Stochastic MPC for Autonomous Vehicles in Uncertain Situations

Navigating Through Complexity: Enhancing Autonomous Heavy Vehicle Control with Stochastic MPC in Uncertain Environments

Saeed Ponanrikkal Salih and Qun Zhang

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2024

www.chalmers.se

Master's thesis 2024

Stochastic MPC for Autonomous Vehicles in Uncertain Situations

Navigating Through Complexity: Enhancing Autonomous Heavy Vehicle Control with Stochastic MPC in Uncertain Environments

QUN ZHANG

SAEED PONNARIKKAL SALIH

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems, Control, and Mechatronics
Chalmers University of Technology
Gothenburg, Sweden 2024

Stochastic MPC for Autonomous Vehicles in Uncertain Situations
Navigating Through Complexity: Enhancing Autonomous Heavy Vehicle Control with Stochastic MPC in Uncertain Environments
QUN ZHANG, SAEED PONNARIKKAL SALIH

Supervisors:
Erik Börve, Volvo GTT/Department of Electrical Engineering, Chalmers University of Technology
Leo Laine, Volvo GTT/Department of Mechanics and Maritime Sciences, Chalmers University of Technology

Examiner: Nikolce Murgovski, Department of Electrical Engineering, Chalmers University of Technology

Master's Thesis 2024
Department of Electrical Engineering
Division of Systems Controls and Mechatronics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The ego vehicle traversing through heavy traffic simulated in the CARLA environment.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Stochastic MPC for Autonomous Vehicles in Uncertain Situations
Navigating Through Complexity: Enhancing Autonomous Heavy Vehicle Control with Stochastic MPC in Uncertain Environments
QUN ZHANG, SAEED PONNARIKKAL SALIH
Department of Electrical Engineering
Chalmers University of Technology

Abstract

In this thesis, an Model Predictive Control (MPC) based trajectory planning algorithm is first introduced for controlling trucks on highways. Given the uncertainties that exist between theoretical models and real vehicles, this study further analyzes these uncertainties and proposes an Stochastic Model Predictive Control (SMPC) based trajectory planning algorithm. The algorithm avoids collisions by tightening constraints and is validated in the CARLA simulation environment. Experimental results show that the SMPC-based trajectory planning algorithm has obvious advantages in terms of safety performance compared with the standard MPC. However, the method also sacrifices certain driving performance and increases computational complexity, which is mainly due to the tightened traffic constraints. This study not only verifies the effectiveness of SMPC in handling uncertainty and enhancing safety but also provides both an experimental and theoretical basis for future work.

Keywords: Model Predictive Control, Stochastic Model Predictive Control, Collision Avoidance, CARLA Simulation, Optimal Control, Chance Constraints.

Acknowledgements

Firstly, we would like to acknowledge our deepest gratitude to Erik Börve, industrial PhD student at Volvo GTT, and the Department of Electrical Engineering at Chalmers University of Technology, who supervised our thesis and taught us many essential skills and foundational knowledge. Without his constant guidance, this thesis project would have faced many challenges.

In addition, we are especially grateful to our examiner Professor Nikolce Murgovski at the Department of Electrical Engineering at Chalmers University of Technology. He was the first person to introduce us to the Model Predictive Control (MPC) topic and guided us in exploring this field. His support throughout our graduate studies was instrumental in completing this thesis project.

We would also like to thank our industrial partner, Volvo GTT, and Professor Leo Laine at Volvo GTT for providing essential tools, funding, and guidance throughout the project. Their support was crucial for the successful completion of our thesis. Additionally, we extend our gratitude to Chalmers University of Technology for their invaluable guidance and support during the project.

Finally, we would like to express our sincere gratitude to our family and friends. They have been with us all along the way and have inspired us to overcome all the challenges to reach this important milestone.

Qun Zhang & Saeed Ponnarikkal Salih, Gothenburg, June 2024

List of Acronyms

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

AVs	Autonomous Vehicles
MPC	Model Predictive Control
RMPC	Robust Model Predictive Control
SMPC	Stochastic Model Predictive Control
EKF	Extended Kalman Filter
DM	Decision Master
COG	Center of Gravity
CV	Constant Velocity
PID	Proportional Integral Derivative
CDF	Cumulative Distribution Function
GLQC	Gaussian Linear Quadratic Control
LQR	Linear Quadratic Regulator
IMU	Inertial Measurement Unit

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis. The following nomenclature is sorted in order of appearance:

- Vectors are bold lower case letters "$\mathbf{x}$"
- Matrices are bold upper case letters "$\mathbf{G}$"
- Sets are in blackboard-bold letters "$\mathbb{S}$"
- Estimation or Prediction are marked by "$\hat{y}$"
- Time derivatives are marked by "$\dot{z}$"

Indices

e	Index for ego vehicle
(i)	Index for surrounding vehicles
j	Index for lane changes (left or right) or iterations in the optimization process
k	Time step index or index representing time steps or stages in the prediction horizon
(l)	Index for leading vehicle
n	Index for number of vehicles
nc, lc, rc	Indices for different control strategies (trailing, left lane change, right lane change)

Sets

$\mathbb{V}$	Set of vehicles
$\mathbb{L}$	Set of lanes
$\mathbb{X}$	Set of states
$\mathbb{U}$	Set of inputs
$\mathbb{W}$	Set of Noise

Parameters

l^e	Wheelbase of the vehicle
l^e_{lf}	Front wheelbase length
l^e_{lr}	Rear wheelbase length
T_s	Sampling Time
$\mathbf{C}$	Measurement matrix
$\mathbf{r}_k$	Measurement noise
$\mathbf{Q}_w$	Process noise covariance for Kalman Filter
$\mathbf{R}_w$	Measurement noise covariance
$\mathbf{x}^e_{\min}$	Minimum allowed state values of the ego vehicle
$\mathbf{x}^e_{\max}$	Maximum allowed state values of the ego vehicle
$\mathbf{u}^e_{\min}$	Minimum allowed control input values of the ego vehicle
$\mathbf{u}^e_{\max}$	Maximum allowed control input values of the ego vehicle
T_{pred}	Prediction horizon
N	Time horizon
T	Minimum headway distance required for braking
d	Minimum longitudinal distance to the vehicle ahead
L^{truck}	Length of the truck
w	Lateral span of the truck
$\mathbf{Q}_p$	State weighting matrix
$\mathbf{Q}_s$	Slack variable weighting matrix
$\mathbf{R}$	Input weighting matrix
$\mathbf{Q}_f$	Terminal state weighting matrix
$\mathbf{R}_d$	Input change weighting matrix
q_e	Cost function significance
q_s	Controller switching parameter
Δt	Time discretization step (time interval)
K_p	Proportional gain for Longitudinal control
K_i	Integral gain for Longitudinal control
K_d	Derivative gain for Longitudinal control
$K_{p,n}$	Proportional gain for near point in PI controller
$K_{i,n}$	Integral gain for near point in PI controller
$K_{p,f}$	Proportional gain for far point in proportional controller

x_n	Longitudinal distance to near point (m)
x_f	Longitudinal distance to far point (m)
f	Update frequency of MPC
h	General selection vector of the constraints for states and inputs
b	General upper bound of the constraints for states and inputs
$\mathbf{Q}$	General weighting matrix
ϵ	Small predetermined probability of constraint violation
p	The minimum probability of constraint satisfaction
v^e_{ref}	Reference velocity (m/s)
$y^e_{\text{ref},l}$	Reference y position at left lane (m)
$y^e_{\text{ref},m}$	Reference y position at middle lane (m)
$y^e_{\text{ref},r}$	Reference y position at right lane (m)
d_{min}	Minimum distance between vehicles in test scenarios (m)
v_{min}	Minimum velocity of vehicles in test scenarios (m/s)
v_{max}	Maximum velocity of vehicles in test scenarios (m/s)
n_{vehicles}	Number of vehicles of vehicles in test scenarios

Variables

x^e	Longitudinal position (m)
y^e	Lateral position (m)
v^e_x	Longitudinal velocity (m/s)
ψ^e	Orientation angle (rad)
δ^e	Steering angle (rad)
a^e	Acceleration (m/s^2)
β^e	Slip angle (rad)
$\mathbf{x}^e$	Ego vehicle states
$\mathbf{u}^e$	Ego vehicle control inputs
$\bar{\mathbf{x}}^e$	Operating point of states
$\bar{\mathbf{u}}^e$	Operating point of control inputs
$\mathbf{A}_c$	State transition matrix in continue time
$\mathbf{B}_c$	Control input matrix in continue time
$\mathbf{g}_c$	Remainder vector after linearization in continue time
$\mathbf{A}_d$	State transition matrix in discrete time

$\mathbf{B}_d$	Control input matrix in discrete time
$\mathbf{g}_d$	Remainder vector after linearization in discrete time
$\mathbf{y}^e$	Measurement vector
$\hat{\mathbf{x}}^e$	Estimation of ego vehicle states
$\mathbf{P}(k+1\|k)$	Covariance matrix of the predicted estimate at time $k+1$
$\mathbf{P}(k+1\|k+1)$	Covariance matrix of the updated estimate at time $k+1$
$\mathbf{L}(k+1)$	Kalman Gain at time $k+1$
$\hat{\mathbf{x}}^e(k+1\|k)$	Predicted state estimate at time $k+1$
$\hat{\mathbf{x}}^e(k+1\|k+1)$	Updated state estimate at time $k+1$
$\hat{v}_x^{(l)}$	Estimated longitudinal velocity of the leading vehicle
$\hat{x}^{(i)}(k)$	Estimated longitudinal position of vehicle i at time step k
$\hat{y}^{(i)}(k)$	Estimated lateral position of vehicle i at time step k
$\hat{x}^{(i)}(k+1)$	Predicted longitudinal position of vehicle i at time step $k+1$
$\hat{y}^{(i)}(k+1)$	Predicted lateral position of vehicle i at time step $k+1$
$\hat{\mathbf{x}}^{(i)}$	Estimated vehicle i states
$\hat{\mathbf{u}}^{(i)}$	Estimated vehicle i inputs
$\hat{\mathbf{X}}_j^{(i)}$	Surrounding vehicle states throughout the total time horizon
$\hat{x}^{(l)}$	Estimated longitudinal position of the leading traffic vehicle (m)
$\hat{v}_x^{(l)}$	Estimated longitudinal velocity of the leading vehicle (m/s)
$\hat{x}^e$	Longitudinal position estimation of the ego vehicle (m)
$\hat{y}^e$	Lateral position estimation of the ego vehicle (m)
$\varepsilon_1^{(i)}$	Scaling variable for lateral constraint
$\varepsilon_2^{(i)}$	Scaling variable for lateral constraint
λ^e	slack variables
$\hat{\mathbf{X}}^e$	Ego vehicle states over the horizon N
$\mathbf{U}^e$	Ego vehicle inputs over the horizon N
$\mathbf{\Lambda}^e$	Slack variables over the horizon N
$\hat{\mathbf{X}}^{e*}$	Optimal ego vehicle trajectory over the horizon N
$e_v(k)$	Speed error at time k (m/s)
$v^e(k)$	Actual vehicle speed at time k (m/s)
$v^{e*}(k)$	Desired target speed at time k (m/s)
$a(k)$	Control output used to modulate either the throttle or brake at time k (m/s^2)
α_n	Angle corresponding to the near point (rad)
α_f	Angle corresponding to the far point (rad)

$\delta^e(k)$	Steering control input at time step k (rad)
$e_y(k)$	Lateral position error (m)
y_n	Lateral distance to near point (m)
y_f	Lateral distance to far point (m)
$\mathbf{w}(k)$	The worst-case disturbance at time step k
$\tilde{\mathbf{x}}(k)$	Nominal system state at time k
$\tilde{\mathbf{u}}(k)$	Nominal system input at time k
$\mathbf{u}_r(k)$	Robust input at time k
$\mathbf{K}$	Feedback gain
$\hat{\mathbf{\Sigma}}(k)$	The covariance matrix of the state at time k
$\mathbf{K}_{LQR}$	LQR Feedback gain
$\mathbf{P}$	Stationary Kalman Filter corvariance
$\hat{\mathbf{\Sigma}}^e(k)$	The covariance matrix of the ego vehicle states at time k
$\mathbf{K}_{LQR}$	LQR Feedback gain
$\mathbf{P}_{LQR}$	Stationary corvariance of LQR
$\mathbf{\Sigma}_w$	The process noise for propagation of uncertainty
$\varepsilon_{2,\text{SMPC}}$	Scaling variable for lateral constraint for SMPC
$\varepsilon_{3,\text{SMPC}}$	Scaling variable for lateral constraint for SMPC

Functions

$f(\cdot)$	State dynamics of ego vehicle
$\Pi_{\text{CV}}(\cdot)$	Position prediction function under the constant velocity model for vehicle i
$\varepsilon_0^{(i)}(\cdot)$	Function to scale the constraint laterally
$\varepsilon_3^{(i)}(\cdot)$	Function to shift the constraint laterally
$c_j(\cdot)$	Constraint function
$c_{j,\text{SMPC}}(\cdot)$	Constraint function for SMPC
$J_j^e(\cdot)$	Cost function of MPC for optimization
$J_j^s(\cdot)$	Cost function of switching the controller j
$J_{j,\text{SMPC}}^e(\cdot)$	Cost function of SMPC for optimization
$h_j^{\text{e}}(\cdot)$	General MPC constraint function with slack variables
$h_{j,\text{SMPC}}^{\text{e}}(\cdot)$	General SMPC constraint function with slack variables
$g(\cdot)$	Decision function selects the optimal control sequence

$H(\cdot)$	The history of the previous choices made
$P(\cdot)$	The probability that the constraints are satisfied
$\varepsilon_{1,\mathrm{SMPC}}(\cdot)$	Function to scale the constraint laterally for SMPC
$\varepsilon_{4,\mathrm{SMPC}}(\cdot)$	Function to shift the constraint laterally for SMPC

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

As the number of vehicles on the road increases, traffic accidents are also rising rapidly [1]. There are about 1.35 million annual fatalities attributed to traffic accidents around the world, the leading cause of death among young people. To reduce this, global commitments have been encouraged as part of the United Nation's Decade of Action for Road Safety 2011–2020 [2]. Most road accidents are attributed to human errors [3]. Compared to human-driven vehicles, Autonomous Vehicles (AVs) are more precise in positioning, speed control, steering, and reaction time, and they also have the potential to further improve them. Therefore, it is imperative to use autonomous vehicles, as they are proven to reduce such errors and accidents [4].

