$

$\mathcal{A}_{accel}$ and $\mathcal{A}_{lateral}$ have similar values to a_k and Δl_k in Section 3.3. At each step, the DQN selects a single action index that encodes both the acceleration command and the lateral command simultaneously.

3.7.4 Reward Function

The reward is defined as the negative scaled stage cost:

$$r_k = -r_{scale} \cdot g(z_k, u_k) \tag{3.73}$$

where $r_{scale} = 10^{-5}$ is a constant scaling factor and $g(z_k, u_k)$ is the stage cost defined in Eq. 3.8. Using the same cost structure for both RL training and MPC optimization is essential, because it ensures that the learned value function represents a valid approximation of the MPC tail cost, and making the hybrid integration in Section 3.8 principled.
It should be noted that the collision reward is applied upon episode termination due to collision:

$$r_{collision} = -P_{collision} \cdot r_{scale} = -10^8 \times 10^{-5} = -1000 \tag{3.74}$$

This strongly distinguishes collision states from near-collision states in the value function, while keeping the reward numerically stable relative to the normal per-step range.

3.7.5 Optimization Objective

The agent learns the optimal action-value function:

$$Q^*(x, u) = \max_{\pi} \mathbb{E}_{\pi} \left[\sum_{i=0}^{\infty} \gamma^i r_{k+i} \,\middle|\, x_k = x,\ u_k = u \right] \tag{3.75}$$

satisfying the Bellman optimality equation:

$$Q^*(x, u) = \mathbb{E} \left[r + \gamma \max_{u'} Q^*(x', u') \,\middle|\, x, u \right] \tag{3.76}$$

The Q-function is approximated by a deep neural network with parameters θ, updated by minimizing the Bellman residual:

$$\mathcal{L}(\theta) = \mathbb{E}_{(x,u,r,x')\sim\mathcal{D}}\left[(y - Q(x,u;\theta))^2\right] \tag{3.77}$$

where $\mathcal{D}$ is the experience replay buffer and the target value is:

$$y = r + \gamma \max_{u'} Q(x', u'; \theta^-) \tag{3.78}$$

with θ^- denoting the periodically updated target network parameters. The training environment is described in Section 4.2.1.

3.7.6 Hyperparameters

Table A.17 in Appendix A.2 summarizes the hyperparameters used for DQN training. The discount factor γ, exploration rate and speed weight are the primary variables investigated across training runs, as discussed in Section 4.2. The reward scaling factor r_{scale} is set to ensure numerical stability of the Q-values relative to the raw stage costs, which can reach 10^8 in collision scenarios.

3.8 Hybrid MPC-RL Formulation

3.8.1 Motivation

As discussed in 2.1.4, setting the terminal cost to zero ($q(x_H) = 0$) for MPC controller defined in Section 3.3 implies that no cost accumulates beyond the prediction horizon H. In practice it causes the controller to be myopic, which may, for example, accelerate to v_{ref} at the end of the horizon without anticipating a slow-moving vehicle just outside its lookahead window, or miss an overtaking opportunity that only becomes apparent several seconds ahead.
The RL agent, on the other hand, implicitly learns a value function $V^\pi(z)$ that encodes long-horizon consequences through training experience. However, a pure RL policy lacks the constraint satisfaction and online optimization over a prediction horizon that the MPC provides at each step.
The hybrid controller combines both components, where the MPC provides short-horizon trajectory optimization with explicit safety constraint enforcement, while the learned RL value function replaces the zero terminal cost with a data-driven estimate of the cost-to-go beyond the horizon. This is the core contribution of the thesis.

3.8.2 Terminal Cost Derivation

Since the RL reward is defined as $r_k = -r_{scale} \cdot g(x_k, u_k)$, the optimal Q-function satisfies:

$$Q^*(x,u) \approx -r_{scale} \sum_{i=0}^{\infty} \gamma^i g(x_{k+i}, u_{k+i}) \tag{3.79}$$

The corresponding state-value function is:

$$V^*(x) = \max_{u \in \mathcal{A}} Q^*(x, u) \tag{3.80}$$

To recover the cost-to-go in the raw MPC cost space, the terminal cost is defined as:

$$\hat{q}(x_H) = -\frac{V^*(x_H)}{r_{scale}} \cdot \alpha = -\frac{\max_u Q(x_H, u; \theta)}{r_{scale}} \cdot \alpha \tag{3.81}$$

where $\alpha \geq 0$ is a weighting factor controlling the influence of the learned terminal cost relative to the MPC stage costs accumulated over the horizon. With $\alpha = 1.0$ the terminal cost is on the same numerical scale as the MPC stage costs, providing Bellman-consistent integration.
As discussed in Sections 2.4 and 3.5.2, a practical consideration is that the RL value function approximates a *discounted* cost-to-go. The terminal cost recovered in Eq. (3.81) therefore represents

$$\hat{q}(x_H) \approx \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i g(x_{k+i}, u_{k+i})\right] \tag{3.82}$$

rather than an exact undiscounted tail cost.

3.8.3 Hybrid Optimization Objective

The hybrid controller replaces the zero terminal cost in the MPC objective with the RL estimate:

$$J^* = \min_{u_0, \ldots, u_{H-1}} \sum_{h=0}^{H-1} g(x_h, u_h) + \hat{q}(z_H) \tag{3.83}$$

subject to the same system dynamics and constraints as in Section 3.3. This is solved via Backward DP with the only modification being the initialization of the terminal cost array:

$$J[H,\ s,\ l,\ v,\ c] = \hat{q}(z_H) \quad \forall\ (s, l, v, c) \in \mathcal{X}_{reach} \tag{3.84}$$

where $\mathcal{X}_{reach}$ is the reachable state space at horizon step H.

3.8.4 State Observation Bridge

The terminal DP state $x_H = [s_H, l_H, v_H, c_H]^T$ must be mapped to the 14-dimensional RL observation $z_H \in \mathbb{R}^{14}$ before querying the Q-network. This mapping is:

$$z_H = \phi(x_H,\ \mathcal{T}_H) \tag{3.85}$$

where $\mathcal{T}_H = \{(s_{i,H}, v_{i,H}, l_{i,H}, len_i)\}_{i=1}^{N}$ is the set of predicted traffic states at horizon step H, obtained from the traffic prediction model described in Section 3.4.4. The mapping ϕ constructs the 14-dimensional observation by:

1. Computing bumper-to-bumper front gaps and relative velocities to the nearest leader in each lane at step H.
2. Computing bumper-to-bumper rear gaps and relative velocities to the nearest follower in each adjacent lane at step H.
3. Deriving $p_{LC,H}$ and $p_{CD,H}$ from the terminal counter value c_H.

All gap measurements use bumper-to-bumper distances, consistent with how the RL agent was trained, by subtracting vehicle lengths from center-to-center distances. The terminal value is then evaluated as in Eq.3.81. This value is filled into $J[H]$ for all reachable (s, l, v, c) combinations before the backward DP sweep begins.

4

Implementation and Results

This chapter describes the implementation of the proposed control framework and presents the evaluation results. The simulation environment, software stack, and key assumptions are introduced first, followed by the RL training analysis, baseline evaluations, and a comparative assessment of all controller variants.

4.1 Simulation Platform and Implementation

4.1.1 Simulation Environment

As introduced in Section 3.1, all experiments are conducted using the SUMO platform. This section describes the specific highway scenario, vehicle models, traffic generation procedures, and simulation parameters used for evaluating the proposed controllers.

The simulation environment is a three-lane, one-way straight highway of length 30 km with a road speed limit of 35 m/s. It is also translation-invariant, i.e. no intersections, traffic lights, on-ramps, or road geometry changes are present. The ego vehicle is a heavy truck with length 12 m, maximum speed 33.33 m/s, and acceleration limits $a_{\min} = -2.0\,m/s^2$, $a_{\max} = 2.0\,m/s^2$. Surrounding traffic consists of a mix of passenger cars (length 4.8 m) and trucks (length 16.5 m) in a ratio of 9 : 1, spawned and maintained by a moving-window pool manager that keeps traffic density approximately constant at 30 vehicles/km around the ego vehicle. Each surrounding vehicle follows SUMO's default Krauss car-following model with reaction time $\tau = 1.0\,s$ and driver imperfection parameter $\sigma = 0.5$. Surrounding vehicle speeds are sampled uniformly from $[20, 30]\,m/s$ at spawn time. Lane changes by surrounding vehicles are permitted and governed by SUMO's LC2013 model [46]. If a vehicle steps outside of the moving window, another will be spawned from the other side.

The simulation time step is $\Delta t = 0.2\,s$, consistent with the MPC sampling time. Each evaluation episode runs for a maximum of 1500 steps (300 seconds) or until a collision is detected, whichever occurs first. The ego vehicle's lane-change duration is set to $T_{LC} = 2.0\,s$.

4.1.2 Software Stack

The controller and training pipeline are implemented in Python 3.12.1. The key dependencies are listed in Table 4.1.

Table 4.1: Software dependencies.

Package	**Version**	**Purpose**
SUMO	1.25.0	Traffic simulation
TraCI / traci	22	Simulator interface
Stable-Baselines3	2.8.0	DQN implementation
PyTorch	2.11.0	Neural network backend
Numba	0.64.0	JIT compilation of DP and rollout kernels

The backward DP kernel and rollout policy are implemented as Numba JIT-compiled functions with parallel execution over the state grid. This avoids the Python interpreter overhead at each grid point and reduces the per-step DP computation to 17.49 ms on average for a grid of approximately 4.63 million states.

4.1.3 Computational Setup

RL training was conducted on Volvo's internal compute cluster, which is managed through Kubernetes using the KubeRay operator, with Ray providing the distributed-execution framework. Training ran on a single Ray head node; no GPU-accelerated worker nodes were provisioned for these experiments. The resource allocation of the head container is summarized in Table 4.2.

Table 4.2: Compute resource allocation of the Ray head container used for RL training.

Resource	Allocation
CPU (limit)	4 vCPU
CPU (request)	1 vCPU
Memory (limit)	16 GB
Memory (request)	4 GB
GPU	None
Orchestration	Ray (KubeRay)

The container was allocated a maximum of four virtual CPUs and 16 GB of RAM, of which Ray reserved approximately 14.9 GiB for object storage and task execution. Training was therefore performed entirely on CPU. The absence of GPU acceleration was confirmed both by the container environment, in which the CUDA and NVIDIA device visibility variables were set to `none`, and by the unavailability of the `nvidia-smi` utility within the container. This CPU-only configuration influenced the choice of training algorithm and the wall-clock training times reported in the following sections.

4.1.4 Scope and Limitations

The experimental scope is intentionally restricted to isolate the effect of the terminal cost approximation method. Several simplifying assumptions are made that bound the conclusions of this work:

1. The highway scenario is a straight, homogeneous road with no geometric complexity. Real highway networks include curves, merges, exits, and variable speed limits, all of which would affect both the MPC formulation and the RL training distribution. The straight-road assumption ensures that longitudinal position is translation-invariant.
2. The ego vehicle model is a discrete-time kinematic point mass. Tire dynamics, engine response lag, and load-dependent braking performance are not modeled. The controller commands acceleration directly, whereas a real system would command throttle or brake pressure through a lower-level actuation layer.
3. Full observability within the sensor range is assumed. Position, speed, acceleration, length, and vehicle type parameters of all surrounding vehicles within $d_{sensor} = 150\,m$ are available without noise in the deterministic evaluation. The stochastic extension in Section 3.6 relaxes this assumption by introducing Gaussian measurement noise, but the noise model is simplified relative to real sensor characteristics.
4. The rollout base policy does not model lane changes by surrounding vehicles during the rollout horizon. As discussed in Section 3.5, this makes the rollout optimistic in scenarios where a surrounding vehicle cuts into ego's lane during the rollout extension.
5. The longitudinal control input is also discretized. The acceleration is selected from a fixed set of five uniformly spaced values over $[a_{\min}, a_{\max}]$. This discretization is a consequence of the backward DP formulation, which requires a finite action space for exhaustive enumeration. Finer discretization increases the grid size and computational cost quadratically. The five-point grid provides adequate resolution for the speed ranges considered but may produce slightly jerky longitudinal profiles.
6. By using the kinematic point mass model as mentioned in Chapter 3, the lateral dynamics are neglected entirely. A more physically accurate model for a heavy truck would be a bicycle model or a full planar rigid-body model, where it captures the coupling between longitudinal and lateral forces, load transfer under braking, and tire saturation. However, the control interface available in this work does not provide a wheel steering angle command, and therefore the lateral action is limited to a discrete lane-change command to the simulator, with the actual lateral trajectory executed autonomously by SUMO's internal lane-change model. There is therefore no steering input over which the controller has authority, making a bicycle model or any lateral dynamics model inapplicable at this control level. Incorporating continuous lateral control with a steering interface would require a fundamentally different actuation architecture and is left for future work.
7. The stochastic uncertainty model assumes constant Gaussian measurement noise independent of vehicle distance. In reality, the accuracy of radar, LiDAR, and camera based perception systems generally decreases with distance due to sensor resolution limits, occlusion, and environmental effects. Therefore, far away surrounding vehicles would typically have larger uncertainty than nearby vehicles. While the adopted constant noise model simplifies the analysis and allows controlled comparison between controller variants, it does

not fully capture the distance dependent characteristics of real autonomous driving perception systems.

4.2 RL Training Analysis

Many training runs were conducted to investigate the effect of the discount factor, exploration rate and the speed cost weight on the learned value function, and to select a checkpoint suitable to use in the Pure RL controller or hybrid controller's terminal cost. The final training hyperparameters are listed in Table A.17.

4.2.1 Training Environment

The agent was trained in a stochastic highway environment, matching the evaluation environment described in Section 4.1.1. The ego vehicle starts at a randomized position within the road at each episode to improve generalization. The stochasticity in traffic density, vehicle speeds, and ego start position ensures the learned value function generalizes across a range of traffic configurations rather than overfitting to a single scenario.

4.2.2 Experiment 1 ($\gamma = 0.995$ and $w_{speed} = 1$ and $d_{sensor} = 100$)

The first experiment used a discount factor of $\gamma = 0.995$, corresponding to an effective planning horizon of approximately 120 seconds. The training mean cumulative reward curve exhibited an oscillatory pattern. Figure 4.1 shows that this model has failed to converge. A notable phase transition occurred at 1.2 million steps, at which point the collision rate dropped from $70-80\%$ to zero, indicating that the agent had discovered safe behavior. The 1.4 million step checkpoint achieved the best holdout metric of the entire experiment (mean reward -9.1, zero collisions on 10 seeds).

However, visual inspection revealed that checkpoint is a degenerate stopping policy, since the agent had learned to decelerate to near-zero speed, which eliminates collisions by avoiding motion rather than by driving safely. This policy would result in a bad terminal cost for the hybrid controller, and it would force the DP toward decelerating the ego vehicle at the end of every planning horizon rather than guiding it toward efficient, safe states. To prevent this behavior, the checkpoint selection is done on a large holdout set rather than on the 10-seed evaluation used during training, since degenerate solutions may achieve near-perfect performance on small sets while failing to generalize.

Severe forgetting events (distancing from optimal point) occurred at 1.8 and 4.8 million steps, with holdout collision rates returning to 90% and 50% respectively, followed by divergence at $6.0 - 6.8$ million steps. Four candidate checkpoints were evaluated on 500 independent test seeds, and the results are presented in Table 4.3. The 2.2 million step checkpoint was selected as the best active-driving candidate, where it achieved a 3.6% collision rate and mean reward -100.1.