In the field of autonomous driving, Model Predictive Control (MPC) [5][6] as a planner has significant advantages. MPC can optimize the path planning in real-time, considering the vehicle's motion and constraints over a future period, to ensure the safety and stability of the vehicle in the complex environment. Through prediction and adjustment, MPC can cope with dynamic traffic conditions and variable road conditions, improving the accuracy and efficiency of path planning. While MPC offers the potential to solve and revolutionize transportation safety, the current landscape reveals a crucial challenge — the struggle to achieve robust performance in real-world scenarios and prevent collisions with stochastic uncertain scenarios [7].

The complexity of real-world driving environments introduces uncertainties in the measurements and estimations of both the ego vehicle and the surrounding vehicles [8]. This inherent uncertainty poses a significant hurdle, particularly in ensuring the safety of AVs. Unlike controlled environments like simulated scenarios or test tracks, real-world scenarios are unpredictable and dynamic. As a result, the AVs must adhere to these changes and make the right decisions at the right time. However, while sensors like LIDARs are quite accurate, their ability to capture and predict the future behaviors and uncertainties of dynamic scenarios is limited, making it difficult for AVs to perceive and anticipate these changes.

To address these problems in the AVs, this project will incorporate the capabilities of Stochastic Model Predictive Control (SMPC). Using SMPC, the controller will now be able to perform the decision-making process by incorporating uncertainties,

enabling the AVs to account for the inherent unpredictability that arises in real-world scenarios. By optimizing trajectories and control inputs over a finite time horizon while considering probabilistic forecasts of future states, SMPC enables the AVs to make robust and adaptive decisions in uncertain conditions.

Recognizing the critical importance of addressing these challenges, this thesis aims to contribute to the advancement of autonomous vehicle technology through the application of a robust control design.

1.2 Aim

This thesis aims to develop and implement an SMPC framework for AVs operating in uncertain environments. The project will construct safety-critical scenarios in the CARLA simulator and develop a controller that can handle uncertainties in vehicle dynamics.

1.3 Limitations

This project will be focused on implementing the safety-critical scenarios for a heavy vehicle. The heavy vehicle from the CARLA simulator will be the ego vehicle, and the traffic vehicles will be chosen as Tesla models from the CARLA. The heavy vehicle chosen from CARLA does not have a truck and trailer combination (semi-truck) and therefore the project will be limited to using a simple truck.

For testing and simulation, the project relies on the CARLA simulator. Although CARLA provides a significant step towards real-world scenarios, it operates within predefined parameters and constraints. This setup allows for the creation of realistic traffic scenarios, with the ego vehicle and the surrounding traffic vehicles modeled as a bicycle model. Through this modeling framework and simulation environment, the project aims to accurately replicate complex driving situations, enabling the evaluation and validation of safety-critical scenarios for heavy vehicles.

In addition, we are constraining our scenarios to highways, under the assumption that both the ego vehicle and the surrounding traffic operate within straight highway roads. This limitation will simplify the simulation setup with a focused exploration of safety-critical scenarios specific to highway driving conditions. However, it also implies that the scenarios will not fully capture the complexity of driving on narrow roads with traffic signals, roundabouts, and intersections which significantly influence the vehicle dynamics and decision-making processes.

The traffic vehicles are considered to have constant velocity and only move on straight roads without lane changing, thus ignoring the unpredictable traffic behavior.

1.4 Main Research Questions

The main research questions to be answered in this project are summarized as follows:

1. How to build a safety-critical scenario for a heavy vehicle in CARLA and make it fit the realistic scenario? This includes determining how to select the appropriate map that reflects actual driving environments, setting realistic vehicle speed profiles, and calibrating vehicle physical model parameters to match those of actual heavy vehicles.
2. How can the challenges associated with the uncertainty in the vehicle's propagation between the vehicle model and the CARLA simulation environment be addressed?
3. How can the integration of stochastic control with the traditional MPC improve the robustness of autonomous heavy vehicles operating in uncertain environments?
4. How does the performance of SMPC compare to traditional MPC in terms of safety and efficiency in high-fidelity simulation environments like CARLA?

1.5 Thesis Overview

Chapter 2 describes the specific problems to be studied and their methodology, including architecture, observer, and controller building. Chapter 3 introduces SMPC and provides the theoretical foundation for this paper. Chapter 4 details the SMPC-based controller and its algorithms. Chapter 5 shows the results of evaluating MPC and SMPC in the CARLA simulation environment. Chapter 6 discusses the research findings and suggests future work. Finally, Chapter 7 summarizes the main conclusions of the study.

2

Methodology and Problem Formulation

This chapter provides a comprehensive overview of the methodology used, focusing on the construction of the problem and the development of a framework for trajectory planning in an autonomous vehicle control environment. Firstly, the structure of the architecture used in the research is described, followed by a detailed explanation of the problem construction for trajectory planning. In addition, this chapter explores the complexity of vehicle modeling, describing how the physical properties and dynamics of the vehicle are represented in the simulation environment. Further, the implementation of motion estimation techniques is explored, with a particular focus on the application of the Extended Kalman Filter (EKF) for accurate state estimation. In addition, the chapter discusses the constraint formulation and the definition of cost functions, which are key components in the construction of the MPC problem. Finally, the construction of the MPC problem itself is outlined, highlighting key considerations and objectives, and clarifying the role of the Decision Master (DM) in optimizing control decisions within the MPC framework. Following the DM, the local controller is introduced, whose task is to refine and execute the DM's high-level commands to ensure that the control commands are real-time and precise, thereby optimizing the overall vehicle dynamic response and performance.

2.1 Architecture

The overall architecture of the project is shown in Figure 2.1. The workflow integrates an EKF with an MPC for enhanced accuracy and robustness. Notably, we simulate sensor inputs rather than using actual CARLA sensors to streamline our process. First, the process model in the architecture uses a kinematic model to predict the future position, velocity, and heading angle of the ego vehicle. This model is based on simplifying assumptions about vehicle dynamics and can efficiently predict the basic motion states of the vehicle.

Next, to obtain accurate state information on the autonomous vehicle, the system combines data from the sensors and performs state estimation via EKF. This step critically improves the accuracy of state prediction, especially in complex dynamic environments.

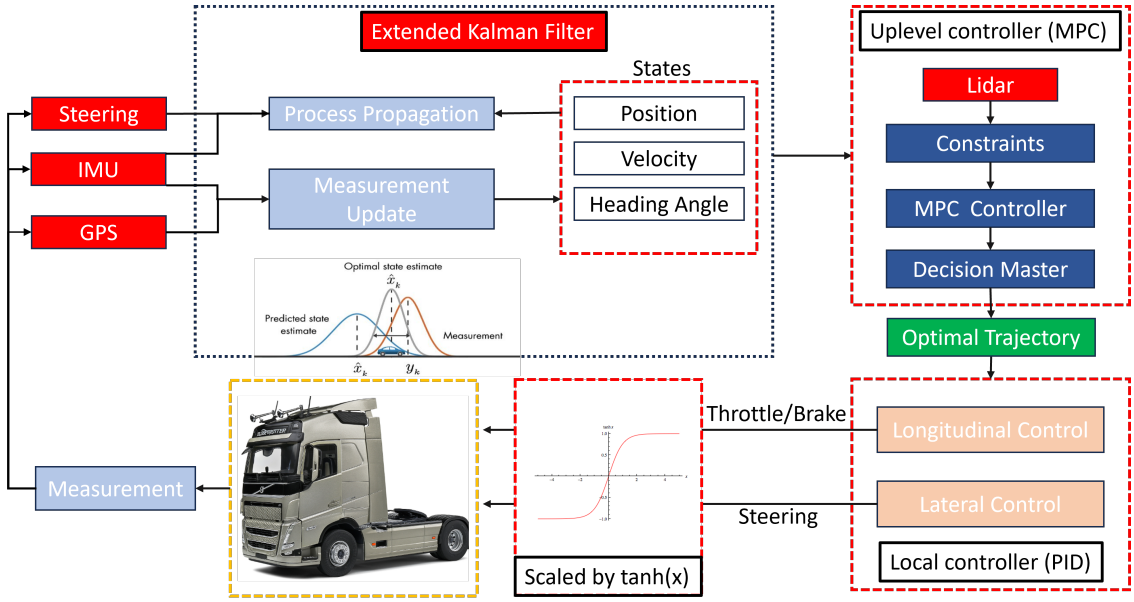


Figure 2.1: Overall control architecture for autonomous truck.

Using information about the surrounding vehicles obtained from LiDAR measurements and data from autonomous vehicles, the system constructs traffic constraints for the MPC. These constraints are an integral part of the MPC strategy and are used to evaluate the optimal states and control input.

After the MPC controller computes the optimal trajectory, this trajectory is passed to the local controller. The local controller is responsible for executing specific actions, including throttle, brake, and steering control. Before executing these commands, the local controller normalizes the control signals using a hyperbolic tangent function (tanh) to adapt to the control requirements in the CARLA simulation environment.

Such an architectural design ensures not only the high efficiency and real-time performance of the control strategy but also the adaptability and safety of the system in complex traffic environments. Through this hierarchical control strategy, high-level decision-making and low-level control execution are effectively and closely integrated to optimize the overall vehicle driving performance.

2.2 Motion Estimation with the Extended Kalman Filter

2.2.1 Estimator Design

Motion estimation [9] is crucial in fields like robotics and AVs, where accurately predicting object movement is essential for navigation. The Extended Kalman Filter (EKF) [10] addresses challenges posed by noise and uncertainties in observed movements by linearizing non-linear dynamics and integrating them into a robust

estimation framework. This integration significantly enhances the precision and reliability of motion estimations. Such improved accuracy is particularly beneficial for non-linear applications, distinguishing it from the standard Kalman Filter (KF).

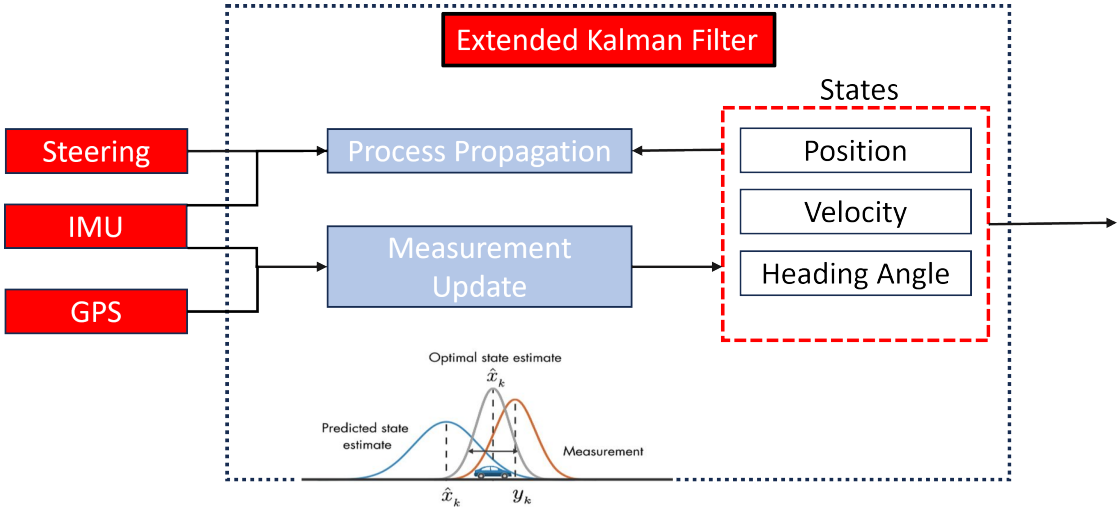


Figure 2.2: Schematic depicting the structure of the Extended Kalman Filter.