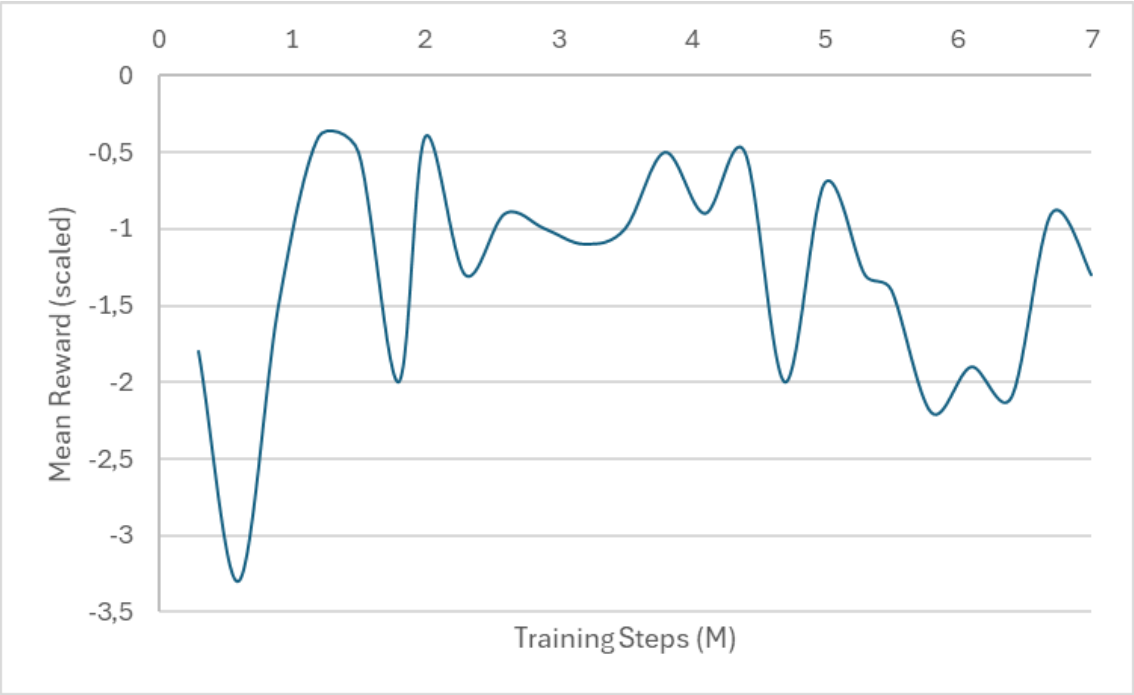


Figure 4.1: Mean cumulative reward on 10 holdout seeds at every 200,000-step checkpoint over 7 million training steps (Experiment 1, $\gamma = 0.995$).

Table 4.3: Pure RL evaluation on 500 test seeds for candidate checkpoints from Experiment 1 ($\gamma = 0.995$).

Checkpoint	Mean Reward	Collision Rate
$1.4M$	-110.9	10.2%
2.2M	**-100.1**	**3.6%**
$4.0M$	-146.7	4.4%
$4.8M$	-191.7	10.0%

4.2.3 Experiment 2 ($\gamma = 0.98$ and $w_{speed} = 1$ and $d_{sensor} = 100$)

By reducing the discount factor to $\gamma = 0.98$, the effective planning horizon shortened from approximately 120 seconds to 30 seconds. The final exploration rate was reduced from $\varepsilon_{end} = 0.05$ to $\varepsilon_{end} = 0.01$ to obtain cleaner Q-values at convergence. Figure 4.2 shows the per-checkpoint mean reward across the full 20 million steps of Experiment 2. The training curve still exhibits persistent oscillation throughout. However, despite the oscillation, a clear improving trend is visible compared to Experiment 1, which the ratio of zero-collision checkpoints increases from 47% in the first 5 million steps to 70% in the $10 - 15$ million step window. The policy does not fully converge within 20 million steps, which its likely causes are the stochastic traffic environment, the sparse collision reward signal, and residual exploration noise. The five best-performing zero-collision checkpoints, selected by mean reward on the 10-seed holdout are evaluated across 500 seeds, and the results are presented in Table 4.4. The $16.6M$ checkpoint is selected as the primary candidate for all hybrid

Figure 4.2: Mean cumulative reward on 10 holdout seeds at every 100,000-step checkpoint over 20 million training steps (Experiment 2, $\gamma = 0.98$).

and pure RL evaluations in this chapter, as it combines the lowest collision rate with the smallest reward standard deviation (54.3) among all competitive checkpoints, indicating more consistent behavior.

Table 4.4: Pure RL evaluation on 500 test seeds for the five candidate checkpoints from Experiment 2 ($\gamma = 0.98$).

Checkpoint	Mean Reward	Collision Rate
$7.4M$	-13.3	0.4%
$9.7M$	-16.3	0.2%
$11.3M$	-81.3	4.2%
$11.9M$	-43.9	1.2%
16.6M	**-14.7**	**0.2%**

Qualitative inspection of the trained policy reveals that the agent adopts a predominantly conservative driving strategy, as it follows the reference velocity when the road ahead is clear, maintains large safety gaps, and avoids lane changes unless there is a substantial speed advantage. This conservatism is not detrimental for use as a terminal cost, it provides a cautious value signal that discourages the hybrid controller from entering states with insufficient safety margins at the end of the planning horizon.

4.2.4 Experiment 3 ($\gamma = 0.90$ and $w_{speed} = 1$ and $d_{sensor} = 150$)

Following the finding that reducing the discount factor from $\gamma = 0.995$ to $\gamma = 0.98$ improved training stability in Experiment 2, Experiment 3 investigated whether a more aggressive reduction to $\gamma = 0.90$ would further benefit learning. A discount factor of 0.90 corresponds to an effective planning horizon of approximately 6 seconds, substantially shorter than the 30 seconds of Experiment 2. The sensor range was simultaneously extended from $d_{sensor} = 100\,m$ to $d_{sensor} = 150\,m$ to provide the agent with a wider observation window.

Figure 4.3 shows the mean cumulative reward on 10 holdout seeds across 20 million training steps. The reward curve exhibits severe and persistent oscillation throughout the entire experiment, with reward values ranging from near zero to below -600, substantially worse than Experiment 2. No clear improving trend is visible and the variance between consecutive checkpoints remains high across the full 20 million steps, indicating that the policy never stabilized. The magnitude of oscillation is greater than in Experiment 2 at the same training budget, suggesting that $\gamma = 0.90$ is too short-sighted for the highway driving task. With a planning horizon of only 6 seconds, the agent cannot adequately account for the consequences of lane changes and speed adjustments that unfold over longer timescales, making it difficult to learn a stable policy.

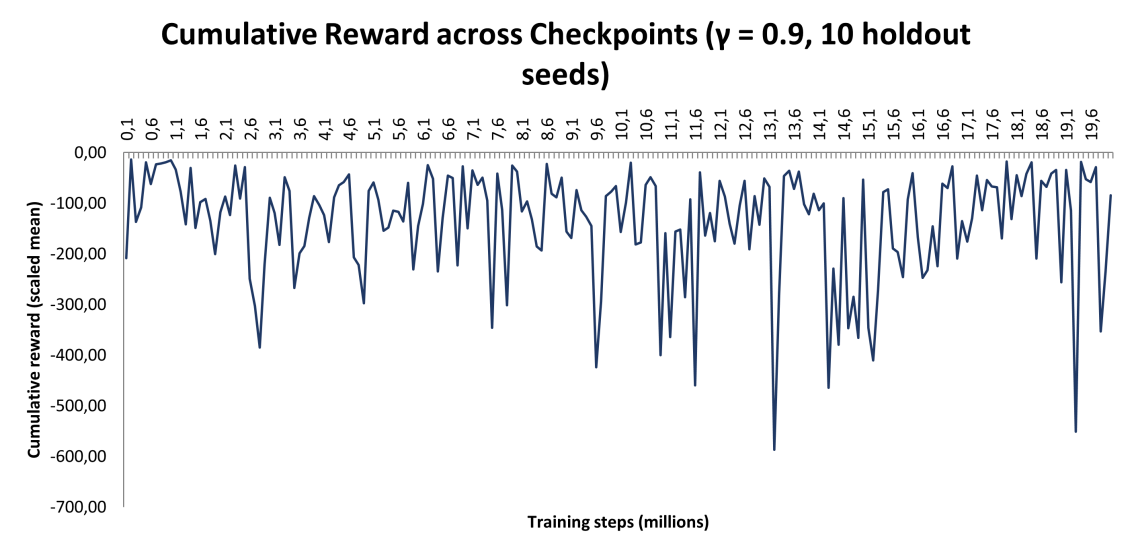


Figure 4.3: Mean cumulative reward on 10 holdout seeds at every 100,000-step checkpoint over 20 million training steps (Experiment 3, $\gamma = 0.90$).

4.2.5 Experiment 4 ($\gamma = 0.95$ and $w_{speed} = 1$ and $d_{sensor} = 150$)

Following the failure of $\gamma = 0.90$ in Experiment 3, Experiment 4 tested an intermediate discount factor of $\gamma = 0.95$, corresponding to an effective planning horizon of approximately 12 seconds. The sensor range was kept at $d_{sensor} = 150\,m$.

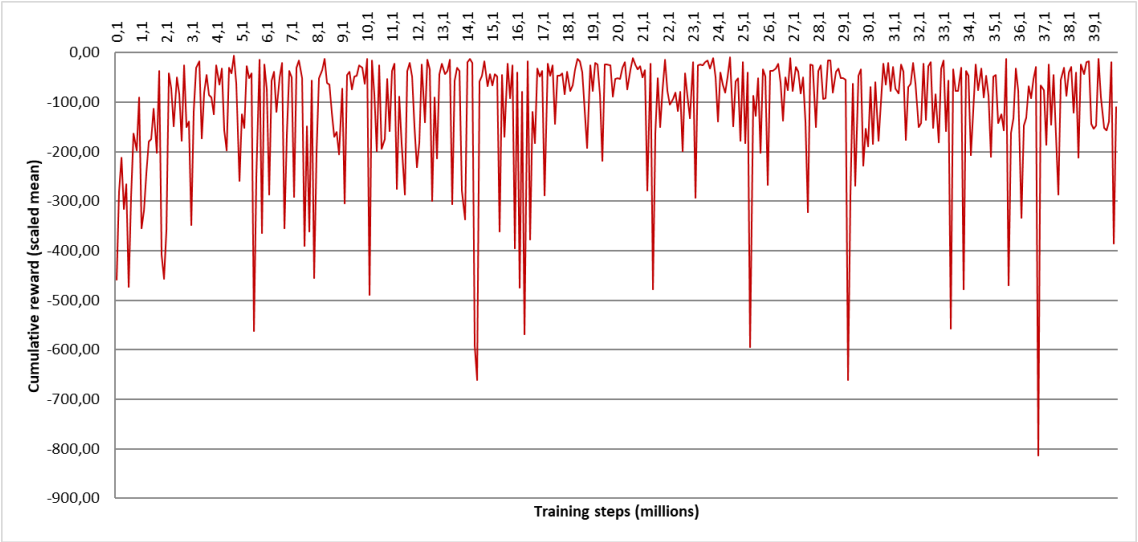


Figure 4.4: Mean cumulative reward on 10 holdout seeds at every 100,000-step checkpoint over 40 million training steps (Experiment 4, $\gamma = 0.95$).

Figure 4.4 shows the reward curve over 40 million steps. Compared to Experiment 3, the oscillation is mitigated and the curve is closer to zero reward, with 40% of checkpoints achieving a mean reward above -50. The best holdout checkpoint at 4.8 million steps reached a mean reward of -5.66 on 10 holdout seeds. However, evaluating the best checkpoints on 500 test seeds revealed a persistent instability, where the best checkpoint achieved a collision rate of 0.2% and mean reward of -45.6, and no checkpoint reached the combination of low collision rate and low cost achieved by Experiment 2. The curve also shows no clear convergence trend over 40 million steps, with large reward drops reappearing at 13M, 22M, and beyond 35M steps.

4.2.6 Experiment 5 ($\gamma = 0.98$ and $w_{speed} = 1$ and $d_{sensor} = 150$)

Having established $\gamma = 0.98$ as the most suitable discount factor through Experiments 1- 4, Experiment 5 retains these settings while extending the sensor range from $d_{sensor} = 100\,m$ to $d_{sensor} = 150\,m$. This wider observation window provides the agent with earlier visibility of approaching slow-moving vehicles, allowing the value function to encode longer-range traffic patterns. This experiment represents the final training configuration and the checkpoint selected from this experiment is used for all controller evaluations in this chapter.
Figure 4.5 shows the mean cumulative reward on 10 holdout seeds over 40 million training steps. The training exhibits persistent oscillation consistent with previous experiments, but 42% of checkpoints achieve a holdout reward above -50, an improvement over Experiment 4 (40%) at the same training budget. The best holdout reward of -4.40 is achieved at 2.6 million steps, though early checkpoints are typically unreliable indicators of true generalization performance. No clear convergence is observed, which is consistent with the findings of Experiments 1 and 2, and re-

flects the inherent difficulty of stabilizing Q-learning in a stochastic, multi-agent environment.

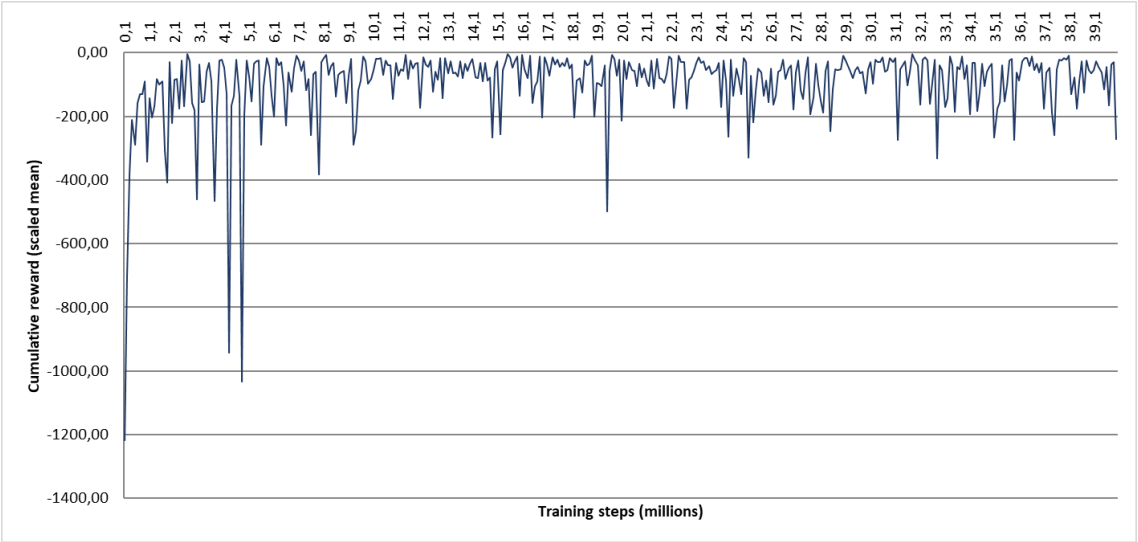


Figure 4.5: Mean cumulative reward on 10 holdout seeds at every 100,000-step checkpoint over 40 million training steps (Experiment 5, $\gamma = 0.98$, $d_{sensor} = 150\,m$).

The top-performing checkpoints were evaluated on 500 independent test seeds, with results presented in Table 4.5. Three checkpoints achieved zero collisions, 8.2M (-33.56, std 42.14), 11.4M (-38.34, std 53.65), and 16.1M (-31.46, std 42.44). The 16.1M checkpoint is selected as the primary candidate for all hybrid and pure RL evaluations, since it combines the lowest reward standard deviation (42.44) with a competitive mean reward (-31.46) and zero collisions, indicating the most consistent behavior across diverse seeds.

Table 4.5: Pure RL evaluation on 500 test seeds for the top-10 candidate checkpoints from Experiment 5 ($\gamma = 0.98$, $d_{sensor} = 150\,m$).

Checkpoint	Collisions	Collision %	Mean Reward	Std Reward
2.6M	4	0.8%	−42.59	97.15
7.0M	16	3.2%	−69.50	180.02
8.2M	0	0.0%	−33.56	42.14
11.4M	0	0.0%	−38.34	53.65
15.5M	1	0.2%	−31.38	60.92
16.1M	**0**	**0.0%**	**-31.46**	**42.44**
18.9M	2	0.4%	−39.03	84.99
19.7M	1	0.2%	−36.08	66.65
22.4M	5	1.0%	−47.55	137.06
31.8M	2	0.4%	−43.91	87.09

The qualitative behavior of the selected checkpoint is similar to the Experiment 2 candidate. The agent follows the reference velocity when the road ahead is clear, maintains large safety gaps, and avoids unnecessary lane changes. The mean speed

of $20.71\,m/s$ on 500 seeds remains below $v_{ref} = 25\,m/s$, due to the dense mixed traffic conditions.