The **Process Model** of a system describes how the state of the system evolves based on the kinematics of a vehicle. It is defined as:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \tag{2.1}$$

where $\mathbf{x}$ is the states of the system in consideration and $\mathbf{u}$ comprises control inputs to this system. The dynamics are linearized about a nominal state to facilitate implementation in the EKF:

$$\dot{\mathbf{x}} \approx f(\bar{\mathbf{x}}, \bar{\mathbf{u}}) + \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} (\mathbf{x} - \bar{\mathbf{x}}) + \left.\frac{\partial f}{\partial \mathbf{u}}\right|_{\bar{\mathbf{u}}} (\mathbf{u} - \bar{\mathbf{u}}) \tag{2.2}$$

The discretized version of the system with noise is as shown:

$$\mathbf{x}(k+1) = \mathbf{A}_d\mathbf{x}(k) + \mathbf{B}_d\mathbf{u}(k) + \mathbf{w}(k) \tag{2.3}$$

with $\mathbf{A}_d = (\mathbf{I} + T_s\mathbf{A}_c)$, $\mathbf{B}_d = T_s\mathbf{B}_c$, representing the discretized dynamics and control matrices, and $\mathbf{w}(k)$ the process noise, capturing the uncertainties in system dynamics and control inputs, assumed to be Gaussian.

The **Measurement Model** correlates the actual system state with the observed measurements from the sensors:

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{r}_k \tag{2.4}$$

where $\mathbf{y}(k)$ is the measurement vector, $\mathbf{C}$ is an identity matrix representing a direct measurement of the state variables, and $\mathbf{r}_k$ is the measurement noise, assumed to be Gaussian. This model simplifies the implementation and interpretation of sensor data within the EKF framework.

2.2.2 Kalman Filter Steps

The Kalman Filter algorithm consists of two steps:

- **Prediction**: This step predicts the next state of the system using the process model.

$$\hat{\mathbf{x}}(k+1|k) = \mathbf{A}_d\hat{\mathbf{x}}(k|k) + \mathbf{B}_d\mathbf{u}(k) + \mathbf{w}(k) \tag{2.5}$$

  anticipating the system's evolution based on the latest estimated state and control inputs, and including the effect of process noise. Additionally, the covariance of the prediction is updated as follows:

$$\mathbf{P}(k+1|k) = \mathbf{A}_d\mathbf{P}(k|k)\mathbf{A}_d^T + \mathbf{Q}_w \tag{2.6}$$

  where $\mathbf{Q}_w$ represents the process noise covariance.
- **Update:** In this step, we obtain the measurement of the current state and refine the state estimate.

$$\mathbf{L}(k+1) = \mathbf{P}(k+1|k)\mathbf{C}^T\left(\mathbf{C}\mathbf{P}(k+1|k)\mathbf{C}^T + \mathbf{R}_w\right)^{-1} \tag{2.7}$$

$$\hat{\mathbf{x}}^e(k+1|k+1) = \hat{\mathbf{x}}^e(k+1|k) + \mathbf{L}(k+1)\left(\mathbf{y}^e(k+1) - \mathbf{C}\hat{\mathbf{x}}^e(k+1|k)\right) \tag{2.8}$$

$$\mathbf{P}(k+1|k+1) = (\mathbf{I} - \mathbf{L}(k+1)\mathbf{C})\mathbf{P}(k+1|k) \tag{2.9}$$

  where $\mathbf{L}(k+1)$ is the Kalman Gain, $\mathbf{C}$ is the measurement matrix, $\mathbf{R}_w$ is the measurement noise covariance, $\mathbf{y}^e(k+1)$ represents the actual measurements, and $\mathbf{P}(k+1|k)$ and $\mathbf{P}(k+1|k+1)$ are the covariance matrices of the predicted and updated estimates, respectively. This step adjusts the state estimate based on the differences between actual measurements and predicted measurements, taking into account the uncertainties present in the state estimate and the measurement [11].

2.3 Problem Formulation for Trajectory Planning

Trajectory planning is a central part of autonomous driving. It uses sensor data from the perception system to generate a smooth trajectory for the vehicle to follow, thereby minimizing the risk of collisions. Different numerical techniques, such as graph-search methods, incremental tree-based approaches, and optimization techniques, are outlined in [12]. These methods however are suboptimal or require huge computation for optimal solutions. Autonomous driving presents a complex control problem involving real-time decision-making amidst uncertain and dynamic environments [13] [14]. MPC has shown promising results that could handle the above-mentioned issues [15].

MPC has a rich history dating back several decades, with early indications in academic literature and subsequent explosive growth driven by its adoption in various industries. Its remarkable success in handling complex systems with hard control constraints is attributed to its conceptual simplicity and efficient handling of multiple inputs and outputs [3].

MPC utilizes predictive models to forecast future process outputs, computes a control sequence to optimize a performance index, and employs a receding horizon strategy, applying only the first control signal of the sequence at each step [16]. MPC aims to minimize a cost function designed according to the specific objectives throughout the prediction horizon while satisfying the constraint sets. For a trajectory planning controller, the cost function can aim to minimize the control input and the error in reference trajectory tracking. The constraints can include collision avoidance, vehicle velocity limits, or road boundaries.

In this project, the analysis environment is set as a straight three-lane highway. As shown in Figure 2.3, the red ego vehicle is surrounded by multiple vehicles, and it tries to change lanes or follow the vehicles while maintaining the reference speed to reach the end point safely. This process requires the ego vehicle to precisely control its trajectory to avoid potential collisions with surrounding vehicles.

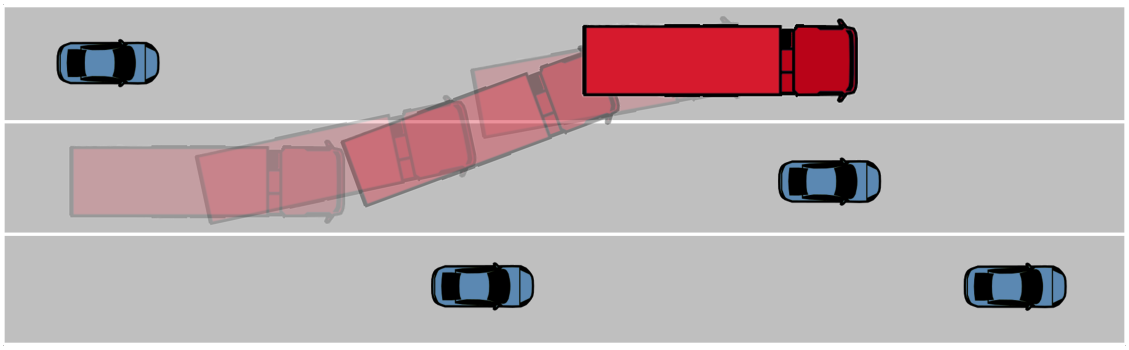


Figure 2.3: Schematic simulation of lane changing and trailing by an ego vehicle on a three-lane highway.

Figure 2.4 illustrates the description of the coordinate system in the problem setup, which provides the basis for subsequent modeling and analysis. In this coordinate system, the red autonomous vehicle is attempting to make a lane change. The red arrow in the figure indicates the vehicle's heading angle, $-\psi$, which is the angle with respect to the x axis. In the y direction, the red dashed line in the center lane indicates the vehicle's reference line, p_y^{ref}, and in the subsequent simulations of MPC, all y direction references will be to the center line of the individual lanes.

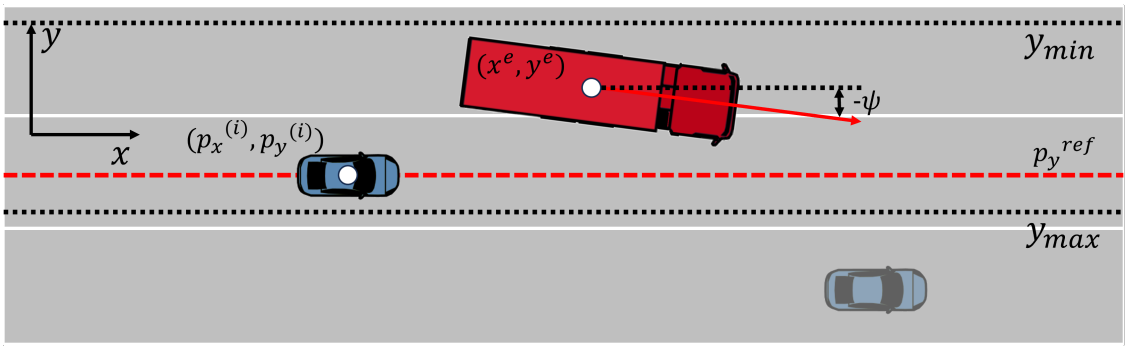


Figure 2.4: Schematic of the coordinate system of the vehicle during a lane change. The red one is the ego vehicle, the blue one is the traffic vehicle, and the red dotted line p_y^{ref} is the reference coordinate in the y direction of the ego vehicle during the lane change.

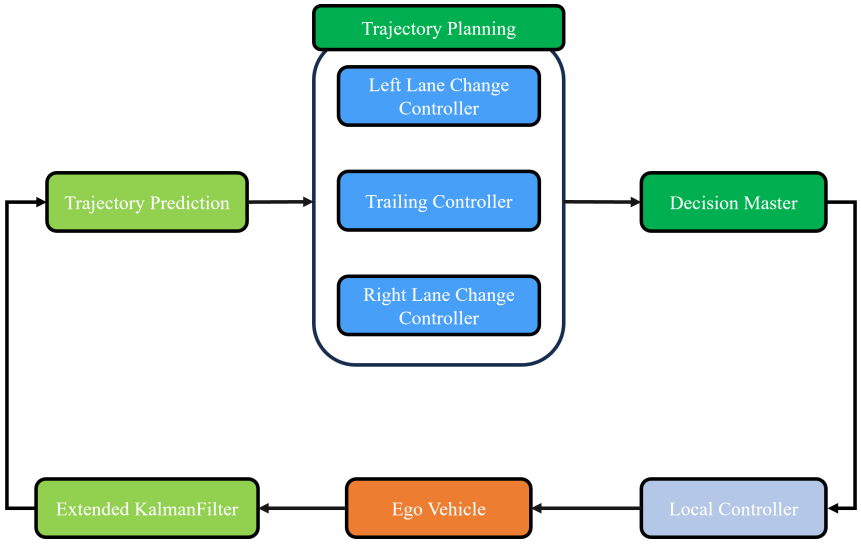


Figure 2.5: The ego vehicle uses three separate MPCs each for trailing, left, and right lane change respectively. A Decision Master then evaluates the cost associated with the three controllers and chooses the trajectory output from the optimal controller.

In this project, the core of the trajectory planning is based on MPC to find the optimal travel path for the ego vehicle, and its detailed architecture is shown in Figure 2.5. First, the current state of the ego vehicle is estimated by the EKF, and the kinematic model is combined to predict the future N steps of the ego vehicle trajectory.

The trajectory planning module consists of three independent MPC controllers, each responsible for calculating the optimal states and control inputs for different maneuvers: maintaining the current lane (no change, nc), making a left lane change (lc), and making a right lane change (rc). These controllers also evaluate the cost associated with each operation and provide the optimized driving trajectory accordingly. The Decision Master (DM) determines which controller output to use and

passes this decision to the local controller. The algorithm for the Decision Master is adapted from our supervisor Erik Börve's GitHub repository [17] and is discussed in more detail in Section 2.4.

Once the controller and optimal trajectory are selected, the local controller precisely controls the ego vehicle's movements, including throttle, brakes, and steering, based on this trajectory. The control commands of the local controller are further processed by functions to ensure compatibility with the CARLA simulation environment.

This architecture ensures accurate and real-time trajectory planning, enabling efficient and safe driving decisions in complex traffic environments. By employing careful module partitioning and specialized control strategies, the project effectively manages a variety of driving situations and optimizes the behavioral responses of autonomous vehicles.

2.3.1 Vehicle Modelling

A kinematic bicycle model is used to model the ego vehicle. A schematic of the bicycle model is shown in Figure 2.6:

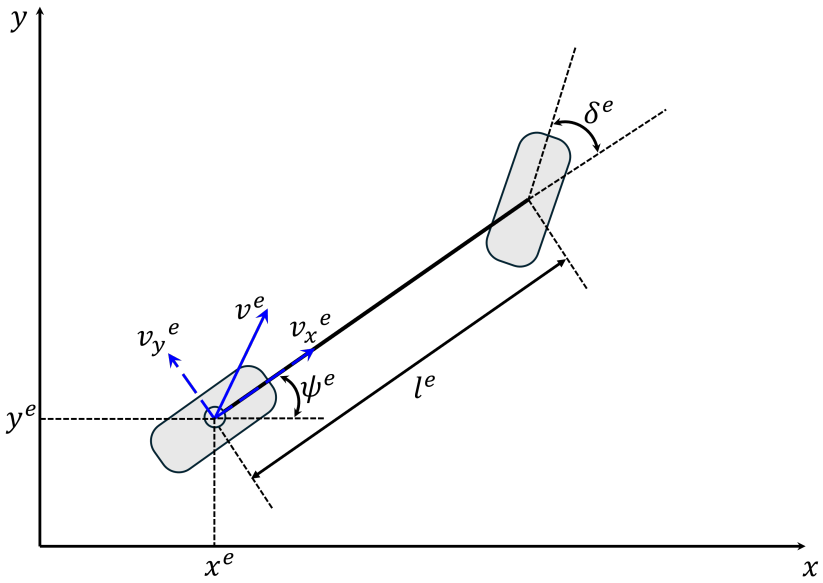


Figure 2.6: Vehicle model for the ego vehicle, l^e is the wheelbase, x^e and y^e are the position represented from the middle of the rear axle of the vehicle, ψ^e is the yaw of the vehicle with respect to the simulation environment, δ^e is the steering angle.

The state representation of the vehicle when the reference is at the rear axle is given

as,

$$\dot{\mathbf{x}}^e = \begin{bmatrix} \dot{x}^e \\ \dot{y}^e \\ \dot{v_x}^e \\ \dot{\psi}^e \end{bmatrix} = \begin{bmatrix} v_x^e \\ v_x^e \tan(\psi^e) \\ a^e \cos(\psi^e) \\ v_x^e \frac{\tan(\delta^e)}{L \cos(\psi^e)} \end{bmatrix}, \quad \mathbf{u}^e = \begin{bmatrix} \delta^e \\ a^e \end{bmatrix} \tag{2.10}$$

where x^e and y^e are the coordinates represented from the middle of the rear axle of the vehicle, v_x^e is the longitudinal velocity of the vehicle, a^e is the longitudinal acceleration of the vehicle, ψ^e is the yaw of the vehicle, δ^e is the steering angle, and l^e is the length of the vehicle. The vehicle states are controlled by the steering angle δ^e and the longitudinal acceleration a^e.

This model simplifies the kinematics by assuming the motion of the vehicle is controlled directly at the rear axle, which is particularly useful for analyses where this simplification provides sufficient accuracy (e.g., low-speed maneuvers or specific types of vehicles like some rear-wheel steering models).

The kinematic model when the reference point is at the center of gravity (COG) is shown in Figure 2.7:

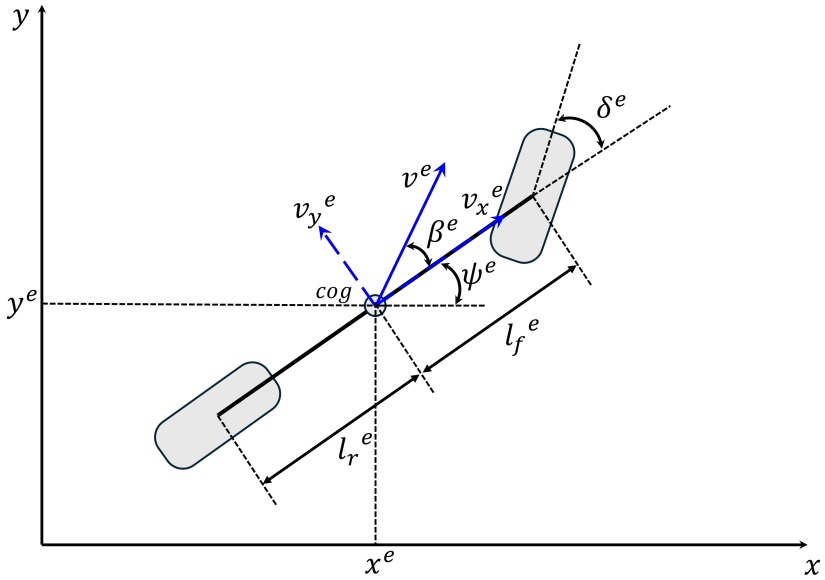


Figure 2.7: Vehicle model for the ego vehicle with a reference point at COG, l_f^e is the front wheelbase length, l_r^e is the rear wheelbase length, β^e is the slipping angle, x^e and y^e are the position represented from the middle of the rear axle of the vehicle, ψ^e is the yaw of the vehicle with respect to the simulation environment, δ^e is the steering angle.

For more accurate dynamics, particularly at higher velocities, the reference point is shifted to the COG, as shown in Figure 2.7. This adjustment is based on the observations from the simulations done in CARLA, which indicate improved modeling

accuracy at higher speeds when using the COG reference point.

$$\beta^e = arctan(\frac{l_r^e tan(\delta^e)}{l_r^e + l_f^e}) \tag{2.11}$$

$$\dot{\mathbf{x}}^e = \begin{bmatrix} \dot{x}^e \\ \dot{y}^e \\ \dot{v}^e \\ \dot{\psi}^e \end{bmatrix} = \begin{bmatrix} v^e \cos(\beta^e + \psi^e) \\ v^e \sin(\beta^e + \psi^e) \\ a^e \cos(\psi^e) \\ v^e \frac{\tan(\delta^e)\cos(\beta^e)}{l_r^e + l_f^e} \end{bmatrix}, \quad \mathbf{u}^e = \begin{bmatrix} \delta^e \\ a^e \end{bmatrix} \tag{2.12}$$

The COG-based model incorporates a slip angle β^e, reflecting the deviation between the vehicle's velocity vector at the COG and its longitudinal axis. This slip angle is influenced primarily by the steering angle δ^e and the dimensions of the wheelbases (l_r^e and l_f^e).

The main difference is the level of detail included in both models. The rear axle model is simpler and is usually sufficient for low-speed operation. The COG model, on the other hand, includes more detail, such as the slip angle β^e, which allows for a more accurate representation of vehicle dynamics at higher speeds. It's important to note that the inclusion of the slip angle is not limited to the center of gravity model; it can also be integrated into the rear axle model to enhance accuracy. Additionally, choosing the COG as the center of rotation has advantages during cornering, as it more accurately reflects vehicle behavior, especially at high speeds. This is particularly important for modeling the dynamics of long vehicles, such as trucks.

The COG-based model will be utilized to develop the MPC framework within the CARLA simulation environment.

2.3.2 Linearization and Discretization

Equation (2.12) undergoes linearization and discretization using the forward Euler method. The continuous-time vehicle dynamics are defined as:

$$\dot{\mathbf{x}}^e = \begin{bmatrix} \dot{x}^e \\ \dot{y}^e \\ \dot{v}^e \\ \dot{\psi}^e \end{bmatrix} = \begin{bmatrix} v^e \cos(\beta^e + \psi^e) \\ v^e \sin(\beta^e + \psi^e) \\ a^e \cos(\psi^e) \\ v^e \frac{\tan(\delta^e)\cos(\beta^e)}{l_r^e + l_f^e} \end{bmatrix} = f(\mathbf{x}^e, \mathbf{u}^e), \tag{2.13}$$

where $f(\mathbf{x}^e, \mathbf{u}^e)$ represents the state dynamics. Linearizing about a nominal operating point, we approximate the state derivative as:

$$\dot{\mathbf{x}}^e \approx f(\bar{\mathbf{x}}^e, \bar{\mathbf{u}}^e) + \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\bar{\mathbf{x}}^e, \bar{\mathbf{u}}^e} (\mathbf{x}^e - \bar{\mathbf{x}}^e) + \left.\frac{\partial f}{\partial \mathbf{u}}\right|_{\bar{\mathbf{x}}^e, \bar{\mathbf{u}}^e} (\mathbf{u}^e - \bar{\mathbf{u}}^e). \tag{2.14}$$

Subsequently, the system is represented in a discretized form as:

$$\frac{\mathbf{x}^e(k+1) - \mathbf{x}^e(k)}{T_s} = \mathbf{A}_c \mathbf{x}^e(k) + \mathbf{B}_c \mathbf{u}^e(k) + \mathbf{g}_c \tag{2.15}$$

where:

- $\bar{\mathbf{x}}^e$ and $\bar{\mathbf{u}}^e$ are the nominal state and input vectors around which the system is linearized.
- $\frac{\partial f}{\partial \mathbf{x}}\Big|_{\bar{\mathbf{x}}^e, \bar{\mathbf{u}}^e} = \mathbf{A}_c$ is the Jacobian matrix of partial derivatives of f with respect to the state vector, evaluated at the nominal point. This matrix captures how small changes in the state affect the rate of change of the state.
- $\frac{\partial f}{\partial \mathbf{u}}\Big|_{\bar{\mathbf{x}}^e, \bar{\mathbf{u}}^e} = \mathbf{B}_c$ is the Jacobian matrix of partial derivatives of f with respect to the input vector, evaluated at the nominal point. This matrix captures how small changes in the input affect the rate of change of the state.
- $f(\bar{\mathbf{x}}^e, \bar{\mathbf{u}}^e) = \mathbf{g}_c$ represents the state dynamics evaluated at the nominal point, acting as a constant offset.

The discrete-time state-space model is then formulated as follows:

$$\mathbf{x}^e(k+1) = \mathbf{A}_d \mathbf{x}^e(k) + \mathbf{B}_d \mathbf{u}^e(k) + \mathbf{g}_d, \tag{2.16}$$

where $\mathbf{A}_d = (\mathbf{I} + T_s \mathbf{A}_c)$, $\mathbf{B}_d = T_s \mathbf{B}_c$, and $\mathbf{g}_d = T_s \mathbf{g}_c$.

2.3.3 Constraint Formulation

The physical limitations of the vehicle are expressed as box constraints to ensure safe operation:

$$\mathbf{x}^e_{\min} \leq \hat{\mathbf{x}}^e(k) \leq \mathbf{x}^e_{\max} \tag{2.17}$$
$$\mathbf{u}^e_{\min} \leq \mathbf{u}^e(k) \leq \mathbf{u}^e_{\max} \tag{2.18}$$

To obtain a feasible, smooth, and collision-free trajectory, constraints are formulated on the states and the inputs. Following the rules of the road, the vehicle is restricted to driving only in the permitted area on the roads. Moreover, the constraints on the maximum allowed steering angle and acceleration input should be taken into account.

Moreover, the position of surrounding traffic vehicles is estimated using a Constant Velocity (CV) model, which allows for rapid computation by assuming that the vehicle's speed remains constant throughout the prediction horizon (T_{pred}). This is given as,

$$(\hat{x}^{(i)}(k+1), \hat{y}^{(i)}(k+1)) = \Pi_{\mathrm{CV}}(\hat{x}^{(i)}(k), \hat{y}^{(i)}(k)), \quad \forall i \in \mathbb{V}(k) \tag{2.19}$$
$$\hat{\mathbf{X}}^{(i)}_j = \left\{\hat{\mathbf{x}}^{(i)}_j(0), \hat{\mathbf{x}}^{(i)}_j(1), \ldots, \hat{\mathbf{x}}^{(i)}_j(N)\right\}, \quad j \in \{\mathrm{nc}, \mathrm{lc}, \mathrm{rc}\} \tag{2.20}$$

where, Π_{CV} represents the position prediction function under the constant velocity model, $\mathbb{V}(k)$ represents all vehicles around the ego vehicle at time step k, $\hat{\mathbf{X}}^{(i)}_j$ represents the surrounding vehicle i's states throughout the total time horizon. Specifically, the function takes the vehicle's estimated positions at the current time step k, $\hat{x}^{(i)}(k)$ and $\hat{y}^{(i)}(k)$, and predicts their positions for the next time step $k+1$. On a straight and endless highway scenario, since the road is straight, the y coordinates of vehicles are assumed to be constant, while the x coordinates are updated as follows,

$$\hat{x}^{(i)}(k+1) = \hat{x}^{(i)}(k) + \dot{x}^{(i)}(0)T_s \tag{2.21}$$

Where, $\dot{x}^{(i)}(0)$ is the initial velocity of the vehicle at time step $k = 0$, and T_s is the length of the prediction time step. This model assumes that all vehicles move in a straight line at a constant speed.

These estimated positions of the traffic vehicles are used to formulate the constraints for the three MPCs {nc, lc, rc}. To avoid collisions with the traffic vehicles, the constraints are formulated in two different ways, one for the trailing scenario {nc} and the other one for the lane-changing scenario {lc, rc}.