4.2.7 Experiment 6 ($\gamma = 0.98$ and $w_{speed} = 3$ and $d_{sensor} = 150$)

Having established the base configuration in Experiment 5, Experiment 6 investigated the effect of increasing the speed tracking weight from $w_{speed} = 1$ to $w_{speed} = 3$. The motivation was to produce a value function that places greater emphasis on maintaining the reference velocity.
Figure 4.6 shows the mean reward curve over 40 million steps. The increased speed weight improves training stability noticeably, where 60% of checkpoints achieve a holdout reward above -50, compared to 42% in Experiment 5, and the best holdout reward of -4.31 is reached at 38.8 million steps. The curve shows fewer catastrophic drops compared to previous experiments, suggesting that the stronger speed signal provides a more consistent learning target.
Table 4.6 presents the 500-seed evaluation of the top candidate checkpoints. Three checkpoints achieved zero collisions, with the 18.3M checkpoint performing as best one, mean reward -29.60, and the lowest standard deviation (41.93) among all zero-collision checkpoints in any training experiments. This makes the 18.3M checkpoint the strongest Pure RL candidate overall and it is retained as an additional candidate for evaluation alongside the Experiment 5 checkpoint.

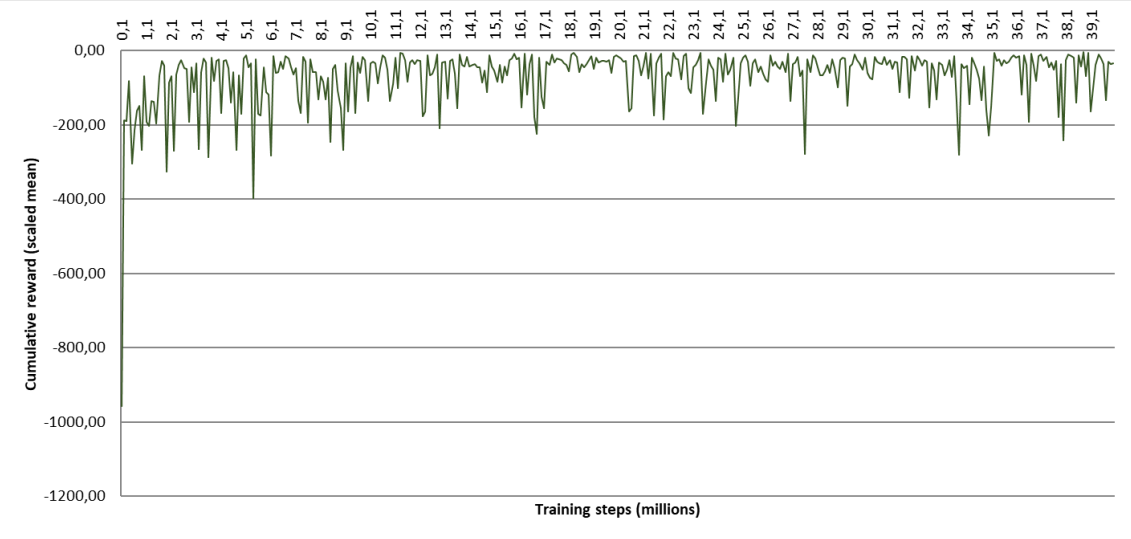


Figure 4.6: Mean cumulative reward on 10 holdout seeds at every 100,000-step checkpoint over 40 million training steps (Experiment 6, $\gamma = 0.98$, $w_{speed} = 3$).

However, a higher speed weight introduces a scale mismatch between the RL value function and the MPC cost function. The MPC uses $w_{speed} = 1$, meaning speed deviations are weighted equally with safety costs. When the 18.3M checkpoint is used as a terminal cost in the hybrid controller, the Q-values reflect a cost landscape where speed deviations are three times more penalized than in the MPC stage costs. This mismatch may cause the hybrid controller to make suboptimal decisions by over-prioritizing speed at the terminal state relative to safety.

Table 4.6: Pure RL evaluation on 500 test seeds for the top candidate checkpoints from Experiment 6 ($\gamma = 0.98$, $w_{speed} = 3$).

Checkpoint	Collisions	Collision %	Mean Reward	Std Reward
11.3M	1	0.2%	−36.90	65.85
16.3M	0	0.0%	−33.91	43.06
18.3M	**0**	**0.0%**	**-29.60**	**41.93**
21.2M	4	0.8%	−43.91	103.61
22.3M	0	0.0%	−35.45	45.88
23.4M	1	0.2%	−38.36	64.02
24.4M	5	1.0%	−52.11	106.12
35.2M	8	1.6%	−46.46	131.23
36.7M	16	3.2%	−71.60	181.31
38.8M	6	1.2%	−44.68	117.26

4.2.8 Experiment 7 ($\gamma = 0.98$ and $w_{speed} = 5$ and $d_{sensor} = 150$)

Experiment 7 further increased the speed weight to $w_{speed} = 5$ to investigate whether a stronger velocity signal could improve on Experiment 6. Figure 4.7 shows the mean reward curve over 40 million steps. While 50% of checkpoints achieve a holdout reward above −50, slightly below Experiment 6 (60%), the curve shows frequent deep drops below −400, indicating persistent instability. The best holdout reward of −2.63 was achieved at 3.1 million steps. However, evaluation on 500 test seeds reveals significantly worse safety, where no checkpoint achieved zero collisions, with the best result being 0.2% collision rate at 12.6M and 15.3M steps. The high and inconsistent collision rates across candidates indicate that at $w_{speed} = 5$ the agent over-prioritizes speed tracking relative to safety during training, leading to an aggressive policy that generalizes poorly.

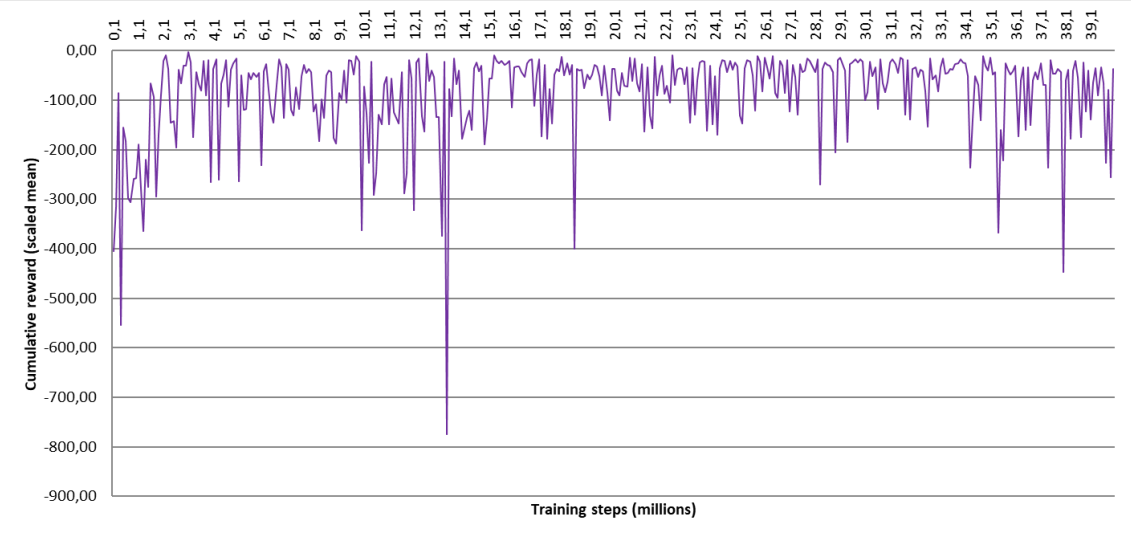


Figure 4.7: Mean cumulative reward on 10 holdout seeds at every 100,000-step checkpoint over 40 million training steps (Experiment 7, $\gamma = 0.98$, $w_{speed} = 5$).

4.2.9 Experiment 8 ($\gamma = 0.98$ and $w_{speed} = 10$ and $d_{sensor} = 150$)

Experiment 8 tested an even higher speed weight of $w_{speed} = 10$. Figure 4.8 shows the mean reward curve over 40 million steps. Despite the further increase in speed penalty, training stability recovered partially relative to Experiment 7. The curve oscillates less severely in the first 15 million steps, and 50% of checkpoints reach above -50 reward. The best holdout reward of -3.22 was achieved at 22.6 million steps. On 500 test seeds, one checkpoint at 22.6M achieved zero collisions with mean reward -36.47 and standard deviation 46.66. While this represents a valid zero-collision policy, it is inferior to the best candidates from Experiment 5 and Experiment 6 on both reward and consistency, confirming that $w_{speed} = 3$ from Experiment 6 represents the optimal speed weight among those investigated.

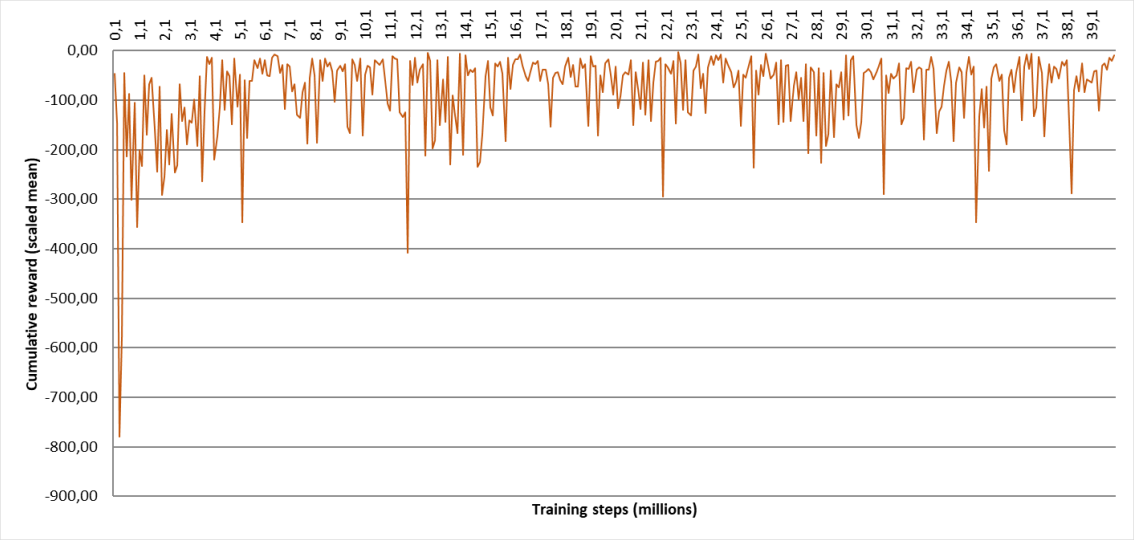


Figure 4.8: Mean cumulative reward on 10 holdout seeds at every 100,000-step checkpoint over 40 million training steps (Experiment 8, $\gamma = 0.98$, $w_{speed} = 10$).

4.3 Trajectory Prediction Error Analysis

Before comparing controller variants, the best method of trajectory prediction is chosen by quantifying the accuracy of each method as described in Section 3.4.4. This analysis serves two purposes:

1. It provides the empirical justification for selecting Method 2 (constant acceleration) as the trajectory prediction model.
2. It establishes the measurement noise parameters σ_s, σ_v, σ_a used in the stochastic extension of Section 3.6.

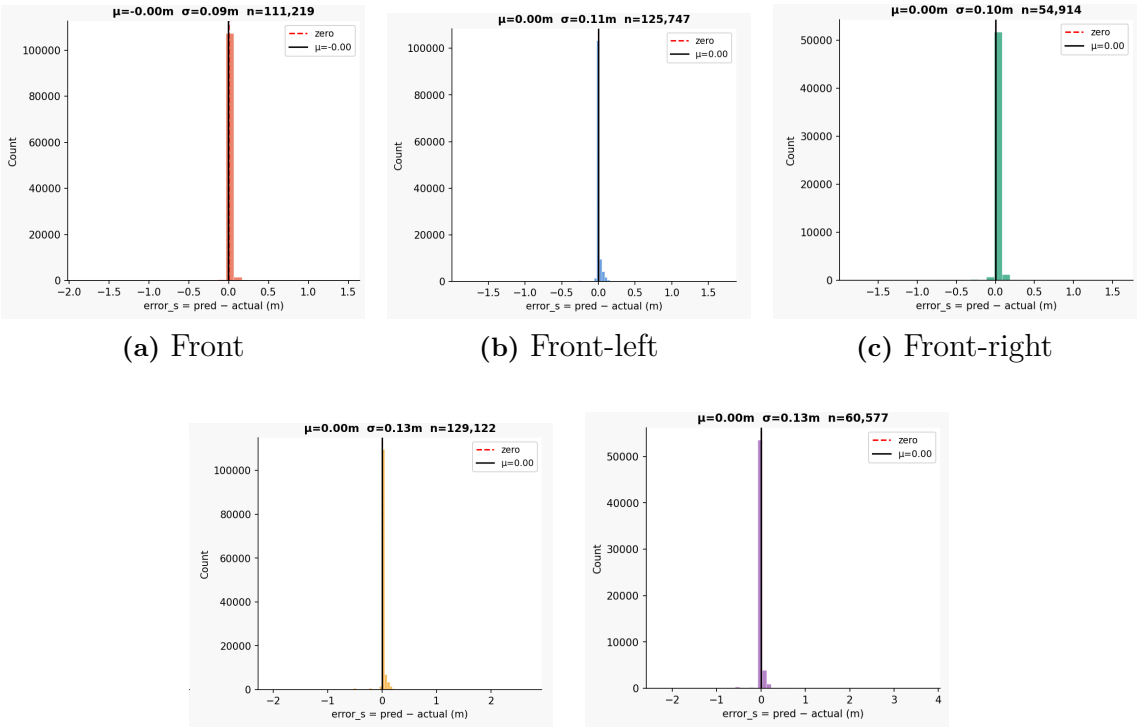
4.3.1 Evaluation Methodology

To measure the prediction errors, at each simulation step k, the predicted state $(s_{i,H}, v_{i,H}, l_{i,H})$ of the five nearest surrounding vehicles (front, front-left, front-right, rear-left, rear-right) was recorded. Then, H steps later, the actual state of each

vehicle was queried from SUMO and compared to the prediction. This procedure was repeated across 100 seeds using the pure MPC controller with $H = 1, \cdots, 5$ steps. Each of the two methods was evaluated identically by running separate evaluations. The primary metrics are the mean error $\bar{e}_{s,h} = \hat{s}_{i,H} - s_{i,H}$ (bias) and the standard deviation of the error $\sigma_{s,h}$ (spread) for longitudinal position. Velocity error $\bar{e}_{v,h}$ and its standard deviation are also reported for use in safety and collision costs.

4.3.2 Results and Method Comparison

The complete per-role, per-step error statistics for Methods 1 and 2 are provided in Tables A.1-A.10 in Appendix A.1.
Method 1 (reactive multi-agent projection) achieves low bias by modeling car-following reactions, but introduces different error sources: 1) Model error due to the mismatches between the discrete prediction model and SUMO's internal state update model. 2) The lack of lane change policy in the prediction model and its effect on the following vehicles on the target lane. The standard deviation of Method 1 is larger than Method 2 at all horizon steps.



(a) Front **(b)** Front-left **(c)** Front-right

(d) Back-left **(e)** Back-right

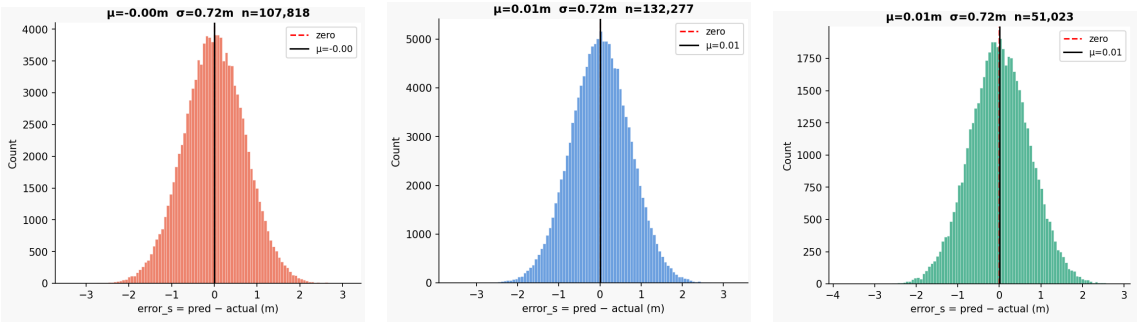
Figure 4.9: Position prediction error distributions per surrounding vehicle role at $h = 5$ steps using Method 2 prediction model.