For the trailing scenario {nc}, to keep a safe longitudinal distance with the leading vehicle, the constraint is formulated as,

$$c_{\mathbf{nc}}\left(\hat{x}^{e}, \hat{x}^{(l)}, \hat{v}_x^{(l)}\right) = d + L^{\text{truck}} + T\hat{v}_x^{(l)} + \hat{x}^{e} - \hat{x}^{(l)} < 0 \tag{2.22}$$

Subscript e is to denote the ego vehicle and l is to denote the leading traffic vehicle. $\hat{x}^e$ is the estimated x position of the ego vehicle and $\hat{x}^{(l)}$ is the estimated x position of the leading traffic vehicle from the constant velocity model. The constraint (2.22) ensures that the sum of the safe trailing distance d, the length of the vehicle L^{truck}, the reaction time T multiplied by the estimated velocity of the leading vehicle $\hat{v}_x^{(l)}$, and the difference between the estimated positions of ego vehicle and the lead vehicle remains negative. In other words, it ensures that the ego vehicle does not get too close to the lead vehicle, maintaining a safe trailing distance. If the constraint is violated (i.e., if the sum becomes positive), it indicates that the ego vehicle is too close to the lead vehicle, and adjustments need to be made to ensure safety. This scenario is illustrated in the Figure 2.8

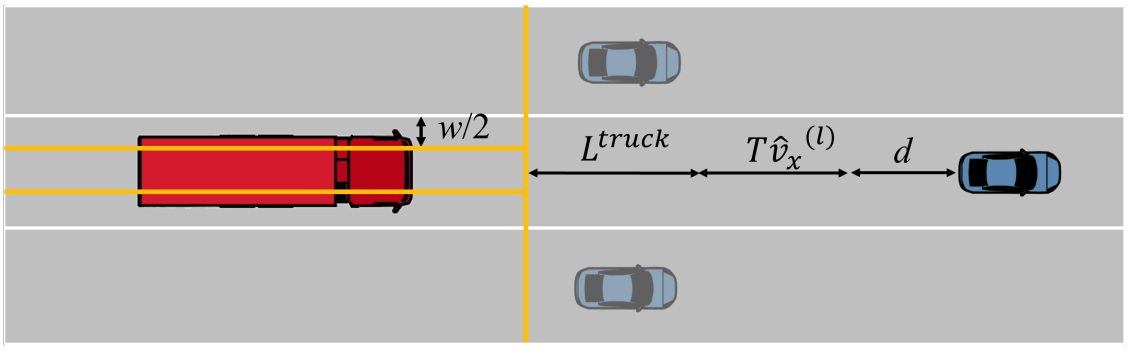


Figure 2.8: Trailing constraint where the ego vehicle is depicted in the red-colored truck and the leading vehicle is depicted in the blue-colored car. $\hat{v}_x^{(l)}$ is the estimated velocity of the leading vehicle, T is the minimum required headway in time during braking, and d represents the desired minimum longitudinal distance to the vehicle in front in the same lane. In addition, the lateral restriction for ego vehicles states that their COG plus one-half of their maximum lateral incline ($w/2$) must not exceed the edge of the lane.

For the lane-changing MPCs {lc, rc}, the traffic vehicles are constrained both laterally and longitudinally. The constraints are formulated as,

$$b^{(i)}\left(\hat{x}^e, y^e, \hat{x}^{(i)}, \hat{y}^{(i)}\right) = \frac{\varepsilon_0^{(i)}\left(\hat{y}^{(i)}\right)}{2}\left[\tanh\left(\hat{x}^e - \hat{x}^{(i)} + \varepsilon_1^{(i)}\right) + \tanh\left(\hat{x}^{(i)} - \hat{x}^e + \varepsilon_2^{(i)}\right)\right] + \varepsilon_3^{(i)}\left(\hat{y}^e, \hat{y}^{(i)}\right) \tag{2.23}$$

$$c_j\left(\hat{x}^e, \hat{y}^e, \hat{x}^{(i)}, \hat{y}^{(i)}\right) = F_j^{(i)}\left(b^{(i)}(\cdot) - \hat{y}^e + w\right) < 0 \tag{2.24}$$

In (2.23), the function $b^{(i)}$ encapsulates the lateral and longitudinal constraints imposed on surrounding vehicles i in relation to the ego vehicle's position $(\hat{x}^e, \hat{y}^e)$ and the predicted position of the vehicle i $(\hat{x}^{(i)}, \hat{y}^{(i)})$. Specifically, the parameter $\varepsilon_0^{(i)}$ influences lateral scaling and depends on the planned y-position of the ego vehicle, which includes its maximum lateral span and safety margin, as well as the predicted y-position of the surrounding vehicle i, considering its width. This ensures that the lateral constraints account for the spatial requirements of both the ego vehicle and the surrounding vehicles.

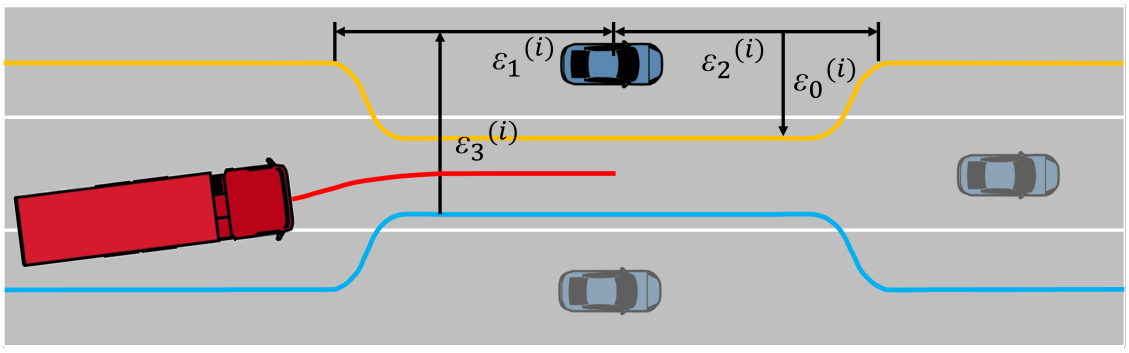
In the longitudinal direction, the constraints are scaled by parameters $\varepsilon_1^{(i)}$ and $\varepsilon_2^{(i)}$, which incorporate a defined safety margin, the front and rear length of the vehicle i. These parameters adjust the constraints to account for the longitudinal spacing between vehicles, ensuring safe distances are maintained during lane changes.

Additionally, the parameter $\varepsilon_3^{(i)}$ adjusts the lateral constraint based on the predicted y position of the vehicle i and the planned y position of the ego vehicle. This ensures that the lateral constraints are dynamically adapted based on the relative positions of the vehicles.

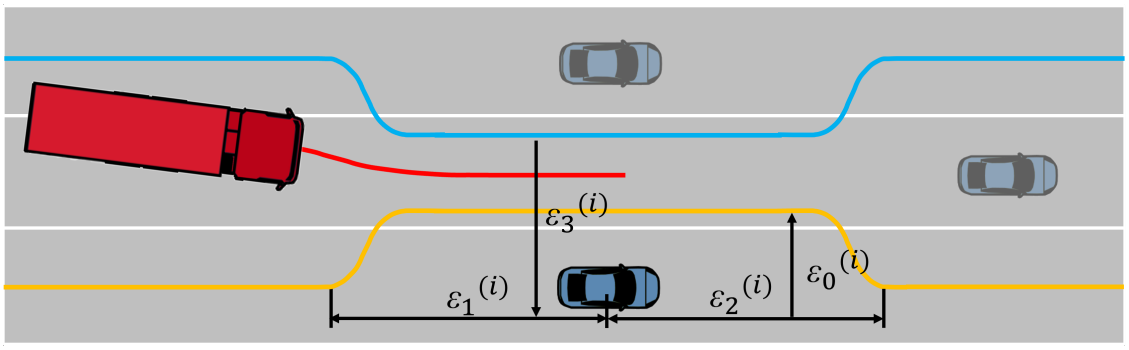
Equation (2.24) represents the final constraint function. The function $F_j^{(i)}$ mirrors the constraints based on the lane and vehicle being observed, with values of either -1 or 1. This parameter accounts for which lane the ego vehicle is located in and which vehicle is being considered for the lane change maneuver. It also considers the specific controller being optimized, with reference to the index j representing left or right lane changes {lc, rc}. The constraint equation (2.24) states that the result of the mirrored constraint function must be less than zero for the constraint to be satisfied. This ensures that the constraint is enforced to prevent collisions during the lane change maneuver. A visualization of the constraints for both left and right lane changes is displayed in Figure 2.9.

2.3.4 Ensuring Feasibility with Slack Variables

In certain scenarios, the solution to the MPC can become infeasible due to factors like tight traffic conditions. To address this potential in-feasibility, slack variables, λ_j^e are introduced in every controller $j = \{\text{nc, lc, rc}\}$. These slack variables allow the ego vehicle to intrude into the space of surrounding vehicles, acting as a



(a) Left lane change scenario constraint.



(b) Right lane change scenario constraint.

Figure 2.9: The ego vehicle is depicted in the red-colored truck and the surrounding vehicle is depicted in the blue-colored car. The constraint on the vehicle i, depicted as yellow, is defined by $\varepsilon_0^{(i)}$, which is used to scale the constraint laterally, the two continuous tanh functions scaled by $\varepsilon_1^{(i)}$ and $\varepsilon_2^{(i)}$ respectively and $\varepsilon_3^{(i)}$ which is used to shift the constraint to correct the lateral position on the highway.

mechanism to accommodate minor discrepancies between predictions and true trajectories. Slack variables are added into the objective function with a high penalty to discourage the relaxation of constraints too much. However, while the enforcement of strict collision avoidance constraints that lead to in-feasibility is mitigated, it is important to note that collision avoidance is no longer guaranteed. The high penalty significantly reduces the likelihood of collisions but does not eliminate the possibility entirely.

As discussed, for the lane-keeping controller, $j = \text{nc}$, longitudinal distance is controlled to keep a safe distance from the vehicle in front to avoid a collision. The minimum allowed longitudinal distance given by (2.22) is reformulated as,

$$c_{\text{nc}}\left(\hat{x}^e, \hat{x}^{(l)}, \hat{v}_x^{(l)}, \lambda_{\text{nc}}^e\right) = d + L^{\text{truck}} + T\hat{v}_x^{(l)} + \hat{x}^e - \hat{x}^{(l)} + \lambda_{\text{nc}}^e < 0 \tag{2.25}$$

Similarly, for lane changing scenario $j = \{\text{lc}, \text{rc}\}$, (2.24)is reformulated as,

$$c_j\left(\hat{x}^e, \hat{y}^e, \hat{x}^{(i)}, \hat{y}^{(i)}, \lambda_j^{\text{e}}\right) = F_j^{(i)}\left(b^{(i)}(\cdot) - \hat{y}^e + w\right) + \lambda_j^e < 0 \tag{2.26}$$

Finally, the constraints outlined in (2.24) and (2.26), along with the physical limi-

tations on the states and control input of the ego vehicle given in (2.18), collectively form the general constraints equation for each controller j as,

$$h_j^{\mathrm{e}}\left(\hat{\mathbf{x}}_j^e(k), \hat{\mathbf{u}}_j^e(k), \lambda_j^{\mathrm{e}}(k), \hat{\mathbf{x}}_j(k), \hat{\mathbf{u}}_j(k)\right) \leq 0 \tag{2.27}$$

where $\hat{\mathbf{x}}_j^e(k), \hat{\mathbf{u}}_j^e(k)$ are the ego vehicle state and control inputs, $\hat{\mathbf{x}}_j(k), \hat{\mathbf{u}}_j(k)$ are the estimated state and control inputs of all surrounding vehicles.

2.3.5 Optimal Control Problem Formulation

The MPC's objective is to track a reference lateral position and a longitudinal velocity while minimizing the control input. The MPC also aims to minimize the change in the control input to ensure a smoother driving experience. The cost function J_j^e for each of the MPCs, $j = \{\mathrm{nc}, \mathrm{lc}, \mathrm{rc}\}$, with slack variables are formulated as follows,

$$J_j^e\left(\hat{\mathbf{X}}_j^e, \mathbf{U}_j^e, \mathbf{\Lambda}_j^e\right) = \sum_{k=0}^{N-1} (\left\|\hat{\mathbf{x}}_j^e(k) - \mathbf{x}_{\mathrm{ref},j}^e(k)\right\|_{\mathbf{Q}_p}^2 + \left\|\lambda_j^e(k)\right\|_{\mathbf{Q}_s}^2) \tag{2.28a}$$

$$+ \sum_{k=0}^{N-1} \left\|\mathbf{u}_j^e(k)\right\|_{\mathbf{R}}^2 \tag{2.28b}$$

$$+ \left\|\hat{\mathbf{x}}_j^e(N) - \mathbf{x}_{\mathrm{ref},j}^e(N)\right\|_{\mathbf{Q}_f} + \left\|\lambda_j^e(N)\right\|_{\mathbf{Q}_s}^2 \tag{2.28c}$$

$$+ \sum_{k=0}^{N-2} \left\|\mathbf{u}_j^e(k+1) - \mathbf{u}_j^e(k)\right\|_{\mathbf{R}_d}^2 \tag{2.28d}$$

Where, the matrices $\mathbf{Q}_p$, $\mathbf{R}$, and $\mathbf{R}_d$ are positive semi-definite and their higher values signify a greater emphasis on minimizing deviations from desired states, control effort, and changes in control inputs, respectively. $\mathbf{Q}_s$ is the weighing matrix for the slack variables. The terminal cost matrix $\mathbf{Q}_f$ is the infinite time-domain linear quadratic regulator (LQR) determined by solving the discrete-time Riccati differential equations (discussed later in Section 4.1). Larger values in $\mathbf{Q}_f$ imply greater penalties for larger deviations from the reference final state. Ensuring that the final state closely matches the reference state can enhance the stability and performance of the controlled system. It helps prevent large deviations at the end of the prediction horizon, which could lead to instability or undesirable behavior [18]. $\mathbf{x}_{\mathrm{ref},j}^e = [0, y_{\mathrm{ref},j}^e, v_{\mathrm{ref}}^e, 0]^T$ is the state reference for each controller, $\mathbf{u}_{\mathrm{ref},j}^e = [0, 0]^T$ is the input reference for each controller. The ego vehicle states over the horizon N are given as:

$$\hat{\mathbf{X}}_j^e = \left\{\hat{\mathbf{x}}_j^e(0), \hat{\mathbf{x}}_j^e(1), \ldots, \hat{\mathbf{x}}_j^e(N)\right\}, \quad j \in \{\mathrm{nc}, \mathrm{lc}, \mathrm{rc}\} \tag{2.29}$$

The control input $\mathbf{U}_j^e$ and slack variable $\mathbf{\Lambda}_j^e(k)$ are represented similarly as Equation 2.29.