Method 2 (constant acceleration) achieves the lowest mean absolute error and standard deviation at all horizon steps $h \leq 5$. For all $H = 1, \cdots, 5$, the mean position error is near zero because the current acceleration is the dominant predictor of position for a few steps ahead. At $h = 5$ (1 second), the standard deviation remains

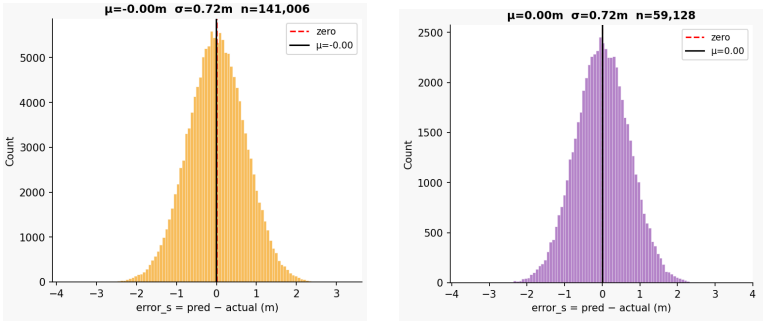
below 0.150 m. Method 2 is therefore adopted as the prediction model for all controller evaluations.
Figure 4.9 shows the position error distributions at horizon $h = 5$ (1.0 s ahead) for all five surrounding vehicle roles using Method 2.

4.3.3 Noise Model Parameter Selection

Since the mean and standard deviation of the Method 2 prediction error at each horizon step h remain relatively small, the dominant uncertainty in the stochastic model originates primarily from the measurement noise model discussed in Section 3.6. By combining the prediction uncertainty with the measurement noise parameters listed in Table A.16, the final error statistics for both position and velocity states were obtained for different prediction horizons. The corresponding values are summarized in Appendix Tables A.11–A.15.
These values represent the final measurement uncertainty, and are used throughout to compute the tightened safe distances and robust collision thresholds. The error distributions for the $H = 5$ case are illustrated through histograms for each surrounding vehicle role in Figure 4.10.
It should be noted that the values given to the rollout/RL functions for surrounding vehicles in stochastic implementations are the trajectories based on the noisy measurements.



(a) Front **(b)** Front-left **(c)** Front-right



(d) Back-left **(e)** Back-right

Figure 4.10: Position final prediction error distributions per surrounding vehicle role at $h = 5$ steps.

4.4 Controller Comparison

This section presents the evaluation of all controllers on 100 independently seeds. Controllers are compared in three respects:

1. Safety (collision rate)
2. Driving quality (mean reward, cost per step and its decomposition)
3. Computational cost (mean time per step)

The evaluation starts from the simplest baseline to the most complex variant. All evaluations use the same set of seeds that are completely separate from the training and holdout seeds used during RL training. The the risk tolerance ϵ for both collision and safe distance for all stochastic models is 1%. It should be noted that the reported computation times are influenced by the workload of the shared computational cluster at the time of evaluation. Since different controller configurations were evaluated in separate simulation runs, some timing variations may originate from fluctuations in available computational resources and concurrent workloads rather than from the controller algorithms themselves.

4.4.1 Pure MPC

The pure MPC controller is evaluated across five prediction horizons $H \in \{1, 2, 3, 4, 5\}$ on 100 independently seeds with terminal cost $q(x_H) = 0$. Results are reported in Tables 4.7 - 4.9.

Table 4.7: Pure MPC collision results on 100 seeds ($v_{ref} = 25\,m/s$).

H	Collisions %	Mean Speed
1	88%	25.0
2	0%	22.14
3	0%	22.6
4	1%	22.6
5	0%	22.7

Table 4.8: Pure MPC mean cost per step on 100 seeds. $C_{col,b}$ is 0 for all cases.

H	Total	C_{speed}	C_{accel}	$C_{safe,f}$	$C_{safe,b}$	$C_{col,f}$	C_{lc}	C_{cd}	C_{left}
1	241,871	0.0	0.0	131,041	844	109,984	0.0271	0.0040	0.9958
2	1,084	8.2053	0.1136	866.59	208.20	0	0.0126	0.0022	0.9808
3	647	5.7638	0.1599	444.50	195.68	0	0.0133	0.0016	0.9953
4	1,590	5.7555	0.1790	677.98	238.30	666.80	0.0130	0.0022	0.9649
5	448	5.3117	0.1756	236.46	205.16	0	0.0133	0.0022	1.0086

Table 4.9: Pure MPC mean computation time per step ($\Delta t = 0.2\,\mathrm{s}$).

H	Time ($ms/step$)
1	2.51
2	4.34
3	6.28
4	11.37
5	17.49

4.4.1.1 Safety

At $H = 1$ (0.2 seconds of lookahead), the controller collides in 81 out of 100 episodes. With only a single optimization step, the DP selects the acceleration that minimizes the immediate stage cost at $h = 0$ without any awareness of the state at $h = 1$ and beyond. When the ego truck approaches a slow leader, the DP does not anticipate that the gap will continue to close over the next several steps. The speed tracking cost encourages maintaining v_{ref}, and even though the front safety cost is high, changing the speed will not have enough effect on decreasing it. Therefore, the only option is to either keep getting closer to the leader, which will lead to collision, or change the lane, which the target lane may have its own slow leader. This is confirmed by the cost decomposition in Table 4.8, $C_{speed} = 0$, while the front safety cost $c_{front} = 115,506$ and front collision cost $C_{col,f} = 117,011$ together dominate the total cost.
From $H = 2$ onward, the collision rate drops to zero across all 100 seeds, except for $H = 4$ which has one collision among 100 episodes. A two-step lookahead is sufficient for the DP to evaluate the immediate consequence of accelerating or a lane change, which makes the controller aware of the gap from the leader in the current or target lane one step ahead and prevents unsafe behavior. This result shows that a single additional step of lookahead provides a massive improvement in the safety.

4.4.1.2 Driving Quality

While most of $H \geq 2$ cases achieve zero collisions, driving quality mostly improves substantially with increasing horizon. The total cost per step falls from $241,871$ at $H = 1$ to $1,084$ at $H = 2$ and 448 at $H = 5$, a reduction of 59% from $H = 2$ to $H = 5$. The dominant cost term throughout $H = 2$ to $H = 5$ is the front safety cost, which drops from 866.59 at $H = 2$ to 236.46 at $H = 5$. The reason behind this drop is that at $H = 2$, the controller sees only 0.4 seconds ahead and frequently finds itself inside the safe-distance threshold of a slow leader before it can react, accumulating large quadratic safety penalties. At $H = 5$ the controller anticipates these situations a full second in advance, which can either decelerate earlier or execute a lane change before the gap becomes critical. The only abnormal behavior is for $H = 4$ where it has higher safety costs compared to $H = 3$, which leads to one collision. It indicates that this trend of convergence is not linear, and there might be some local optimal points.
The mean speed is $22.1 - 22.7\,m/s$ for all $H \geq 2$, always below $v_{ref} = 25$, reflecting

the mixed traffic conditions where the ego truck frequently follows slower vehicles. The speed cost per step is small relative to the safety cost, confirming that speed deviation is not the primary limitation, and the controller prioritizes safety over speed tracking, as intended by the cost weights.

The rear safety cost is present at all horizons but no rear collision is recorded, confirming that the physics-based rear collision condition described in Section 3.4.2.5 successfully prevents unsafe merges. Lane change and cooldown costs are negligible across all horizons.

4.4.1.3 Computational Cost

Table 4.9 shows the mean computation time per step. At $H = 1$ the controller requires $2.51\,ms$ per step, where at $H = 5$, this increases to $17.49\,ms$, a 6.96 times increase. Both remain well within the $200\,ms$ control time step $\Delta t = 0.2$, using at most 8.7% of the available computation time. The nonlinear growth with H is expected due to the fact that the reachable state space expands with H, where both the number of grid points evaluated and the number of backward DP passes increased.

4.4.2 Rollout ($H_{roll} = 10$)

This section evaluates the rollout-based controller with rollout horizon $H_{roll} = 10$ across prediction horizons $H \in \{1, 2, 3, 4, 5\}$, a hybrid method with transparent policy. Results are reported in Tables 4.10 - 4.12.

Table 4.10: Rollout controller general results on 100 seeds with $H_{roll} = 10$.

H	Collisions %	Mean Speed	Total cost per step
1	34%	22.48	53,562
2	1%	22.55	2,078
3	2%	22.61	4,079
4	2%	22.60	5,203
5	3%	22.57	5,072

Table 4.11: Rollout mean cost per step on 100 seeds with $H_{roll} = 10$.

H	C_{speed}	C_{accel}	$C_{safe,f}$	$C_{safe,b}$	$C_{col,f}$	$C_{col,b}$	C_{lc}	C_{cd}	C_{left}
1	6.36	0.15	1,418	1,082	25,526	25,526	0.0231	0.0077	0.4921
2	6.02	0.21	1,317	86	667	0	0.0183	0.0042	0.3609
3	5.71	0.24	2,004	65	1,335	0	0.0192	0.0032	0.4081
4	5.75	0.24	3,763	88	1,345	667	0.0184	0.0026	0.4582
5	5.91	0.25	2,984	49	2,030	0	0.0194	0.0028	0.4562

Table 4.12: Rollout mean computation time per step with $H_{roll} = 10$ ($\Delta t = 0.2\,\mathrm{s}$).

H	Time ($ms/step$)
1	7.78
2	14.42
3	21.75
4	37.43
5	75.14

4.4.2.1 Safety

The rollout controller with $H_{roll} = 10$ substantially improves the short-horizon behavior of pure MPC at $H = 1$. While pure MPC with $H = 1$ produced collisions in 88% of episodes, the rollout extension reduces this to 34%. This confirms that the rollout estimate successfully introduces long-term awareness beyond the immediate DP horizon. By simulating the future trajectory using the base policy, the controller becomes aware that maintaining aggressive acceleration may lead to unsafe states several steps later, even when the immediate stage cost appears favorable.
However, unlike the pure MPC controller at $H \geq 2$, rollout does not completely eliminate collisions. The collision rate remains between $1\% - 3\%$ for $H = 2$ to $H = 5$. This behavior reflects a fundamental limitation of rollout methods, which is the quality of the terminal estimate is bounded by the quality of the base policy itself. Since the rollout uses the SUMO Krauss model as the future policy, discussed in Section 3.5, inaccuracies or myopic behaviors in this heuristic propagate into the terminal cost approximation. Consequently, the DP may still select trajectories that appear safe under the rollout policy but later become unsafe in the actual closed-loop interaction, even by considering the fact that discount factor decrease the effect of further steps in the future.
The collision cost decomposition in Table 4.11 confirms this observation. Even for horizons with low collision percentages, the front collision cost remains nonzero, indicating occasional failures in long-term safety prediction. Nevertheless, compared to pure MPC at $H = 1$, rollout dramatically improves safety with only a modest increase in computational complexity.

4.4.2.2 Driving Quality

The $H_{roll} = 10$ rollout controller achieves mean speeds between 22.48 and 22.61 m/s, comparable to pure MPC for $H \geq 2$. The speed cost remains relatively low across all horizons, showing that the controller still tracks the reference velocity reasonably well while accounting for future safety consequences.
However, unlike pure MPC, the total cost does not monotonically decrease with increasing H. The lowest total cost occurs at $H = 2$ $(2,078)$, after which the cost increases substantially for larger horizons. This increase is primarily driven by the front safety cost and collision cost terms. For example, the front safety cost grows from $1,317$ at $H = 2$ to over $3,763$ at $H = 4$. Similarly, the front collision cost increases from 667 to over $2,000$ by $H = 5$.

This behavior suggests that the rollout approximation introduces inconsistencies into the DP optimization. As the exact DP horizon becomes larger, the states of surrounding vehicles get further away from what will actually happen, and the abnormal behavior appears. If the rollout policy poorly represents the true optimal long-term behavior, the resulting terminal estimate can bias the optimization toward trajectories that appear favorable during rollout but lead to unsafe or inefficient outcomes later. In other words, increasing H does not necessarily improve performance when the terminal approximation itself is imperfect.

But the back safety cost and back collision cost decrease as H increases, indicating more conservative lane change actions.

Compared to pure MPC, rollout also produces significantly lower left-lane occupancy costs ($C_{left} \approx 0.36 - 0.49$), indicating that the controller spends less time in overtaking lanes. This likely results from the rollout policy preferring conservative car-following behavior rather than aggressive lane changes. Lane-change and cooldown costs remain negligible across all horizons.

4.4.2.3 Computational Cost

Table 4.12 reports the mean computation time per step. The rollout controller is consistently more expensive than pure MPC because each terminal state requires an additional forward simulation of the rollout trajectory. Computation time increases from $7.78\,ms$ at $H = 1$ to $75.14\,ms$ at $H = 5$.

Despite this increase, all configurations remain below the available control interval of $\Delta t = 0.2\,s$. Even the most expensive configuration uses approximately 37.6% of the available computation budget. The growth in runtime is caused by the larger DP state space at higher H and the increasing number of rollout evaluations required at terminal nodes.

4.4.3 Rollout ($H_{roll} = 20$)

By increasing the rollout horizon to $H_{roll} = 20$, the rollout trajectory is extended twice as far into the future, providing a longer terminal estimate for the DP optimization. Results are reported in Tables 4.13 - 4.15.

Table 4.13: Rollout controller general results on 100 seeds with $H_{roll} = 20$.

H	Collisions %	Mean Speed	Total cost per step
1	32%	22.76	49,333
2	13%	22.60	12,735
3	10%	22.69	8,946
4	15%	22.64	12,363
5	7%	22.76	8,942

Table 4.14: Rollout mean cost per step on 100 seeds with $H_{roll} = 20$.

H	C_{speed}	C_{accel}	$C_{safe,f}$	$C_{safe,b}$	$C_{col,f}$	$C_{col,b}$	C_{lc}	C_{cd}	C_{left}
1	5.03	0.16	2,091	1,345	27,224	18,668	0.0299	0.0159	0.5215
2	5.76	0.20	1,498	570	9,239	1,421	0.0220	0.0075	0.4544
3	5.34	0.21	790	519	6,938	694	0.0215	0.0052	0.4247
4	5.55	0.22	1,158	542	10,656	0	0.0221	0.0045	0.4358
5	5.00	0.22	3,124	317	4,809	687	0.0212	0.0031	0.4869

4.4.3.1 Safety

Increasing the rollout horizon from $H_{roll} = 10$ to $H_{roll} = 20$ does not improve safety performance. While the collision rate at $H = 1$ decreases slightly from 34% to 32%, the performance for larger DP horizons becomes significantly worse. For example, at $H = 2$ the collision rate increases from 1% to 13%, and at $H = 4$ it reaches 15%. Even the best-performing case, $H = 5$, still produces collisions in 7% of episodes.

This behavior indicates that extending the rollout horizon amplifies the inaccuracies of the terminal approximation instead of improving it. The rollout controller assumes that the future behavior generated by the base policy provides a reasonable estimate of long-term cost-to-go. However, over long horizons, the accumulated prediction error becomes large because surrounding traffic evolves differently from the rollout assumptions. Consequently, the DP optimization may favor trajectories that appear safe or efficient under the rollout simulation but become unsafe during actual closed-loop execution.

The cost decomposition confirms this observation. Safety related costs remain extremely high for all horizons, both the front safety cost and front collision cost. For example, $C_{safe,f}$ remains above 4,800 even at $H = 5$, and exceeds 10,000 at $H = 4$. Unlike the $H_{roll} = 10$ case, rear collision costs also become significant, especially at $H = 1$ and $H = 2$, indicating that the longer rollout occasionally encourages lane changes that appear feasible during rollout evaluation but later create unsafe rear-vehicle interactions.

These results demonstrate that a longer rollout horizon does not necessarily improve policy quality when the rollout policy itself is imperfect, even by accounting the effect of discount factor. Instead, the terminal estimate becomes increasingly biased by the inaccuracies of the heuristic future simulation.

4.4.3.2 Driving Quality

The rollout controller with $H_{roll} = 20$ maintains mean speeds between 22.60 and 22.76 m/s, comparable to both pure MPC and the $H_{roll} = 10$ rollout controller. The speed-tracking cost remains small across all horizons, showing that the controller still prioritizes maintaining traffic flow close to the reference speed.

However, the total cost per step is substantially higher than both pure MPC and rollout with $H_{roll} = 10$. Even the best case, $H = 5$, produces a total cost of approximately 8,942 per step, which is nearly 20 times larger than pure MPC at $H = 5$ (448). The dominant contributors remain the collision and safety costs.

Unlike the shorter rollout horizon, the front safety cost does not follow a consistent trend with increasing H. It initially decreases from $2,091$ at $H = 1$ to 790 at $H = 3$, but then rises sharply again to over $3,123$ at $H = 5$. This instability indicates that the optimization becomes increasingly sensitive to rollout inaccuracies as both the DP horizon and rollout horizon grow.
The rear safety costs are also noticeably larger than for the $H_{roll} = 10$ configuration, especially for smaller horizons. This suggests that the longer rollout occasionally encourages more aggressive lane changes or merges that later place rear vehicles in unsafe conditions. Similarly, rear collision costs appear in multiple configurations, unlike the shorter rollout case where rear collisions were mostly eliminated.
The left-lane occupancy cost remains relatively low $(0.42 - 0.52)$, consistent with the previous rollout configuration and lower than pure MPC. This again suggests that the rollout policy prefers conservative car-following behavior rather than frequent overtaking maneuvers. Lane-change and cooldown costs remain negligible.
Overall, extending the rollout horizon to 20 steps does not improve driving quality. Instead, it introduces larger safety violations and substantially higher collision penalties, showing that longer rollout simulations are only beneficial if the rollout policy itself accurately predicts long-term traffic evolution.

4.4.3.3 Computational Cost

Table 4.15 reports the mean computation time per step. Increasing the rollout horizon from 10 to 20 approximately doubles the rollout simulation workload at every terminal state, resulting in a substantial increase in runtime.
The computation time increases from $11.37\,ms$ at $H = 1$ to $122.00\,ms$ at $H = 5$. Compared to the corresponding $H_{roll} = 10$ configuration, the runtime increase ranges from approximately 46% at $H = 1$ to over 62% at $H = 5$.
Although all configurations still remain below the available control interval of $\Delta t = 0.2\,s$, the highest-cost configuration already consumes approximately 61% of the available computation budget. This shows that longer rollout horizons rapidly become computationally expensive due to the repeated forward simulations required for every terminal node in the DP tree.

Table 4.15: Rollout mean computation time per step with $H_{roll} = 20$ ($\Delta t = 0.2\,\mathrm{s}$).

H	**Time** $(ms/step)$
1	11.37
2	23.14
3	33.10
4	59.75
5	122.00

4.4.4 Pure RL

The pure RL controller uses the $16.6M$ checkpoint from Experiment 5 ($\gamma = 0.98$, $w_{speed} = 1$, $d_{sensor} = 150$), operating without any MPC optimization, where the DQN

policy selects actions directly from the observation vector at each step. Results are reported in Tables 4.16 and 4.17.

Table 4.16: Pure RL evaluation results on 100 seeds ($v_{ref} = 25\,\mathrm{m/s}$).

Coll. %	**Mean Reward**	**Mean Speed** (m/s)	**Time (ms/step)**
0%	−10.71	20.75	0.32

Table 4.17: Pure RL mean cost per step on 100 seeds. $C_{col,b}$ is 0 for all cases.

Total	C_{speed}	C_{accel}	$C_{safe,f}$	$C_{safe,b}$	$C_{col,f}$	C_{lc}	C_{cd}	C_{left}
551	18.02	2.38	517	11.57	0	0.017	0.023	2.171

4.4.4.1 Safety

The pure RL controller achieves zero collisions across all 100 seeds, matching the pure MPC configurations ($H \geq 2$). This confirms that the trained policy has successfully learned to avoid collisions in diverse traffic scenarios.

4.4.4.2 Driving Quality

The pure RL controller's total cost of 551 is slightly better than pure MPC $H = 3$ (647), but according to the cost decomposition in Table 4.17, it has a different driving strategy. The dominant cost term is the front safety cost ($C_{safe,f} = 517$ per step), which is comparable to pure MPC $H = 3$ ($C_{front,f} = 444.5$), but the RL agent maintain a speed of well below v_{ref}. The mean speed of 20.75 m/s is the lowest among all controllers, approximately $4.25 m/s$ below the reference velocity, and the speed cost per step ($C_{speed} = 18.02$) is $3-4$ time higher than for pure MPC at $H \geq 2$. The acceleration cost ($C_{accel} = 2.38$) is also notably higher than for any MPC cases, reflecting the less smooth longitudinal profiles.
The left-lane occupancy cost ($C_{left} = 2.17$) is higher than for most MPC variants, indicating more tendency to linger in left lanes. However, the back safety cost ($C_{safe,b} = 11.57$) is noticeably lower than MPC variants, indicating that more cautious lane changing than other controllers. The lane change and cooldown costs ($C_{lc} = 0.017$, $C_{cd} = 0.018$) are comparable to MPC, which indicates similar lateral activity frequency.

4.4.4.3 Computational Cost

The pure RL controller requires only 0.32 ms per step, approximately 54 times faster than pure MPC $H = 5$ (17.49 ms) and 8 times faster than pure MPC $H = 1$ (2.51 ms). This reflects the fact that the DQN forward pass through a two-layer [256, 256] network is cheaper than the backward DP grid search. The pure RL controller uses 0.16% of the available 200 ms simulation step, making it by far the most computationally efficient option.

4.4.5 Hybrid MPC - RL

The main goal of this thesis is to introduce and evaluate a hybrid MPC - RL controller. For a deterministic environment, Tables 4.18 - 4.20 report the results across prediction horizons $H \in \{1, 2, 3, 4, 5\}$. In this configuration, the DQN network is used as the terminal value function approximation inside the DP optimization, replacing the handcrafted rollout policy.

Table 4.18: Hybrid MPC - RL evaluation results on 100 seeds.

H	Collisions %	Mean Speed
1	0%	21.08
2	0%	21.09
3	1%	21.18
4	0%	21.15
5	2%	21.14

4.4.5.1 Safety

The hybrid MPC - RL controller achieves excellent safety performance for small and moderate horizons. The controller produces zero collisions for $H = 1$, $H = 2$, and $H = 4$, while only one collision occurs at $H = 3$ and two collisions at $H = 5$. Compared to pure MPC, the hybrid approach dramatically improves the short-horizon case. In particular, pure MPC at $H = 1$ produced collisions in 88% of episodes, whereas the hybrid controller eliminates all collisions at the same horizon.

This result confirms that the RL value function successfully provides long-term information beyond the finite DP horizon. Unlike the rollout-based controller, which depends on explicit future simulation using the heuristic policy, the learned value function directly estimates long-term future cost from the current state. Consequently, even when the exact DP horizon is very short, the controller can still recognize that aggressive acceleration or unsafe lane changes will lead to large future penalties.

However, the results also show that increasing the DP horizon does not always improve safety. Small collision rates reappear at $H = 3$ and $H = 5$, with nonzero front collision costs. This indicates that the learned value approximation is imperfect and may occasionally bias the optimization toward trajectories that appear favorable according to the neural network estimate but later become unsafe during closed-loop interaction. Nevertheless, the overall collision rates remain substantially lower than rollout-based methods.

The rear safety costs remain relatively small across all horizons and no rear collisions occur. This confirms that the collision filtering logic described in Section 3.4.2.5 continues to successfully reject unsafe lane-change maneuvers even when the RL value function is used as the terminal approximation.

4.4.5.2 Driving Quality

The hybrid MPC - RL controller achieves the lowest total cost among all controllers for several horizons. The best result occurs at $H = 2$, where the total cost per step decreases to only 257, outperforming both pure MPC $H = 5$ (448) and pure RL (551). Similarly, the $H = 4$ configuration achieves a total cost of approximately 300, also substantially lower than the corresponding pure MPC and pure RL cases. The dominant cost component remains the front safety cost, but its magnitude is significantly reduced compared to both pure MPC and rollout methods. For example, at $H = 4$, the front safety cost is only 151, compared to 236 for pure MPC $H = 5$, 517 for pure RL, and several thousand for rollout configurations. This indicates that the learned value function successfully guides the DP optimization toward safer long-term spacing behavior.

However, the hybrid controller adopts a noticeably more conservative driving strategy than pure MPC. The mean speed remains approximately $21.1\,m/s$ across all horizons, consistently below the $22.1 - 22.7\,m/s$ achieved by pure MPC. Consequently, the speed-tracking cost remains relatively high ($C_{speed} \approx 15$), approximately three times larger than pure MPC $H = 5$. This behavior closely resembles the pure RL controller, indicating that the learned value function biases the optimization toward conservative car-following policies that prioritize safety over aggressive speed tracking.

The acceleration cost is also higher than pure MPC across all horizons, though still relatively small in absolute value. The left-lane occupancy cost remains significantly larger than for pure MPC or rollout methods, indicating that the hybrid controller spends more time in overtaking lanes. This again reflects the behavioral tendencies learned by the RL policy embedded inside the value function approximation.

Unlike rollout, the hybrid controller does not suffer from catastrophic growth in safety and collision costs as the horizon increases. Although the $H = 5$ case shows some degradation, the overall performance remains substantially more stable than rollout-based approaches. This suggests that the learned neural-network value function provides a more consistent terminal approximation than long heuristic rollout simulations.

Table 4.19: Hybrid MPC - RL mean cost per step on 100 seeds.

H	Total	C_{speed}	C_{accel}	$C_{safe,f}$	$C_{safe,b}$	$C_{col,f}$	C_{lc}	C_{cd}	C_{left}
1	883	15.39	1.43	774.85	88.57	0	0.0241	0.0168	3.1338
2	257	15.30	1.27	194.81	42.30	0	0.0169	0.0092	3.6070
3	1,015	14.62	1.26	245.19	79.42	670.72	0.0182	0.0080	3.5297
4	300	14.85	1.28	151.20	128.75	0	0.0161	0.0068	3.4599
5	2,178	14.86	1.24	668.01	152.89	1,337.93	0.0227	0.0138	3.3221

4.4.5.3 Computational Cost

Table 4.20 reports the mean computation time per step. The hybrid MPC+RL controller is significantly more computationally expensive than both pure MPC and rollout-based controllers. The mean computation time increases from $17.12, ms$ at

$H = 1$ to $242.26, ms$ at $H = 5$. Unlike pure MPC and rollout, the $H = 5$ configuration exceeds the available control interval $\Delta t = 0.2, s$, requiring approximately 121% of the real-time computation budget.

The computational growth is caused by two factors. First, the DP state space still expands with increasing prediction horizon H, similarly to pure MPC and rollout. Second, unlike rollout methods that use relatively lightweight forward simulations, the hybrid controller evaluates a deep neural network terminal value function for every reachable terminal state. The terminal estimator uses a fully connected $[256, , 256]$ DQN architecture, which is computationally expensive compared to the rollout policies with $H_{roll} = 10$ or 20, where the terminal estimate is obtained through direct simulation.

This difference becomes particularly important at larger horizons. For example, rollout with $H_{roll} = 10$ requires $75.14, ms$ at $H = 5$, while the hybrid MPC+RL controller requires $242.26, ms$ for the same horizon, more than three times larger. Although rollout additionally simulates multiple future steps, the rollout policy itself is a simple heuristic traffic model with relatively cheap state updates. In contrast, the neural network-based terminal evaluation requires repeated forward passes through a large nonlinear function approximator for every terminal node of the DP tree.

Table 4.20: Hybrid MPC - RL mean computation time per step ($\Delta t = 0.2\,\mathrm{s}$).

H	**Time** $(ms/step)$
1	17.12
2	49.77
3	113.74
4	171.50
5	242.26

Nevertheless, for moderate horizons such as $H \leq 4$, the hybrid controller still remains within the real-time budget. The results therefore illustrate the tradeoff between terminal value quality and computational complexity, where learned value functions provide substantially better long-term guidance than heuristic rollout policies, but at a significantly higher computational cost.

4.4.6 Pure MPC under Stochastic Prediction and Measurement Noise

This section evaluates the pure MPC controller under the stochastic extension described in Section 3.6. In this setting, both the surrounding vehicle measurement uncertainty and the predicted trajectory uncertainty described in Section 4.3 are considered. The controller uses noisy observations and tightened safety constraints derived from the uncertainty bounds. Results are reported in Tables 4.21 - 4.23.

Table 4.21: Stochastic pure MPC evaluation results on 100 seeds.

H	Collisions %	Mean Speed
1	86%	25.0
2	0%	22.17
3	0%	22.59
4	0%	22.61
5	0%	22.62

4.4.6.1 Safety

The stochastic MPC results remain qualitatively similar to the deterministic case. At $H = 1$, the controller still performs poorly, producing collisions in 86% of the episodes. Although the new robust safety constraints increase the safety margins, a single-step horizon remains fundamentally too short to anticipate the future consequences of maintaining high speed behind slower traffic. The controller still minimizes only the immediate stage cost, and therefore cannot react early enough to avoid collisions.
From $H = 2$ onward, the controller again achieves near-perfect safety performance, with zero collisions across all seeds for $H = 2$ to $H = 5$. This confirms that the stochastic extension preserves the main safety advantage of MPC once sufficient lookahead is available. The additional safety margins introduced by the uncertainty model successfully compensate for the noisy measurements and prediction uncertainty without destabilizing the optimization.
Compared to the deterministic implementation, the stochastic controller generally produces slightly higher safety costs, especially in the front safety term. This is expected because the tightened safety constraints effectively enlarge the required separation distances. As a result, the controller behaves more conservatively even when no actual collision risk exists.

Table 4.22: Stochastic pure MPC mean cost per step on 100 seeds. $C_{col,b}$ is 0 for all cases.

H	Total	C_{speed}	C_{accel}	$C_{safe,f}$	$C_{safe,b}$	$C_{col,f}$	C_{lc}	C_{cd}	C_{left}
1	234,519	0	0	129,913	913	103,692	0.0286	0.0067	0.9703
2	1,769	8.00	0.36	1,547	212	0	0.0149	0.0038	1.1031
3	830	5.82	0.51	713	110	0	0.0147	0.0030	1.0109
4	1,259	5.69	0.58	1,015	237	0	0.0136	0.0030	0.9717
5	754	5.68	0.58	587	160	0	0.0133	0.0030	0.9629

4.4.6.2 Driving Quality

The driving behavior under uncertainty remains close to the deterministic MPC results. The mean speed for $H \geq 2$ remains approximately $22.2 - 22.6\,m/s$, showing that the stochastic extension does not significantly reduce traffic flow efficiency despite the more conservative safety margins.

However, several cost components increase under uncertainty. The front safety cost increases substantially compared to the deterministic case, particularly at shorter horizons. For example, at $H = 2$, the front safety cost rises from approximately 867 in the deterministic case to $1,547$ in the stochastic case. This increase directly reflects the uncertainty-aware safety margins introduced into the collision and safe-distance calculations. The higher front safety cost in the stochastic setting does not necessarily indicate less safe behavior. In the stochastic formulation, uncertainty margins are incorporated into the safe-distance computation to account for measurement and prediction uncertainty. Consequently, the effective safety threshold becomes more conservative than in the deterministic case. The controller therefore accumulates larger safety penalties even while maintaining safer physical behavior and avoiding collisions. In other words, the increase in $C_{safe,f}$ primarily reflects tighter robustness requirements rather than more aggressive driving.

The acceleration cost increases slightly across all horizons, indicating less smoother and more cautious longitudinal control actions. The controller tends to brake earlier and avoid aggressive acceleration because uncertainty in the future trajectories reduces confidence in close-gap maneuvers.

Despite these increases, the overall qualitative trend with horizon length remains unchanged. The best performance is again obtained at larger horizons, with the total cost decreasing from $1,769$ at $H = 2$ to 754 at $H = 5$. The front safety cost remains the dominant component throughout all stochastic MPC configurations.

The lane change, cooldown, and left-lane occupancy costs remain small and largely unchanged relative to the deterministic implementation, indicating that the stochastic extension primarily affects longitudinal safety behavior rather than lateral driving strategy.

4.4.6.3 Computational Cost

The computation time per step is unchanged from the deterministic MPC implementation because the stochastic extension modifies only the cost evaluation and safety thresholds, without changing the DP state-space structure itself. The controller therefore retains real-time feasibility for all horizons.

Table 4.23: Stochastic pure MPC mean computation time per step ($\Delta t = 0.2\,\mathrm{s}$).

H	**Time** $(ms/step)$
1	2.51
2	4.34
3	6.28
4	11.37
5	17.49

4.4.7 Rollout ($H_{roll} = 10$, Stochastic Prediction and Measurement Noise)

Similar to the deterministic case, the rollout-based controller is evaluated under stochastic prediction and measurement uncertainty using rollout horizon $H_{roll} = 10$. Results are reported in Tables 4.24 - 4.26.

Table 4.24: Stochastic rollout controller general results on 100 seeds with $H_{roll} = 10$.