The trajectory optimization problem for motion planning with collision avoidance is formulated in the MPC form as follows:

$$\underset{\hat{\mathbf{X}}_j^e, \mathbf{U}_j^e, \mathbf{\Lambda}_j^e}{\text{minimize}} \quad J_j^e \left(\hat{\mathbf{X}}_j^e, \mathbf{U}_j^e, \mathbf{\Lambda}_j^e \right) \tag{2.30a}$$

$$\text{s.t.} \quad \hat{\mathbf{x}}_j^e(k+1) = \mathbf{A}_\mathrm{d} \hat{\mathbf{x}}_j^e(k) + \mathbf{B}_\mathrm{d} \mathbf{u}_j^e(k) + \mathbf{g}_\mathrm{d}, \tag{2.30b}$$

$$(\hat{x}^{(i)}(k+1), \hat{y}^{(i)}(k+1)) = \Pi_\mathrm{CV}(\hat{x}^{(i)}(t), \hat{y}^{(i)}(k)), \quad \forall i \in \mathbb{V}(k), \tag{2.30c}$$

$$h_j^\mathrm{e} \left(\hat{\mathbf{x}}_j^e(k), \hat{\mathbf{u}}_j^e(k), \lambda_j^\mathrm{e}(k), \hat{\mathbf{x}}_j(k), \hat{\mathbf{u}}_j(k) \right) \leq 0, \tag{2.30d}$$

$$\lambda_j^e(k) \leq 0 \tag{2.30e}$$

$$\hat{\mathbf{x}}_j^e(0) = \hat{\mathbf{x}}^e(t) \tag{2.30f}$$

The objective of MPC is to minimize the cost function J_j^e, given by (2.28), with respect to the states, control inputs of the ego vehicle, and the slack variables. Constraint equation (2.30d) is the general form of the constraints mentioned in (2.27). Constraint (2.30b) is the state dynamics constraint, and the initial condition constraint (2.30f) specifies that the initial state of the system equals the current EKF state estimations. The resulting optimal control sequence from each of the three controllers is then used to calculate the optimal trajectory for motion planning by the DM.

After the control input for each of the three MPC controllers, $j \in \{\mathrm{nc}, \mathrm{lc}, \mathrm{rc}\}$, is calculated as in 2.30, the decision function g selects the optimal control sequence based on the following criterion:

$$g_j \left(\hat{\mathbf{X}}_j^e, \mathbf{U}_j^e, \mathbf{\Lambda}_j^e \right) = q_e J_j^e \left(\hat{\mathbf{X}}_j^e, \mathbf{U}_j^e, \mathbf{\Lambda}_j^e \right) + q_s J_j^s \tag{2.31}$$

$$\underset{\hat{\mathbf{X}}_j^e, \mathbf{U}_j^e, \mathbf{\Lambda}_j^e}{\text{argmin}} \quad g_j \left(\hat{\mathbf{X}}^e, \mathbf{U}^e, \mathbf{\Lambda}^e \right), \quad j \in \{\mathrm{nc}, \mathrm{lc}, \mathrm{rc}\} \tag{2.32}$$

where, q_e, and q_s are the positive scaling parameters that determine the significance of each cost function. J_j^e is the current cost of the j^{th} MPC calculated as in 2.28 and J_j^s is the cost of switching the controller j. This cost is introduced to avoid the frequent decision changes between trailing and lane change, and it is given as,

$$J_j^s = \sum_{i=1}^{m} \mathrm{I}(H(i) - j) \tag{2.33}$$

where $H(i)$ is the history of the previous choices made, j is the current controller, m is the number of previous decisions considered and I is an indicator function. The DM is adapted from and explained in detail in [19].

2.4 Decision Tree Framework

In the control system design shown in Figure 2.10, the decision tree framework is used to assign appropriate controllers in different lanes, aiming to optimize the trajectory and reduce the frequency of controller usage, thus improving the efficiency

of the whole system.

When the ego vehicle is in the middle lane, it has the potential to perform the three operations namely, trailing, left change, and right change, providing maximum flexibility to cope with the surrounding traffic conditions. Therefore, for the ego vehicle in the middle lane, all three controllers will be active.

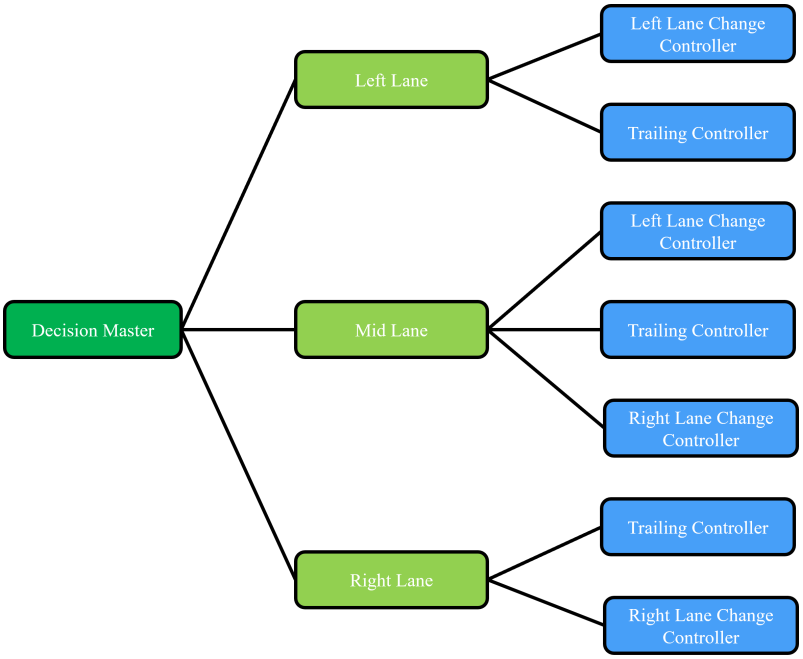
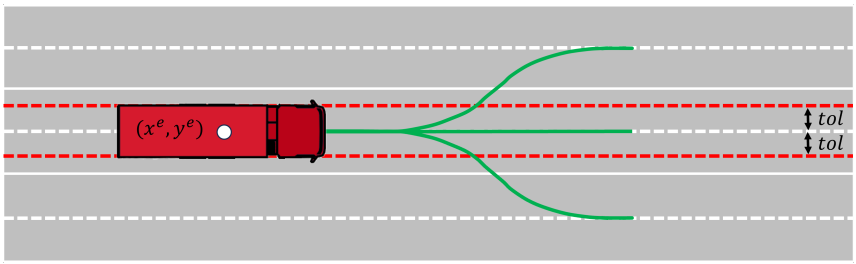


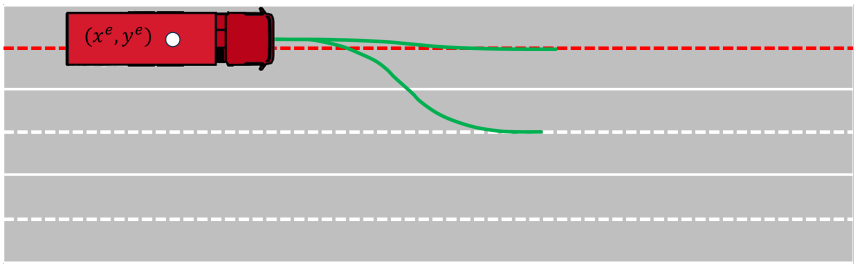
Figure 2.10: A framework of decision tree for controlling vehicles in different lane environments. The DM is located at the root node of the decision tree and decides to assign control to the relevant controller in the left lane, center lane, or right lane based on specific traffic and environmental conditions.

When the ego vehicle is in the left lane, it can either stay in the same lane or change to the right lane. To facilitate this, the system is configured with a left change controller (which switches between the middle and left lanes) and a trailing controller. The reference path for the left change controller is updated to the center line of the middle lane, and the trailing controller's reference path is set to the center line of the left lane. This configuration allows the ego vehicle to switch smoothly between the left and middle lanes without frequently changing controllers. Ignoring slack variables, it should be noted that, due to their respective constraints, the right lane change controller cannot be used in the left lane due to the guaranteed in-feasibility. If the slack variables were defined, the solutions would however be feasible, but with very high slack cost. To avoid calculating this known high cost, the DM is designed to deactivate the right lane change controller in the left lane.

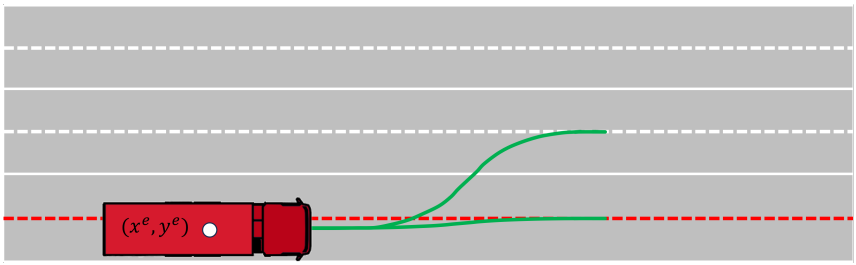
For the ego vehicle in the right lane, a similar logic is applied. The system uses a right change controller (which switches between the middle and right lanes) and a trailing controller. Here, the left lane change controller is deactivated.



(a) Middle lane condition.



(b) Left lane condition.



(c) Right lane condition.

Figure 2.11: Schematic diagram showing the conditions when the reference for each controller is updated. The ego vehicle is depicted as a red truck. The condition boundaries are illustrated by red dashed lines.

As shown in Figure 2.11a, if the ego vehicle is in the center lane and its position is within the tolerance of the center lane, the reference line of the trailing controller is changed to the center line of the center lane, the reference line of the left change controller is changed to the center line of the left lane, and the reference line of the right change controller is changed to the center line of the right lane. As shown in Figure 2.11b, if the ego vehicle is located to the left of the center line of the left lane, change the reference line of the trailing to the center line of the left lane, and change the reference line of the left change to the center line of the middle lane. As shown in Figure 2.11c, if the ego vehicle is located to the left of the center line of the right lane, change the reference line of the following vehicle to the center line of the right lane and change the reference line of the right lane change to the center line of the middle lane.

2.5 Local Controller

Following the DM selection of the optimal control strategy from the left lane change controller, trailing controller, and right lane change controller, an optimal trajectory $\mathbf{X}^{e*}$ is computed. This trajectory is subsequently tracked by the local controller, illustrated in Figure 2.12. The local controller integrates both longitudinal and lateral controls. The longitudinal control is tasked with speed regulation, whereas lateral control manages the vehicle's lateral displacement.

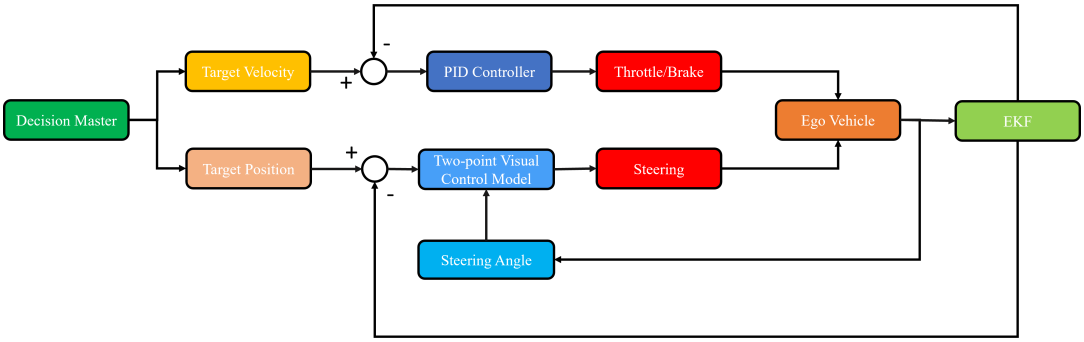


Figure 2.12: Block diagram of the local controller, showing longitudinal and lateral control modules for tracking the DM's optimal trajectory.

2.5.1 Longitudinal Control Using a PID Controller

Effective longitudinal control is crucial for the safety and comfort of autonomous driving systems. This control is primarily achieved using a Proportional Integral Derivative (PID) controller [20]. The PID controller adjusts the vehicle's speed to closely match a set target speed, effectively minimizing speed discrepancies.

The operational principle of the PID controller is expressed as:

$$a^e(k) = K_p \cdot e_v(k) + K_i \cdot \int e_v(k)\, dt + K_d \cdot \frac{d}{dt} e_v(k) \tag{2.34}$$

where $e_v(k)$ represents the speed error, defined as $e_v(k) = v^{e*}(k) - v^e(k)$. Here, $v(k)$ is the actual vehicle speed and $v^{e*}(k)$ is the desired target speed. The coefficients K_p, K_i, and K_d represent the proportional, integral, and derivative gains, respectively. The control output $a(k)$ is used to modulate either the throttle or brake commands.