H	Collisions %	Mean Speed	Total cost per step
1	39%	22.64	65,159
2	0%	22.52	1,072
3	1%	22.58	3,261
4	2%	22.58	3,978
5	1%	22.62	2,499

4.4.7.1 Safety

Under stochastic uncertainty, the rollout controller with $H_{\text{roll}} = 10$ maintains the same qualitative behavior observed in the deterministic case. At $H = 1$, rollout substantially improves the short-horizon safety of pure MPC by reducing the collision rate from 86% to 39%. This again confirms that the rollout estimate successfully injects long-term awareness beyond the one-step DP horizon.
Compared to deterministic rollout, the stochastic version shows mixed behavior depending on the horizon. At $H = 2$, the collision rate improves from 1% to 0%, while at larger horizons the collision rate remains low but nonzero ($1\% - 2\%$). This indicates that uncertainty-aware safety margins improve robustness in some scenarios, but cannot completely eliminate rollout approximation errors.
Unlike deterministic MPC, rollout relies on a terminal approximation generated by the base policy. Under stochastic uncertainty, this approximation becomes even more sensitive to prediction mismatch because the rollout trajectory itself is generated from noisy vehicle states. Consequently, the DP optimization may still evaluate some trajectories as safe even though the actual closed-loop interaction later produces unsafe situations.
The collision cost decomposition in Table 4.25 confirms this behavior. Although collision frequency is low for $H \geq 2$, nonzero front and rear collision costs remain present at several horizons. Compared to deterministic rollout, the stochastic implementation generally exhibits higher front and rear safety costs due to the uncertainty-aware tightening of safe-distance constraints discussed in Section 3.6. Therefore, the larger safety penalties primarily reflect more conservative robustness margins rather than more aggressive driving behavior.

4.4.7.2 Driving Quality

The stochastic rollout controller maintains mean speeds between 22.52 and 22.64 m/s, very similar to both deterministic rollout and deterministic MPC. The speed track-

ing cost remains relatively small across all horizons, indicating that uncertainty handling does not significantly reduce average cruising speed.

Table 4.25: Stochastic rollout mean cost per step on 100 seeds with $H_{roll} = 10$.

H	C_{speed}	C_{accel}	$C_{safe,f}$	$C_{safe,b}$	$C_{col,f}$	$C_{col,b}$	C_{lc}	C_{cd}	C_{left}
1	5.56	0.60	2,054	1,552	29,963	31,583	0.0364	0.0284	0.5256
2	6.17	0.65	973	92	0	0	0.0198	0.0052	0.4532
3	5.86	0.62	1,833	82	669	669	0.0208	0.0048	0.4511
4	5.85	0.61	1,182	104	1,343	1,343	0.0207	0.0070	0.4553
5	5.68	0.60	1,052	102	669	669	0.0216	0.0070	0.4478

At $H = 2$, the stochastic rollout controller achieves its best overall performance with zero collisions and total cost per step of 1,072, which is approximately half of the deterministic rollout cost at the same horizon (2,078). However, similar to the deterministic rollout case, increasing H beyond 2 does not consistently improve performance. The total cost increases again for $H = 3$ and $H = 4$, mainly due to larger front safety and collision costs.
This non-monotonic behavior again reflects the sensitivity of rollout methods to the quality of the terminal approximation. As the exact DP horizon becomes larger, optimization decisions increasingly depend on the rollout estimate generated by the heuristic future policy. Since the rollout policy is based on simplified traffic behavior, now with noisy measurements, errors in the predicted long-term interactions can bias the optimization toward trajectories that later become inefficient or unsafe.
Compared to deterministic rollout, the stochastic implementation generally produces larger acceleration and safety costs because the controller reacts more cautiously to uncertainty. However, left-lane occupancy costs remain relatively low ($C_{left} \approx 0.45$), indicating that the controller still prefers conservative lane-following behavior over aggressive overtaking maneuvers. Lane-change and cooldown costs remain negligible across all horizons.

4.4.7.3 Computational Cost

Table 4.26 reports the mean computation time per step. Runtime increases from $9.10\,ms$ at $H = 1$ to $55.77\,ms$ at $H = 5$, remaining below the available control interval $\Delta t = 0.2\,s$ for all cases.
Compared to deterministic rollout, the stochastic implementation shows only a small increase in computation time. This slight runtime increase is likely caused by variations in cluster workload and shared computational resources during the separate simulation runs rather than by the stochastic formulation itself. This interpretation is supported by the fact that the stochastic pure MPC controller produced essentially identical runtimes to the deterministic MPC implementation.
The dominant computational cost still comes from the rollout simulations themselves rather than the stochastic safety calculations. However, the stochastic rollout controller remains significantly cheaper than the hybrid MPC - RL controller because the rollout policy uses lightweight analytical traffic simulation instead of repeated forward passes through the large DQN network. Nevertheless, runtime

grows rapidly with H because both the DP reachable state space and the number of rollout evaluations increase combinatorially with horizon length.

Table 4.26: Stochastic rollout mean computation time per step with $H_{roll} = 10$ ($\Delta t = 0.2\,\mathrm{s}$).

H	Time ($ms/step$)
1	9.10
2	15.05
3	25.59
4	41.51
5	55.77

4.4.8 Rollout ($H_{roll} = 20$, Stochastic Prediction and Measurement Noise)

By doubling the rollout horizon to $H_{roll} = 20$, the effect of a longer heuristic rollout prediction under stochastic prediction and measurement uncertainty is evaluated. Compared to the $H_{roll} = 10$ configuration, the terminal rollout simulation extends twice as far into the future, providing a longer approximate cost-to-go estimate. Results are reported in Tables 4.27 - 4.29.

Table 4.27: Stochastic rollout controller general results on 100 seeds with $H_{roll} = 20$.

H	Collisions %	Mean Speed	Total cost per step
1	43%	22.70	65,150
2	3%	22.54	5,851
3	4%	22.65	6,395
4	4%	22.62	5,129
5	2%	22.65	2,390

Table 4.28: Stochastic rollout mean cost per step on 100 seeds with $H_{roll} = 20$.

H	C_{speed}	C_{accel}	$C_{safe,f}$	$C_{safe,b}$	$C_{col,f}$	$C_{col,b}$	C_{lc}	C_{cd}	C_{left}
1	5.31	0.62	2,753	1,948	28,631	31,812	0.0442	0.0396	0.6592
2	6.05	0.63	1,219	574	2,025	2,025	0.0242	0.0124	0.4769
3	5.51	0.60	1,131	521	2,706	2,030	0.0234	0.0091	0.4495
4	5.65	0.60	1,091	648	2,030	1,354	0.0224	0.0085	0.4636
5	5.53	0.58	592	443	1,348	0	0.0239	0.0097	0.4646

Table 4.29: Stochastic rollout mean computation time per step with $H_{roll} = 20$ ($\Delta t = 0.2\,\mathrm{s}$).

H	Time ($ms/step$)
1	12.82
2	23.11
3	37.12
4	59.27
5	85.84

4.4.8.1 Safety

The stochastic rollout controller with $H_{roll} = 20$ shows the same overall trend observed in the deterministic long-rollout configuration. The extension in the rollout horizon does not improve safety performance and, in several cases, degrades it relative to $H_{roll} = 10$.
At $H = 1$, the collision rate increases to 43%, slightly worse than the stochastic rollout controller with $H_{roll} = 10$ (39%). Although the long rollout provides additional future information, the quality of the terminal approximation deteriorates over long horizons because the rollout trajectory increasingly diverges from the actual future traffic evolution, especially under noisy observations and uncertain predictions.
For $H \geq 2$, the collision rate decreases substantially compared to the deterministic $H_{roll} = 20$ implementation, where collision rates ranged from $7\% - 15\%$. Under stochastic uncertainty, the controller achieves collision rates between $2\% - 4\%$, indicating that the tightened safety constraints improve robustness against unsafe rollout predictions. Nevertheless, collisions are still not eliminated completely, unlike stochastic pure MPC at $H \geq 2$.
The collision cost decomposition in Table 4.28 further confirms this behavior. Both front and rear collision costs remain nonzero across most horizons, indicating that the rollout approximation occasionally predicts lane changes or accelerations that later become unsafe during closed-loop interaction. Rear collision costs are particularly more noticeable than in the $H_{roll} = 10$ case, suggesting that the long rollout horizon sometimes encourages overly optimistic merge decisions under uncertain traffic evolution.
Compared to stochastic rollout with $H_{roll} = 10$, the safety costs are generally larger at shorter horizons, especially at $H = 1$, where both front and rear safety penalties become very large. This reflects the compounded effect of uncertainty-aware safety margins and long-horizon rollout prediction mismatch.

4.4.8.2 Driving Quality

The stochastic rollout controller with $H_{roll} = 20$ maintains mean speeds between 22.54 and $22.70\,m/s$, remaining very similar to all previous MPC and rollout configurations. The speed-tracking cost also remains relatively small across all horizons, confirming that the controller continues to prioritize maintaining traffic flow close to the reference velocity.

However, similarly to the deterministic $H_{roll} = 20$ case, extending the rollout horizon does not improve overall driving quality. The total cost per step remains significantly larger than both stochastic pure MPC and stochastic rollout with $H_{roll} = 10$. Even the best-performing configuration, $H = 5$, still produces a total cost of approximately $2,390$ per step, substantially higher than stochastic pure MPC at the same horizon (754).
The dominant contributors remain the safety and collision costs. At shorter horizons, both front and rear collision costs are large, indicating frequent unsafe interactions generated by inaccurate long horizon rollout predictions. Although the total cost gradually decreases as H increases, the improvement is non-monotonic and still significantly worse than shorter rollout horizons.
Compared to deterministic rollout with $H_{roll} = 20$, the stochastic implementation generally produces lower collision rates and smaller collision costs for $H \geq 2$. This indicates that the uncertainty-aware safety tightening partially compensates for the rollout prediction inaccuracies by enforcing more conservative spacing behavior. However, this robustness improvement comes at the expense of increased rear safety costs and slightly more cautious longitudinal control.
The acceleration cost remains relatively small but consistently larger than deterministic pure MPC, reflecting more cautious throttle and braking behavior under uncertainty. Left-lane occupancy costs remain low ($C_{left} \approx 0.45 - 0.66$), again indicating that the rollout controller tends to favor conservative lane-following behavior rather than aggressive overtaking maneuvers. Lane-change and cooldown costs remain negligible across all horizons.
Overall, the results again demonstrate that increasing rollout horizon length is only beneficial when the rollout policy accurately predicts long-term traffic evolution. Under stochastic uncertainty, longer rollout simulations amplify prediction mismatch and accumulate larger approximation errors, limiting the effectiveness of the terminal estimate.

4.4.8.3 Computational Cost

Table 4.29 reports the mean computation time per step. Runtime increases from $12.82\,ms$ at $H = 1$ to $85.84\,ms$ at $H = 5$, remaining below the available control interval $\Delta t = 0.2\,s$ for all configurations.
Compared to the stochastic rollout controller with $H_{roll} = 10$, the longer rollout horizon substantially increases runtime because each terminal node requires approximately twice as many forward rollout simulation steps. However, compared to the deterministic $H_{roll} = 20$ implementation, the stochastic version exhibits slightly lower runtime at larger horizons. Since the stochastic extension itself does not significantly alter the DP state-space structure or rollout simulation complexity, these small timing differences are most likely caused by fluctuations in shared cluster workload and resource allocation during separate evaluation runs rather than by the controller formulation itself.
Despite the increased rollout horizon, the stochastic rollout controller remains substantially cheaper than the hybrid MPC - RL controller because the terminal approximation still relies on lightweight analytical traffic simulation instead of repeated neural-network evaluations. Nevertheless, the computational cost grows rapidly with

increasing H due to the combined effect of DP state-space expansion and repeated long-horizon rollout evaluations at terminal nodes.

4.4.9 Hybrid MPC - RL under Stochastic Prediction and Measurement Noise

This section evaluates the hybrid MPC - RL controller under stochastic prediction and measurement uncertainty. Similar to the stochastic MPC formulation, the controller operates using noisy observations and uncertainty-aware safety constraints. However, the terminal cost approximation is provided by the learned DQN value function instead of a handcrafted rollout policy. Results are reported in Tables 4.30 - 4.32.

4.4.9.1 Safety

The stochastic hybrid MPC - RL controller maintains excellent safety performance across nearly all horizons. Zero collisions are achieved for $H = 1$ through $H = 4$, while only a single collision occurs at $H = 5$. Compared to the deterministic hybrid implementation, the stochastic version slightly improves the safety performance at intermediate horizons by eliminating the isolated collision previously observed at $H = 3$.

Table 4.30: Stochastic hybrid MPC - RL evaluation results on 100 seeds.

H	Collisions %	Mean Speed
1	0%	21.10
2	0%	21.06
3	0%	21.14
4	0%	21.17
5	1%	21.15

These results indicate that the uncertainty-aware safety margins integrate effectively with the learned terminal value function. Similar to deterministic hybrid MPC - RL, the learned DQN approximation successfully provides long-term safety awareness beyond the finite DP horizon. Even at $H = 1$, where deterministic pure MPC failed catastrophically, the hybrid controller remains fully collision-free.

Unlike rollout-based methods, the hybrid controller does not suffer significant degradation as the rollout horizon increases because the terminal estimate is generated directly from the learned value function rather than long heuristic forward simulations. However, a small collision rate reappears at $H = 5$, indicating that the neural-network value approximation remains imperfect under certain long-horizon situations, especially when uncertainty further perturbs the observed traffic state.

The rear safety costs remain relatively small across all horizons and no rear collisions occur, confirming that the uncertainty-aware collision filtering logic continues to successfully reject unsafe lane-change actions.

4.4.9.2 Driving Quality

The stochastic hybrid MPC - RL controller maintains a driving strategy very similar to the deterministic hybrid implementation. The mean speed remains approximately $21.1\,m/s$ across all horizons, consistently below the deterministic and stochastic pure MPC controllers. This again reflects the conservative driving behavior encouraged by the learned RL value function.
The total cost achieves its best value at $H = 3$, where the cost per step decreases to only 329, lower than both deterministic hybrid MPC - RL at the same horizon $(1,015)$ and deterministic pure MPC $H = 5$ (448). The dominant contribution remains the front safety cost, but its magnitude is substantially smaller than rollout-based controllers and generally competitive with stochastic MPC.
Compared to deterministic hybrid MPC - RL, the stochastic implementation produces noticeably larger front safety costs at several horizons, especially for $H = 1$, $H = 4$, and $H = 5$. Similar to stochastic MPC, this increase is mainly caused by the tightened uncertainty-aware safety margins rather than more aggressive behavior. The controller reacts more conservatively to uncertain vehicle states and predicted trajectories, increasing the effective safe-distance penalties.

Table 4.31: Stochastic hybrid MPC - RL mean cost per step on 100 seeds. $C_{col,b}$ is 0 for all cases.

H	**Total**	C_{speed}	C_{accel}	$C_{safe,f}$	$C_{safe,b}$	$C_{col,f}$	C_{lc}	C_{cd}	C_{left}
1	1,475	15.25	2.55	1,400.87	53.47	0	0.0229	0.0184	3.0524
2	585	15.53	1.97	511.11	52.62	0	0.0205	0.0126	3.4646
3	329	14.90	1.80	289.92	19.09	0	0.0194	0.0112	3.4633
4	979	14.70	1.73	864.63	94.20	0	0.0218	0.0146	3.3872
5	1,872	14.85	1.67	1,009.47	174.86	668.11	0.0213	0.0114	3.2741

The speed-tracking cost remains relatively high ($C_{speed} \approx 15$), significantly larger than pure MPC. This confirms that the hybrid controller still prioritizes conservative spacing and long-term safety over aggressive speed tracking. Similarly, the left-lane occupancy cost remains substantially larger than for MPC or rollout approaches, indicating that the RL value function continues to favor extended occupancy of overtaking lanes.
Unlike stochastic rollout, the stochastic hybrid controller does not exhibit catastrophic instability as the horizon increases. Although the total cost increases again at $H = 4$ and $H = 5$, the degradation remains moderate and substantially smaller than rollout-based methods.

4.4.9.3 Computational Cost

Table 4.32 reports the mean computation time per step. Similar to the deterministic implementation, the hybrid MPC - RL controller is the most computationally expensive controller due to the repeated neural-network terminal value evaluations required for every reachable DP terminal state.