As depicted in Figure 2.13, the target speed $v^{e*}(k)$ provided by the DM and the actual speed from the EKF are used to compute the speed error. This error is then used to adjust the accelerator or brake to maintain the desired vehicle speed.

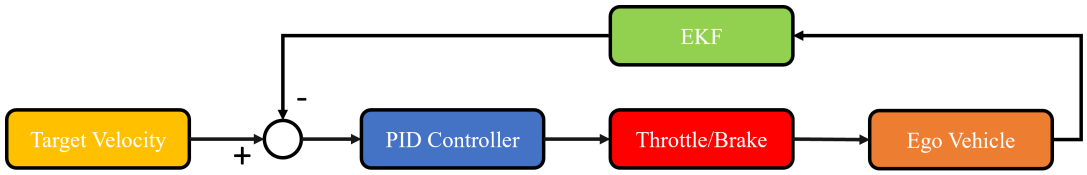


Figure 2.13: Schematic representation of the PID controller processing inputs from the EKF and DM to adjust the vehicle's throttle or brake settings.

The following table lists selected parameters for longitudinal control using the PID controller, all obtained by empirical tuning:

Table 2.1: Parameter values for the PID Controller in Longitudinal Control

Parameter	Value	Description
K_p	1.95	Proportional gain
K_i	0.05	Integral gain
K_d	0.5	Derivative gain

2.5.2 Lateral Control Using Two-Point Visual Control Model

The lateral control of the vehicle is implemented using the Two-Point Visual Control Model, a straightforward and effective method described by Salvucci (2004) [21]. This model utilizes visual tracking of two specific points along the roadway: a near point for immediate trajectory adjustments and a far point for maintaining overall directional stability.

The steering angle, δ, is determined based on the angles α_n and α_f, which correspond to the near and far points, respectively. These angles are calculated as follows:

$$\alpha_n = \arctan\left(\frac{y_n}{x_n}\right), \quad \alpha_f = \arctan\left(\frac{y_f}{x_f}\right) \tag{2.35}$$

where, (x_n, y_n) and (x_f, y_f) are the coordinates of the near and far points within the vehicle's local coordinate system, y_n and y_f is equal to the lateral error $e_y(k)$

Feedback from a PI controller for the near point and proportional control for the far point are integrated to compute the steering control input:

$$\delta^e(k) = K_{p,n} \cdot \alpha_n(k) + K_{i,n} \cdot \int \alpha_n(k)\, dt + K_{p,f} \cdot \alpha_f(k) \tag{2.36}$$

Here, $K_{p,n}$ and $K_{i,n}$ are the proportional and integral gains for the near point, and $K_{p,f}$ is the proportional gain for the far point.

Figure 2.14 shows how the target and actual lateral positions, provided by the DM and the EKF respectively, are used to calculate the lateral error. This error is then

input into the Two-Point Visual Control Model to adjust the steering angle, thus controlling the vehicle's lateral movement.

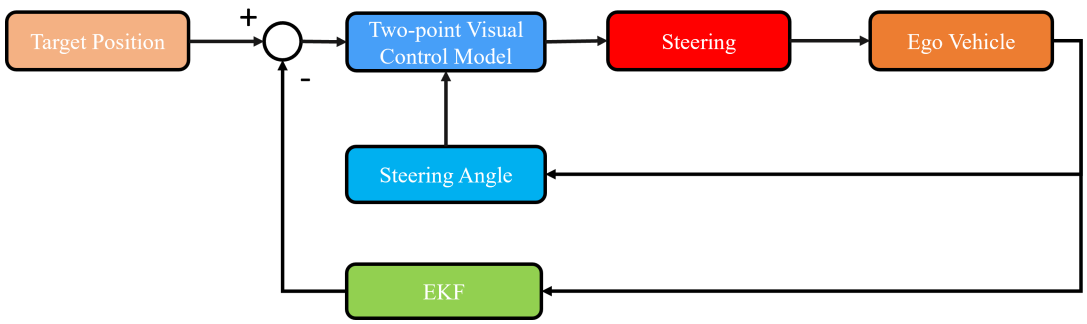


Figure 2.14: Schematic of the Two-Point Visual Control Model for lateral control. The diagram illustrates how input data from the DM and EKF are integrated to adjust steering based on calculations from near and far points, ensuring the vehicle maintains the desired trajectory.

The following table lists selected parameters for the Two-Point Visual Control Model, all obtained by empirical tuning:

Table 2.2: Parameter values for the PID Controller in Lateral Control

Parameter	Value	Description
$K_{p,n}$	1.5	Proportional gain for near point in PI controller
$K_{i,n}$	0.2	Integral gain for near point in PI controller
$K_{p,f}$	0.5	Proportional gain for far point in proportional controller
x_n	$15m$	Longitudinal distance to near point
x_f	$100m$	Longitudinal distance to far point

2.6 MPC Framework

Algorithm 1 details the predictive and planning processes employed by an Autonomous Heavy Vehicle operating in uncertain environments using an MPC framework, as specified in (2.28).

The algorithm starts by linearizing and discretizing the system. Once this is done, the main loop begins. In the main loop, the states of the ego vehicle are estimated continuously by EKF using (2.5). At every time step where $t \mod f = 0$, the states of the surrounding vehicles are estimated. E.g., if $f = 10$, the estimation happens at every 10^{th} time step. The estimation is done using a CV model for N steps ahead. The estimated states of the ego vehicle and the surrounding vehicles are then used to update the constraints for the next N steps. The algorithm then sets up the DM process by initializing the cost functions to infinity for every controller $j = \{$nc, lc, rc$\}$. The cost for each controller is then computed using (2.31). The decision-making process aims to select the control strategy that minimizes the cost. Once

the optimal strategy is identified and selected, the corresponding path $\hat{\mathbf{X}}^{e*}$ becomes the target for the local controller. In the subsequent control and tracking loop, the local controller employs both lateral and longitudinal PID control techniques to the ego vehicle to accurately follow the optimal trajectory determined by the DM.

Algorithm 1 Autonomous Vehicle Control System with MPC

Linearize vehicle kinematics
Discretize system
loop {Iterate over simulation time steps}
  if $t \geq T_{end}$ **then**
    exit loop
  end if
  Estimate ego state: $\hat{\mathbf{x}}^e(t) = \mathrm{EKF}(\mathbf{y}(t))$
  if $t \mod f = 0$ **then**
    Predict states of surrounding vehicles for N steps: $\hat{\mathbf{X}}_j$
    Update constraints $h_j^{\mathrm{e}}(\cdot)$ for N steps with $\hat{\mathbf{x}}^e(t)$ and $\hat{\mathbf{X}}_j$
    Initialize costs: $J^e_{\mathrm{nc}}, J^e_{\mathrm{lc}}, J^e_{\mathrm{rc}} = \infty$
    Solve MPC to compute optimal cost: $J^e_{\mathrm{nc}}, J^e_{\mathrm{lc}}, J^e_{\mathrm{rc}}$
    Compute DM costs: $g_j\left(\hat{\mathbf{X}}^e, \mathbf{U}^e, \mathbf{\Lambda}^e\right)$, for each $j \in \{\mathrm{nc}, \mathrm{lc}, \mathrm{rc}\}$
    Determine optimal path: $j^* = \arg\min g_j, j \in \{\mathrm{nc}, \mathrm{lc}, \mathrm{rc}\}$
    Select best path: $\hat{\mathbf{X}}^{e*} = \hat{\mathbf{X}}^e_{j^*}$
  end if
  Execute lateral and longitudinal control (PID) to follow $\hat{\mathbf{X}}^{e*}$
end loop

3

Background Theory on Stochastic Model Predictive Control

This chapter motivates the necessity of employing SMPC and discusses its underlying theories. It is organized into two main sections: Comparison between SMPC and Robust Model Predictive Control (RMPC) and Theoretical Foundations of SMPC.

3.1 Comparison between RMPC and SMPC

The uncertainty in the vehicle's position progressively increases over time. Although MPC and closed-loop systems are generally effective, they fail to significantly enhance robustness or guarantee performance when confronting systems marked by increasing uncertainty or stochastic variables.

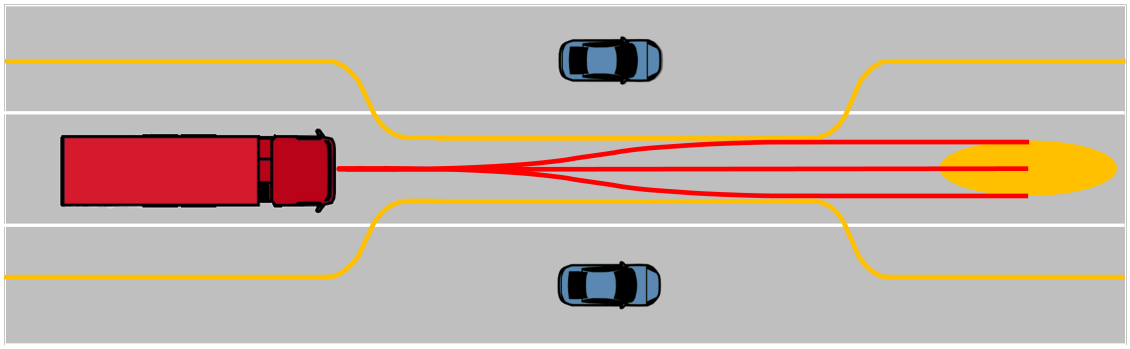


Figure 3.1: Diagram depicting the uncertainty in the vehicle system, where the orange ellipse represents the uncertainty in the x and y directions of the ego vehicle. The orange line denotes constraints, while the red lines illustrate potential trajectories.

3.1.1 Robust Model Predictive Control:

The robust control strategies, Min-Max Model Predictive Control (Min-Max MPC) and Tube-Based Model Predictive Control (Tube-Based MPC) address uncertainty and disturbances in system models.

Min-Max MPC:

Min-Max MPC [22][23] is based on the premise of preparing for the worst-case scenario. It utilizes a min-max optimization framework to safeguard the system against the most detrimental impacts of uncertainties and disturbances. This approach involves not only predicting future states under nominal conditions, but also considering the possible worst-case scenarios due to disturbances.

The state-space representation in Min-Max MPC can be expressed as:

$$\mathbf{x}(k+1) = \mathbf{A}_d\mathbf{x}(k) + \mathbf{B}_d\mathbf{u}(k) + \mathbf{g}_d + \mathbf{w}(k), \tag{3.1}$$

where, $\mathbf{w}(k)$ represents the worst-case disturbance at time step k. The objective is to minimize the maximum expected loss:

$$\begin{aligned} \min_{\mathbf{u}(k)} \max_{\mathbf{w}(k)} & \sum_{k=0}^{N-1} \left(\left\| \mathbf{x}(k) - \mathbf{x}^{\text{ref}}(k) \right\|_{\mathbf{Q}_p}^2 + \|\mathbf{u}(k)\|_{\mathbf{Q}_u}^2 \right) + \left\| \mathbf{x}(N) - \mathbf{x}^{\text{ref}}(N) \right\|_{\mathbf{Q}_f}^2 \\ \text{s.t.} \quad & \mathbf{x}(k+1) = \mathbf{A}_d\mathbf{x}(k) + \mathbf{B}_d\mathbf{u}(k) + \mathbf{g}_d + \mathbf{w}(k), \\ & \mathbf{x}(k) \in \mathbb{X}, \quad \mathbf{u}(k) \in \mathbb{U}, \quad \mathbf{w}(k) \in \mathbb{W}, \quad \forall k \in [0, N-1], \\ & \mathbf{x}(0) = \hat{\mathbf{x}}(t) \end{aligned} \tag{3.2}$$

Control inputs $\mathbf{u}$ are selected by considering the optimal path and worst possible outcomes due to future disturbances, ensuring robust performance across a range of scenarios.

Tube-Based MPC:

Tube-Based MPC [24][25] operates by generating a nominal trajectory, around which a protective tube is constructed. This tube encapsulates all feasible variations in the trajectory, which may arise due to model uncertainties and external disturbances, thus ensuring that the actual trajectory never deviates beyond this protective boundary.

The nominal system is defined by:

$$\tilde{\mathbf{x}}(k+1) = \mathbf{A}_d\tilde{\mathbf{x}}(k) + \mathbf{B}_d\tilde{\mathbf{u}}(k) + \mathbf{g}_d \tag{3.3}$$

where, $\tilde{\mathbf{x}}(k)$ represents the nominal state, and the estimated states $\hat{\mathbf{x}}(k)$ are expected to remain within a bounded region (tube) around $\tilde{\mathbf{x}}(k)$. This tube is dynamically adjusted based on both observed and predicted disturbances.

As is shown in Figure 3.2, the robust control input applied to the real system is expressed as:

$$\mathbf{u}_r(k) = \tilde{\mathbf{u}}(k) + \mathbf{K}(\tilde{\mathbf{x}}(k) - \hat{\mathbf{x}}(k)) \tag{3.4}$$

where, $\mathbf{K}$ denotes the feedback gain matrix, designed to correct deviations from the nominal trajectory. The $\mathbf{K}(\tilde{\mathbf{x}}(k) - \hat{\mathbf{x}}(k))$ is a feedback correction that compensates for the deviation between the nominal state and the estimated state, allowing the controller to adjust the control inputs in real-time to minimize the errors caused by model uncertainties and external disturbances. By incorporating this feedback

mechanism, the control system can dynamically adjust the control inputs to compensate for deviations due to uncertainties and disturbances, thus enhancing the robustness of the system.

The goal is to minimize the following cost function while ensuring that the state remains within the tube:

$$\begin{aligned} \underset{\tilde{\mathbf{u}}(k),\forall k\in[0,N-1]}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \left\|\hat{\mathbf{x}}(k) - \mathbf{x}^{\text{ref}}(k)\right\|_{\mathbf{Q}_p}^2 + \sum_{k=0}^{N-1} \|\mathbf{u}_r(k)\|_{\mathbf{Q}_u}^2 + \left\|\hat{\mathbf{x}}(N) - \mathbf{x}^{\text{ref}}(N)\right\|_{\mathbf{Q}_f} \\ \text{s.t.} \quad & \hat{\mathbf{x}}(k+1) = \mathbf{A_d}\hat{\mathbf{x}}(k) + \mathbf{B_d}\tilde{\mathbf{u}}(k) + \mathbf{g_d} + \mathbf{B_d}\mathbf{K}(\tilde{\mathbf{x}}(k) - \hat{\mathbf{x}}(k)), \\ & \hat{\mathbf{x}}(k) \in \mathbb{X}, \quad \tilde{\mathbf{u}}(k) \in \mathbb{U}, \\ & \hat{\mathbf{x}}(0) = \hat{\mathbf{x}}(t) \end{aligned} \tag{3.5}$$

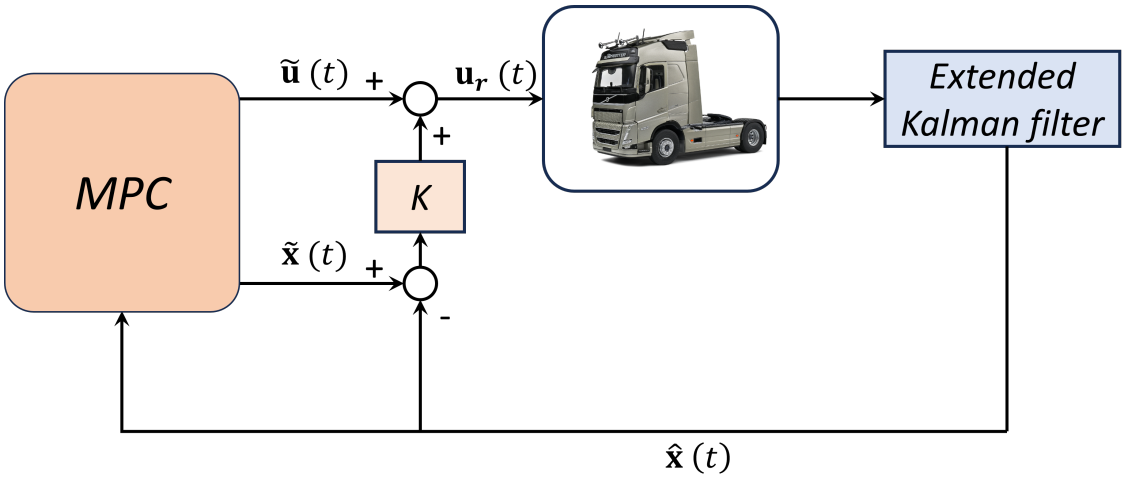


Figure 3.2: Diagram of the Tube-based MPC setup, where $\hat{\mathbf{x}}(t)$ is the state from the observer, $\mathbf{K}$ is the feedback gain, $\tilde{\mathbf{x}}(t)$ and $\tilde{\mathbf{u}}(t)$ are the nominal system states and inputs, respectively, and $\mathbf{u}_r(t)$ is the robust input.

3.1.2 Stochastic Model Predictive Control

SMPC extends the traditional MPC framework to effectively manage uncertainties by explicitly considering probabilistic disturbances within the system dynamics [26, 27]. Unlike RMPC, which operates under the assumption of worst-case scenarios, SMPC allows for a more flexible adherence to constraints by integrating random disturbances directly into the model and optimizing the expected performance over time. Additionally, SMPC permits constraint violations with a certain probability to enhance performance, ensuring that the likelihood of violation remains low [28].

In SMPC, the system dynamics can be described by the following state-space equation:

$$\mathbf{x}(k+1) = \mathbf{A}_d\mathbf{x}(k) + \mathbf{B}_d\mathbf{u}(k) + \mathbf{g}_d + \mathbf{w}(k) \tag{3.6}$$

where $\mathbf{w}(k)$ denotes the process noise, assumed to be normally distributed with zero mean and covariance Σ_w, i.e., $\mathbf{w}(k) \sim \mathcal{N}(0, \Sigma_w)$ in this paper. The matrices $\mathbf{A}_d$ and

$\mathbf{B}_d$ represent the linearized system dynamics and control input effects, respectively.

Probabilistic constraints in SMPC are formulated to allow a controlled degree of constraint violation, which is governed by chance constraints. These constraints specify the maximum acceptable probability of constraint breaches as:

$$P(\mathbf{x}(k) \in \mathbb{X}, \mathbf{u}(k) \in \mathbb{U}) \geq 1 - \epsilon = p \tag{3.7}$$

where ϵ represents a small predetermined probability of constraint violation, and p denotes the minimum probability of constraint satisfaction.

To manage uncertainties, SMPC often tightens constraints based on the risk level ϵ, adjusting deterministic constraints to more conservative levels. This adjustment reduces the feasible sets for $\mathbb{X}$ and $\mathbb{U}$, ensuring safe operation under stochastic disturbances.

The cost function in SMPC is designed to minimize expected operational costs while managing risks associated with probabilistic constraint violations. It is expressed as:

$$\begin{aligned} \underset{\mathbf{u}(k), \forall k \in [0, N-1]}{\text{minimize}} \; & \mathbb{E}\left[\sum_{k=0}^{N-1} \|\mathbf{x}(k)\|^2_{\mathbf{Q}_p} + \sum_{k=0}^{N-1} \|\mathbf{u}(k)\|^2_{\mathbf{Q}_u} + \|\mathbf{x}(N)\|^2_{\mathbf{Q}_f}\right] \\ \text{s.t.} \quad & \mathbf{x}(k+1) = \mathbf{A}_d \mathbf{x}(k) + \mathbf{B}_d \mathbf{u}(k) + \mathbf{g}_d + \mathbf{w}(k), \\ & P(h^T_{\mathbf{x},j} \mathbf{x}(k) \leq b_{\mathbf{x},j}) \geq p_{\mathbf{x},j}, \forall j = 1, ..., N_{c,\mathbf{x}}, \\ & P(h^T_{\mathbf{u},j} \mathbf{u}(k) \leq b_{\mathbf{u},j}) \geq p_{\mathbf{u},j}, \forall j = 1, ..., N_{c,\mathbf{u}}, \\ & \mathbf{x}(0) = \mathbf{x}(t) \end{aligned} \tag{3.8}$$

where $h_{\mathbf{x},j}$ and $h_{\mathbf{u},j}$ are vectors used to define linear constraints on the state and control variables. These vectors project the state and control variables to specific dimensions to enforce specific constraints. The constants $b_{\mathbf{x},j}$ and $b_{\mathbf{u},j}$ are the corresponding upper bounds for these affine constraints. Specifically, $b_{\mathbf{x},j}$ are the bounds for the state constraints, while $b_{\mathbf{u},j}$ are the bounds for the control constraints. Additionally, $N_{c,\mathbf{x}}$ and $N_{c,\mathbf{u}}$ denote the total number of constraints on the state variables and control variables, respectively. These constraints ensure that the state and control actions remain within safe and acceptable limits despite the presence of uncertainty.

The probabilistic constraints $P(h^T_{\mathbf{x},j} \mathbf{x}(k) \leq b_{\mathbf{x},j}) \geq p_{\mathbf{x},j}$ and $P(h^T_{\mathbf{u},j} \mathbf{u}(k) \leq b_{\mathbf{u},j}) \geq p_{\mathbf{u},j}$ ensure that the likelihood of violating these constraints stays below a specified threshold. This enhances the robustness of the control strategy in the face of perturbations and uncertainties.

This formulation allows the controller to balance expected performance with acceptable risk levels, ensuring cost-efficient control actions that adhere to safety standards under uncertainty.

3.2 Tractable Reformulation of SMPC Problems

To define the control policy, we express the state and control variables as deviations from their expected values (choosing an affine tube policy):

$$\mathbf{d}(k) = \mathbf{x}(k) - \tilde{\mathbf{x}}(k), \tag{3.9}$$

$$\mathbf{u}(k) = \mathbf{K}\mathbf{d}(k) + \tilde{\mathbf{u}}(k), \tag{3.10}$$

where $\mathbf{K}$ is the feedback gain matrix, $\tilde{\mathbf{x}}(k)$ and $\tilde{\mathbf{u}}(k)$ are the mean state and control actions, respectively. This indicates that the control input $\mathbf{u}(k)$ is a linear function of the state deviation $\mathbf{d}(k)$ plus a nominal control action $\tilde{\mathbf{u}}(k)$.

Given these definitions, the mean and the estimated covariance of the next state can be derived as follows. For the mean state:

$$\tilde{\mathbf{x}}(k+1) = \mathbf{A}_d\tilde{\mathbf{x}}(k) + \mathbf{B}_d\tilde{\mathbf{u}}(k) + \mathbf{g}_d, \tag{3.11}$$

For the estimated covariance of the states:

$$\hat{\mathbf{\Sigma}}(k+1) = (\mathbf{A}_d + \mathbf{B}_d\mathbf{K})\hat{\mathbf{\Sigma}}(k)(\mathbf{A}_d + \mathbf{B}_d\mathbf{K})^T + \hat{\mathbf{\Sigma}}_w, \tag{3.12}$$

where $\hat{\mathbf{\Sigma}}_w$ represents the estimated covariance of the process noise, and $\hat{\mathbf{\Sigma}}(k)$ is the estimated covariance of the states.

SMPC often models system states as Gaussian distributed to integrate uncertainty directly into the control framework effectively. However, it is not strictly necessary to assume Gaussian distribution for the states. When this assumption is made, at each time step k, the state vector can be considered normally distributed:

$$\mathbf{x}(k) \sim \mathcal{N}(\tilde{\mathbf{x}}(k), \hat{\mathbf{\Sigma}}(k)) \tag{3.13}$$

Constraints within SMPC are designed to keep the states within a specified safe set, defined as:

$$\mathbf{x}(k) \in \mathbb{X} = \{\mathbf{x} \mid h_{\mathbf{x},j}^T\mathbf{x}(k) \leq b_{\mathbf{x},j}, \forall j = 1, ..., N_{c,\mathbf{x}}\} \tag{3.14}$$

The transformation of constraints under Gaussian assumptions is given by:

$$h_{\mathbf{x},j}^T\mathbf{x}(k) \sim \mathcal{N}(h_{\mathbf{x},j}^T\tilde{\mathbf{x}}(k), h_{\mathbf{x},j}^T\hat{\mathbf{\Sigma}}(k)h_{\mathbf{x},j}) \tag{3.15}$$

To clarify the role of the Cumulative Distribution Function (CDF) and its application in ensuring probabilistic constraints, the following example demonstrates its use. The probability that the constraints are satisfied is quantified using the CDF of the standard normal distribution, ϕ and is expressed as:

$$P(h_{\mathbf{x},j}^T\mathbf{x} \leq b_{\mathbf{x},j}) = \phi\left(\frac{b_{\mathbf{x},j} - h_{\mathbf{x},j}^T\tilde{\mathbf{x}}(k)}{\sqrt{h_{\mathbf{x},j}^T\hat{\mathbf{\Sigma}}(k)h_{\mathbf{x},j}}}\right) \tag{3.16}$$

To ensure that the constraint is satisfied with a probability of at least p, we adjust the threshold using the inverse CDF, ϕ^{-1}. This adjustment defines the required critical value for the constraint:

$$b_{\mathbf{x},j} - h_{\mathbf{x},j}^T\hat{\mathbf{x}}(k) \geq \sqrt{h_{\mathbf{x},j}^T\hat{\mathbf{\Sigma}}(k)h_{\mathbf{x},j}}\phi^{-1}(p) \tag{3.17}$$

Therefore, the tightened constraint becomes:

$$h_{\mathbf{x},j}^T \hat{\mathbf{x}}(k) \leq b_{\mathbf{x},j} - \sqrt{h_{\mathbf{x},j}^T \hat{\mathbf{\Sigma}}(k) h_{\mathbf{x},j}} \phi^{-1}(p) \tag{3.18}$$

For example, if we want a 95% probability that the constraint is satisfied, we set $p = 0.95$. The inverse CDF, $\phi^{-1}(0.95)$, is approximately 1.645. This means that we adjust the constraint by subtracting 1.645 times the standard deviation term, resulting in a more conservative constraint. This back-off term accounts for uncertainties in the model and ensures a higher level of safety.

As is shown in Figure 3.3, the example illustrates the application of the above theory. The initial constraint is shown as a