The runtime increases from 29.05 ms at $H = 1$ to 175.97 ms at $H = 5$. Compared to deterministic hybrid MPC - RL, the stochastic implementation produces slightly different runtimes, but the overall computational trend remains unchanged. These small timing differences are likely caused primarily by fluctuations in shared cluster workload during the separate simulation runs rather than by the stochastic formulation itself.
Unlike the deterministic implementation, the stochastic hybrid controller at $H = 5$ now remains slightly below the available control interval $\Delta t = 0.2\,s$, requiring approximately 88% of the real-time computation budget. However, the controller still operates very close to the real-time limit at large horizons.
The dominant computational cost remains the repeated forward passes through the [256, 256] DQN network combined with the exponentially growing DP reachable state space. Nevertheless, for moderate horizons ($H \leq 4$), the stochastic hybrid controller remains computationally feasible while providing substantially stronger safety performance than rollout-based approaches.

Table 4.32: Stochastic hybrid MPC - RL mean computation time per step ($\Delta t = 0.2\,\mathrm{s}$).

H	**Time** ($ms/step$)
1	29.05
2	52.84
3	91.63
4	161.35
5	175.97

4.5 Final Controller Comparison

4.5.1 Deterministic Results

Table 4.33 summarizes all deterministic controller variants across prediction horizons $H \in \{1, \ldots, 5\}$.
Three findings stand out. First, the hybrid MPC-RL controller at $H = 2$ achieves the lowest cost per step of all evaluated configurations (257), outperforming pure MPC $H = 5$ (448) and pure RL (551) while producing zero collisions. This is the central result of the thesis, a short DP horizon of only two steps, combined with the RL terminal value function, surpasses the best pure MPC configuration at more that two times the horizon length. Second, the rollout-based controllers consistently fail to match either pure MPC or hybrid MPC-RL in driving quality, with cost per step 3˘28 times higher than the corresponding pure MPC at the same horizon. Increasing the rollout horizon from $H_{\text{roll}} = 10$ to $H_{\text{roll}} = 20$ degrades rather than improves performance for most of lookahead horizons, demonstrating that longer heuristic rollouts amplify prediction error rather than improving the terminal estimate. Third, pure MPC at $H = 1$ produces an88% collision rate due to insufficient lookahead, while the hybrid controller completely resolves this failure mode at the same horizon, confirming that the RL terminal cost compensates for the

Table 4.33: Summary of all deterministic controller results on 100 seeds.

Controller	H	H_{roll}	Coll. %	Cost/step	Time (ms)
Pure MPC	1	-	88%	241,871	2.51
Pure MPC	2	-	0%	1,084	4.34
Pure MPC	3	-	0%	647	6.28
Pure MPC	4	-	1%	1,590	11.37
Pure MPC	5	-	0%	448	17.49
Rollout	1	10	34%	53,562	7.78
Rollout	2	10	1%	2,078	14.42
Rollout	3	10	2%	4,079	21.75
Rollout	4	10	2%	5,203	37.43
Rollout	5	10	3%	5,072	75.14
Rollout	1	20	32%	49,333	11.37
Rollout	2	20	13%	12,735	23.14
Rollout	3	20	10%	8,946	33.10
Rollout	4	20	15%	12,363	59.75
Rollout	5	20	7%	8,942	122.00
Hybrid MPC-RL	1	-	0%	883	17.12
Hybrid MPC-RL	**2**	**-**	**0%**	**257**	**49.77**
Hybrid MPC-RL	3	-	1%	1,015	113.74
Hybrid MPC-RL	4	-	0%	300	171.50
Hybrid MPC-RL	5	-	2%	2,178	242.26
Pure RL	-	-	0%	551	0.32

lack of explicit trajectory optimization. Also, adding only one step to the prediction horizon for pure MPC, drops the 88% collision rate for $H = 1$ to zero for $H = 2$, indicating the effect of having more perspective of future, even only for 0.2 seconds, significantly improves the performance.

4.5.2 Stochastic Results

Table 4.34 summarizes the stochastic variants.

The stochastic hybrid MPC-RL controller at $H = 3$ achieves the best overall result in the stochastic setting, with zero collisions and a cost per step of 329, lower than any deterministic non-hybrid controller and substantially better than all stochastic configurations. The stochastic pure MPC controller at $H = 5$ achieves zero collisions (improving over the deterministic case where $H = 4$ had 1% collision), confirming that the chance-constrained safety tightening eliminates edge cases at no additional computation cost. The stochastic rollout controllers follow the same qualitative pattern as their deterministic counterparts. $H_{\text{roll}} = 10$ outperforms $H_{\text{roll}} = 20$ at all horizons, and cost is substantially higher than MPC or hybrid approaches.

Table 4.34: Summary of all stochastic controller results on 100 seeds.

Controller	H	H_{roll}	Coll. %	Cost/step	Time (ms)
S-Pure MPC	1	-	86%	234,519	2.51
S-Pure MPC	2	-	0%	1,769	4.34
S-Pure MPC	3	-	0%	830	6.28
S-Pure MPC	4	-	0%	1,259	11.37
S-Pure MPC	5	-	0%	754	17.49
S-Rollout	1	10	39%	65,159	9.10
S-Rollout	2	10	0%	1,072	15.05
S-Rollout	3	10	1%	3,261	25.59
S-Rollout	4	10	2%	3,978	41.51
S-Rollout	5	10	1%	2,499	55.77
S-Rollout	1	20	43%	65,150	12.82
S-Rollout	2	20	3%	5,851	23.11
S-Rollout	3	20	4%	6,395	37.12
S-Rollout	4	20	4%	5,129	59.27
S-Rollout	5	20	2%	2,390	85.84
S-Hybrid MPC-RL	1	-	0%	1,475	29.05
S-Hybrid MPC-RL	2	-	0%	585	52.84
S-Hybrid MPC-RL	**3**	**-**	**0%**	**329**	**91.63**
S-Hybrid MPC-RL	4	-	0%	979	161.35
S-Hybrid MPC-RL	5	-	1%	1,872	175.97

4.5.3 Deterministic versus Stochastic

Table 4.35 compares representative deterministic and stochastic controller configurations. The selected cases correspond to the best-performing pure MPC, rollout, and hybrid MPC-RL controllers from the previous sections.

Table 4.35: Comparison between deterministic and stochastic controller variants.

Controller	H	Det. Coll.	Stoch. Coll.	Det. Cost	Stoch. Cost
Pure MPC	2	0%	0%	1,084	1,769
Pure MPC	4	1%	0%	1,590	1,259
Pure MPC	5	0%	0%	448	754
Rollout ($H_{roll} = 10$)	2	1%	0%	2,078	1,072
Rollout ($H_{roll} = 10$)	5	3%	1%	5,072	2,499
Hybrid MPC-RL	2	0%	0%	257	585
Hybrid MPC-RL	3	1%	0%	1,015	329

The introduction of measurement noise and chance-constrained safety has three effects across controller variants. For pure MPC, the stochastic extension increases cost by 30˘68% relative to deterministic MPC (e.g. $H = 5$: 448 vs 754) due to tightened safety margins, but does not reduce safety. In fact it eliminates the collision at $H = 4$ observed in the deterministic case.

For the rollout controllers, the stochastic extension at $H_{\text{roll}} = 10$ reduces cost at $H = 2$ by 48% (2,078 vs 1,072) and improves collision rates, because the tightened margins prevent the rollout from selecting overconfident lane-change trajectories. For the hybrid controller, the effect is mixed. At $H = 2$, cost increases by 128% (257 vs 585) because the RL terminal value was trained in the deterministic setting and does not account for the tightened safety buffers, while at $H = 3$ cost decreases by 68% (1,015 vs 329) and the collision is eliminated. The stochastic hybrid at $H = 3$ with cost 329 and zero collisions represents the overall best result across both deterministic and stochastic settings when safety robustness is required. The computational cost remains within the 200 ms real-time budget for all stochastic configurations up to $H = 4$.

5
Conclusion

This thesis developed, implemented, and evaluated a hybrid Model Predictive Control and Reinforcement Learning (MPC-RL) framework for autonomous truck motion planning in highway driving scenarios. The objective was to combine the safety, interpretability, and constraint-handling capabilities of MPC with the long-horizon decision-making capabilities of RL. To achieve this, a Deep Q-Network (DQN) value function was incorporated into the MPC formulation as a terminal cost approximation, allowing future consequences beyond the optimization horizon to influence the controller's decisions. The framework was evaluated in a SUMO-based highway environment under both deterministic and stochastic conditions and compared against pure MPC, rollout-based controllers, and a pure RL baseline.

The first research question concerned the design of RL value functions that remain compatible with MPC optimization. The results demonstrated that this can be achieved by aligning the RL reward function with the MPC stage cost and using a consistent discount factor. This allowed the learned Q-values to be directly converted into terminal cost estimates without additional scaling or tuning. As a result, the RL component provided long-horizon guidance while remaining fully compatible with the finite-horizon dynamic programming formulation.

The second research question investigated how the proposed hybrid framework compares with pure MPC and pure RL approaches. The results showed that the hybrid controller successfully combined the strengths of both methods. Among all deterministic configurations, the hybrid MPC-RL controller with prediction horizon $H = 2$ achieved the lowest cost per step (257) while maintaining zero collisions. This performance exceeded that of both the best pure MPC configuration ($H = 5$, cost 448) and the pure RL controller (cost 551). Furthermore, the hybrid controller completely eliminated the 88% collision rate observed for pure MPC at $H = 1$, demonstrating that the RL-based terminal value function effectively compensates for limited lookahead. Compared to pure RL, the hybrid controller produced more stable and interpretable behavior while retaining explicit safety constraints.

The third research question addressed the effect of uncertainty and whether it can be mitigated through stochastic extensions. Measurement uncertainty was modeled through additive Gaussian noise, while chance-constrained safety margins were incorporated into the MPC formulation. The stochastic hybrid controller achieved its best performance at $H = 3$, obtaining zero collisions and a cost per step of 329, outperforming all other stochastic configurations. The results showed that stochastic safety tightening successfully eliminated several rare collision cases observed in deterministic experiments while preserving real-time feasibility. Although the stochastic formulation generally increased conservatism for pure MPC, it improved robustness

and consistency for the hybrid controller, indicating that uncertainty-aware MPC and RL-based terminal value estimation complement each other effectively.

The final research question concerned the relationship between prediction horizon, discount factor, and computational efficiency. The experiments revealed that increasing the prediction horizon does not necessarily improve performance. While longer horizons initially provide additional foresight, the associated computational cost grows rapidly and the performance gains eventually diminish. The best deterministic trade-off was achieved at $H = 2$, while the best stochastic performance was achieved at $H = 3$. These results demonstrate that the RL terminal value function can successfully replace a significant portion of the explicit prediction horizon. This can be interpreted as meaning that inaccuracy in prediction due to position error, even negligible, or not considering the lane changes for surrounding vehicles can have a negative impact on the performance of hybrid model. The discount factor also proved critical, as it determines the balance between immediate and future consequences.

The rollout-based controllers provided an important point of comparison. Although rollout methods are theoretically capable of extending the planning horizon, the results showed that increasing the rollout horizon often degraded performance rather than improving it. Longer rollout horizons accumulated prediction errors and uncertainty, specially due to inaccurate heuristic policy. In both deterministic and stochastic settings, rollout controllers consistently produced higher costs than either pure MPC or hybrid MPC-RL. These findings suggest that learned terminal value functions provide a more effective mechanism for incorporating long-term information than heuristic rollout approximations.

Beyond the quantitative performance improvements, this work demonstrates the potential of combining learning-based and model-based control approaches for safety-critical autonomous systems. Rather than replacing MPC with RL, the results show that a carefully designed hybrid architecture can exploit the advantages of both methods. MPC provides stability, transparency, and explicit constraint handling, while RL contributes long-term strategic information that is difficult to obtain through finite-horizon optimization alone. The introduction of the ghost-vehicle representation further improved the consistency of vehicle tracking during lane-change maneuvers and contributed to more reliable safety assessment throughout the planning process. The main limitation of the hybrid controller is computational cost. Although $H = 2$ and $H = 3$ achieved excellent performance while satisfying real-time requirements, the computation time at $H = 5$ exceeded the 200 ms control budget, limiting the practical use of longer horizons.

Several directions remain for future work. The current experiments were conducted on a straight highway scenario with simplified vehicle dynamics and limited traffic complexity. Future studies should investigate more realistic road geometries, merging scenarios, intersections, and denser traffic environments. Extending the vehicle model to more complicated models, such as bicycle model, to include lateral dynamics and steering control would improve realism and broaden the applicability of the framework. Another promising direction is the training of the RL component directly under stochastic observations, allowing uncertainty to be incorporated into the learned value function itself rather than only through the MPC safety con-

straints. Another valuable improvement would be the extension of the state representation to explicitly include surrounding vehicle information and traffic interaction features. The investigation of scalable optimization and parallelization techniques for RL model to improve the computation time would be another direction for future research. Future work should also explore adaptive horizon selection, more advanced reinforcement learning algorithms, multi-agent traffic interaction models, and hardware-in-the-loop validation on physical vehicle platforms. Such developments would provide further insight into the scalability and practical deployment of hybrid MPC-RL architectures for autonomous driving applications.

Bibliography

[1] D. Pathare, L. Laine, and M. H. Chehreghani, "Tactical decision making for autonomous trucks by deep reinforcement learning with total cost of operation based reward," *Artificial Intelligence Review*, vol. 59, no. 27, 2025.

[2] Pathare, D. and Laine, L. and Chehreghani, M. H., "Improved tactical decision making and control architecture for autonomous truck in sumo using reinforcement learning," in *2023 IEEE International Conference on Big Data (BigData)*, Sorrento, Italy, 2023, pp. 4635–4644.

[3] R. Reiter, J. Hoffmann, D. Reinhardt, F. Messerer, K. Baumgärtner, S. Sawant, J. Bödecker, M. Diehl, and S. Gros, "Synthesis of model predictive control and reinforcement learning: Survey and classification," *Annual Reviews in Control*, vol. 61, p. 101045, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1367578826000015

[4] E. Börve, N. Murgovski, and L. Laine, "Interaction-aware trajectory prediction and planning in dense highway traffic using distributed model predictive control," in *2023 62nd IEEE Conference on Decision and Control (CDC)*, Singapore, Singapore, 2023, pp. 6124–6129.

[5] S. Mallick, G. Battocletti, Q. Dong, A. Dabiri, and B. De Schutter, "Learning-based mpc for fuel efficient control of autonomous vehicles with discrete gear selection," *IEEE Control Systems Letters*, vol. 9, pp. 1117–1122, 2025.

[6] C.-J. Hoel, K. Wolff, and L. Laine, "Tactical decision-making in autonomous driving by reinforcement learning with uncertainty estimation," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, Las Vegas, NV, USA, 2020, pp. 1563–1569.

[7] P. Gupta, J. M. Smereka, and Y. Jia, "Reinforcement learning compensated model predictive control for off-road driving on unknown deformable terrain," *arXiv preprint arXiv:2408.09253*, 2024. [Online]. Available: https://arxiv.org/abs/2408.09253

[8] Bertsekas, D. P., "Model predictive control and reinforcement learning: A unified framework based on dynamic programming," *IFAC-PapersOnLine*, vol. 58, no. 18, pp. 363–383, 2024.

[9] Z. Chen, J. Lai, P. Li, O. I. Awad, and Y. Zhu, "Prediction horizon-varying model predictive control (mpc) for autonomous vehicle control," *Electronics*, vol. 13, no. 8, p. 1442, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/8/1442

[10] B. Zarrouki, M. Spanakakis, and J. Betz, "A safe reinforcement learning driven weights-varying model predictive control for autonomous vehicle motion control," in *2024 IEEE Intelligent Vehicles Symposium (IV)*, 2024, pp. 1401–1408.

[11] Rawlings, J. B. and Mayne, D. Q. and Diehl, M., *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Nob Hill Pub., 2024. [Online]. Available: https://cir.nii.ac.jp/crid/1971150415095544201

[12] B. Brito, A. Agarwal, and J. Alonso-Mora, "Learning interaction-aware guidance for trajectory optimization in dense traffic scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 18 808–18 821, 2022.

[13] Sutton, R. S. and Barto, A. G., *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[14] Sallab, A. E. and Abdou, M. and Perot, E. and Yogamani, S., "End-to-end deep reinforcement learning for lane keeping assist," in *NIPS Workshop on Machine Learning for Intelligent Transportation Systems*, 2016. [Online]. Available: https://arxiv.org/abs/1612.04340

[15] Hoel, C.-J. and Wolff, K. and Laine, L., "Automated speed and lane change decision making using deep reinforcement learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2148–2155.

[16] L. Wang, S. Yang, K. Yuan, Y. Huang, and H. Chen, "A combined reinforcement learning and model predictive control for car-following maneuver of autonomous vehicles," *Chinese Journal of Mechanical Engineering*, vol. 36, no. 1, p. 80, 2023.

[17] Y. Li, C. Huang, D. Yang, W. Liu, and J. Li, "Learning based mpc for autonomous driving using a low dimensional residual model," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 10 958–10 964.

[18] R. Reiter, J. Hoffmann, J. Boedecker, and M. Diehl, "A hierarchical approach for strategic motion planning in autonomous racing," in *2023 European Control Conference (ECC)*, 2023, pp. 1–8.

[19] N. Albarella, D. G. Lui, A. Petrillo, and S. Santini, "A hybrid deep reinforcement learning and optimal control architecture for autonomous highway driving," *Energies*, vol. 16, no. 8, p. 3490, 2023.

[20] Bertsekas, D. P., *A Course in Reinforcement Learning*. Athena Scientific, 2023.

[21] M. Morita, S. Yamamori, S. Yagi, N. Sugimoto, and J. Morimoto, "Goal-conditioned terminal value estimation for real-time and multi-task model predictive control," *arXiv preprint arXiv:2410.04929*, 2024. [Online]. Available: https://arxiv.org/abs/2410.04929

[22] S. Chintalapudi, N. Wagener, and B. Boots, "Improving model predictive control with value function approximation," Undergraduate Research Option Thesis, Georgia Institute of Technology, 2019.

[23] E. Camacho and C. Alba, *Model Predictive Control*, ser. Advanced Textbooks in Control and Signal Processing. Springer London, 2013. [Online]. Available: https://books.google.se/books?id=tXZDAAAAQBAJ

[24] C. E. García, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[25] R. Bellman, R. Corporation, and K. M. R. Collection, *Dynamic Programming*, ser. Rand Corporation research study. Princeton University Press, 1957. [Online]. Available: https://books.google.se/books?id=wdtoPwAACAAJ

[26] D. Bertsekas, *Dynamic Programming and Optimal Control: Volume II; Approximate Dynamic Programming*, ser. Athena Scientific optimization and computation series. Athena Scientific, 2012. [Online]. Available: https://books.google.se/books?id=C1JEEAAAQBAJ

[27] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007, vol. 703.

[28] S. Boyd, "Stochastic model predictive control," Lecture notes for EE364b, Stanford University, accessed: 2026-05-04. [Online]. Available: https://stanford.edu

[29] D. Ng, "Stochastic model predictive control," Ph.D. dissertation, Oxford University, UK, 2011.

[30] E. Börve, N. Murgovski, and L. Laine, "Tight collision avoidance for stochastic optimal control: with applications in learning-based, interactive motion planning," 2025. [Online]. Available: https://arxiv.org/abs/2510.25324

[31] A. Gelb *et al.*, *Applied optimal estimation*. MIT press, 1974.

[32] A. Charnes and W. W. Cooper, "Chance-constrained programming," *Management science*, vol. 6, no. 1, pp. 73–79, 1959.

[33] B. Kouvaritakis and M. Cannon, *Model predictive control*. Springer, 2016.

[34] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[35] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3–4, pp. 279–292, 1992.

[36] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[37] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning*, vol. 8, no. 3, pp. 293–321, 1992.

[38] J. Cui, J. Gao, X. Liu, Y. Liu, and S. Yu, "A characterization method of terminal ingredients for nonlinear mpc using value-based reinforcement learning," *Automatica*, vol. 183, p. 112574, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109825004698

[39] F. Moreno-Mora, L. Beckenbach, and S. Streif, "Predictive control with learning-based terminal costs using approximate value iteration," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 3874–3879, 2023, 22nd IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S240589632301724X

[40] S. V. Raković and S. Zhang, "Model predictive control with implicit terminal ingredients," *Automatica*, vol. 151, p. 110942, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109823000936

[41] S. Menta, J. Warrington, J. Lygeros, and M. Morari, "Learning q-function approximations for hybrid control problems," *IEEE Control Systems Letters*, vol. 6, pp. 1364–1369, 2022.

[42] L. Schwenkel, A. Hadorn, M. A. Müller, and F. Allgöwer, "Linearly discounted economic mpc without terminal conditions for periodic optimal operation," *Automatica*, vol. 159, p. 111393, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109823005605

[43] M. Zanon and S. Gros, "A new dissipativity condition for asymptotic stability of discounted economic mpc," 2022. [Online]. Available: https://arxiv.org/abs/2106.09377

[44] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner, "Microscopic traffic simulation using sumo," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2575–2582.

[45] S. Krauss, "Microscopic modeling of traffic flow: investigation of collision free vehicle dynamics," Ph.D. dissertation, Universität zu Köln, Cologne, Germany, 1998.

[46] J. Erdmann, "Lane-changing model in sumo," *Transportation Research Record*, vol. 2522, no. 1, pp. 10–17, 2015.

A

Appendix 1

A.1 Trajectory Prediction Error Tables

Tables A.1 through A.5 report the prediction errors for Method 1 (Reactive Multi-Agent Projection) at each horizon step $h = 1, \ldots, 5$.

Table A.1: Method 1 prediction error at $h = 1$ step ($0.2\,s$ ahead).

Role	$\bar{e}_s$ **(m)**	σ_{e_s} **(m)**	$\bar{e}_v$ **(m/s)**	σ_{e_v} **(m/s)**
Front	-0.001	0.009	0.015	0.105
Front-left	0.000	0.010	0.037	0.143
Front-right	0.000	0.008	0.037	0.135
Back-left	0.001	0.015	0.151	0.238
Back-right	0.001	0.013	0.083	0.189

Table A.2: Method 1 prediction error at $h = 2$ steps ($0.4\,s$ ahead).

Role	$\bar{e}_s$ **(m)**	σ_{e_s} **(m)**	$\bar{e}_v$ **(m/s)**	σ_{e_v} **(m/s)**
Front	0.002	0.038	0.033	0.227
Front-left	0.008	0.046	0.085	0.290
Front-right	0.007	0.040	0.071	0.265
Back-left	0.023	0.059	0.246	0.433
Back-right	0.016	0.058	0.180	0.396

Table A.3: Method 1 prediction error at $h = 3$ steps ($0.6\,s$ ahead).

Role	$\bar{e}_s$ **(m)**	σ_{e_s} **(m)**	$\bar{e}_v$ **(m/s)**	σ_{e_v} **(m/s)**
Front	0.009	0.084	0.052	0.328
Front-left	0.026	0.106	0.124	0.431
Front-right	0.019	0.091	0.089	0.360
Back-left	0.073	0.144	0.365	0.642
Back-right	0.047	0.136	0.244	0.570

Table A.4: Method 1 prediction error at $h = 4$ steps ($0.8\,s$ ahead).

Role	$\bar{e}_s$ **(m)**	σ_{e_s} **(m)**	$\bar{e}_v$ **(m/s)**	σ_{e_v} **(m/s)**
Front	0.020	0.149	0.068	0.417
Front-left	0.052	0.190	0.159	0.548
Front-right	0.034	0.159	0.107	0.451
Back-left	0.147	0.273	0.478	0.844
Back-right	0.093	0.240	0.305	0.721

Table A.5: Method 1 prediction error at $h = 5$ steps ($1.0\,s$ ahead).

Role	$\bar{e}_s$ **(m)**	σ_{e_s} **(m)**	$\bar{e}_v$ **(m/s)**	σ_{e_v} **(m/s)**
Front	0.031	0.222	0.074	0.471
Front-left	0.079	0.295	0.181	0.652
Front-right	0.059	0.262	0.133	0.563
Back-left	0.243	0.436	0.581	1.033
Back-right	0.167	0.393	0.401	0.908

Tables A.6 through A.10 report the prediction errors for Method 2 (Constant Acceleration Extrapolation) at each horizon step $h = 1, \ldots, 5$.

Table A.6: Method 2 prediction error at $h = 1$ step ($0.2\,s$ ahead).

Role	$\bar{e}_s$ **(m)**	σ_{e_s} **(m)**	$\bar{e}_v$ **(m/s)**	σ_{e_v} **(m/s)**
Front	-0.001	0.009	0.000	0.028
Front-left	0.000	0.009	0.000	0.027
Front-right	0.000	0.009	0.000	0.026
Back-left	0.001	0.015	0.000	0.039
Back-right	0.000	0.011	0.000	0.033

Table A.7: Method 2 prediction error at $h = 2$ steps ($0.4\,s$ ahead).

Role	$\bar{e}_s$ **(m)**	σ_{e_s} **(m)**	$\bar{e}_v$ **(m/s)**	σ_{e_v} **(m/s)**
Front	-0.002	0.027	0.000	0.078
Front-left	0.000	0.031	0.000	0.086
Front-right	0.000	0.023	0.000	0.067
Back-left	0.000	0.034	0.001	0.089
Back-right	-0.001	0.034	0.001	0.091

Table A.8: Method 2 prediction error at $h = 3$ steps ($0.6\,s$ ahead).

Role	$\bar{e}_s$ **(m)**	σ_{e_s} **(m)**	$\bar{e}_v$ **(m/s)**	σ_{e_v} **(m/s)**
Front	−0.002	0.041	0.000	0.114
Front-left	0.000	0.045	0.000	0.122
Front-right	0.000	0.040	0.000	0.108
Back-left	0.001	0.055	0.002	0.147
Back-right	−0.002	0.058	0.001	0.159

Table A.9: Method 2 prediction error at $h = 4$ steps ($0.8\,s$ ahead).

Role	$\bar{e}_s$ **(m)**	σ_{e_s} **(m)**	$\bar{e}_v$ **(m/s)**	σ_{e_v} **(m/s)**
Front	−0.002	0.064	0.000	0.160
Front-left	0.000	0.074	0.000	0.184
Front-right	0.001	0.065	0.000	0.162
Back-left	0.003	0.086	0.004	0.210
Back-right	−0.001	0.086	0.003	0.212

Table A.10: Method 2 prediction error at $h = 5$ steps ($1.0\,s$ ahead).

Role	$\bar{e}_s$ **(m)**	σ_{e_s} **(m)**	$\bar{e}_v$ **(m/s)**	σ_{e_v} **(m/s)**
Front	−0.002	0.089	0.000	0.194
Front-left	0.000	0.112	0.001	0.245
Front-right	0.001	0.096	0.001	0.210
Back-left	0.005	0.133	0.006	0.290
Back-right	0.000	0.131	0.005	0.286

Tables A.11 through A.15 report the final prediction and measurement error statistics for each prediction horizon step $h = 1, \ldots, 5$.

Table A.11: Final prediction and measurement error statistics at $h = 1$ step ($0.2\,s$ ahead).

Role	$\bar{e}_s$ **(m)**	σ_{e_s} **(m)**	$\bar{e}_v$ **(m/s)**	σ_{e_v} **(m/s)**
Front	−0.002	0.509	−0.002	0.500
Front-left	0.003	0.511	0.000	0.503
Front-right	0.003	0.512	−0.002	0.505
Back-left	−0.001	0.510	−0.001	0.500
Back-right	0.000	0.510	0.007	0.499

Table A.12: Final prediction and measurement error statistics at $h = 2$ steps ($0.4\,s$ ahead).

Role	$\bar{e}_s$ **(m)**	σ_{e_s} **(m)**	$\bar{e}_v$ **(m/s)**	σ_{e_v} **(m/s)**
Front	−0.001	0.538	−0.001	0.515
Front-left	0.004	0.539	0.000	0.513
Front-right	0.004	0.540	−0.002	0.511
Back-left	−0.002	0.540	0.001	0.514
Back-right	−0.004	0.542	−0.001	0.513

Table A.13: Final prediction and measurement error statistics at $h = 3$ steps ($0.6\,s$ ahead).

Role	$\bar{e}_s$ **(m)**	σ_{e_s} **(m)**	$\bar{e}_v$ **(m/s)**	σ_{e_v} **(m/s)**
Front	−0.004	0.585	0.000	0.524
Front-left	0.006	0.584	0.004	0.527
Front-right	0.009	0.582	0.005	0.524
Back-left	−0.003	0.586	0.003	0.535
Back-right	0.001	0.584	0.006	0.529

Table A.14: Final prediction and measurement error statistics at $h = 4$ steps ($0.8\,s$ ahead).

Role	$\bar{e}_s$ **(m)**	σ_{e_s} **(m)**	$\bar{e}_v$ **(m/s)**	σ_{e_v} **(m/s)**
Front	−0.005	0.645	−0.001	0.546
Front-left	0.001	0.646	−0.002	0.553
Front-right	0.004	0.646	0.003	0.543
Back-left	0.002	0.649	0.007	0.560
Back-right	−0.005	0.644	0.008	0.556

Table A.15: Final prediction and measurement error statistics at $h = 5$ steps ($1.0\,s$ ahead).

Role	$\bar{e}_s$ **(m)**	σ_{e_s} **(m)**	$\bar{e}_v$ **(m/s)**	σ_{e_v} **(m/s)**
Front	−0.002	0.719	0.002	0.570
Front-left	0.006	0.719	0.000	0.583
Front-right	0.008	0.722	0.001	0.572
Back-left	−0.002	0.722	0.006	0.599
Back-right	0.001	0.717	0.009	0.578

A.2 Hyperparameters

Table A.16 represents the hyperparameters of system dynamics and cost weights implemented in this thesis.

Table A.16: System Dynamics and Cost Hyperparameters.

Symbol	Description	Value
H	Pure MPC Prediction Horizon	5 steps
H_{roll}	Rollout Horizon	20 steps
Δt	Sampling Time Step	0.2 s
S_{STEP}	Longitudinal Position Resolution (1 Δt)	0.2 m
V_{STEP}	Velocity Resolution (1 Δt)	0.2 m/s
T_{TD}	Total Cycle Duration	5.0 s
T_{LC}	Lane Change Duration	2.0 s
T_{CD}	Lane Change Cool Down Duration ($T_{TD} - T_{LC}$)	3.0 s
N_{steps}	Lane Change Counter Threshold ($T_{LC}/\Delta t$)	10 steps
C_{steps}	Total Counter Duration ($T_{TD}/\Delta t$)	25 steps
C_{start}	Transition/Cooldown Threshold ($C_{steps} - N_{steps}$)	15 steps
N_{lanes}	Total Number of Lanes	3
v_{min}, v_{max}	Velocity Limits	0.0, 33.33 m/s
a_{min}, a_{max}	Acceleration Limits	-2.0, 2.0 m/s^2
v_{ref}	Reference Velocity	25.0 m/s
w_v	Speed Tracking Weight	1.0
w_a	Acceleration Weight	1.0
w_{front}, w_{back}	Safety Penalty Weights	1000.0
w_{LL}	Left Lane Occupancy Weight	2.0
P_{LC}	Lane Change Execution Penalty	10.0
P_{CD}	Lane Change Cooldown Penalty	30.0
P_{col}	Collision Penalty	10^8
D_0	Static Safe Distance Buffer	5.0 m
T_{head}	Desired Time Headway	2.0 s
d_{sensor}	Sensor Range For Data Collecting	150 m
a_{break}	Follower maximum deceleration (collision check)	2.0 m/s^2
γ	Discount factor (stage cost weighting)	0.98
τ	Krauss reaction time	1 s
σ_s	Standard deviations of position measurement noise	0.5
σ_v	Standard deviations of velocity measurement noise	0.5
σ_a	Standard deviations of acceleration measurement noise	0.2

Table A.17 represents the training hyperparameters for RL.

Table A.17: DQN Training Hyperparameters.

Parameter	**Description**	**Value**
γ	Discount factor	0.98
$\|\mathcal{D}\|$	Replay buffer size	$400,000$
B	Mini-batch size	64
α_{lr}	Learning rate	10^{-4}
τ_{target}	Target network update interval	$10,000$ steps
f_{train}	Training frequency	every 4 env steps
ϵ_{start}	Initial exploration rate	1.0
ϵ_{end}	Final exploration rate	0.01
$f_{explore}$	Exploration fraction of total steps	0.15
N_{start}	Learning starts	$5,000$ steps
N_{total}	Total training steps	$40,000,000$
Network	Hidden layers	$[256, 256]$, ReLU
r_{scale}	Reward scaling factor	10^{-5}

DEPARTMENT OF ELECTRICAL ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY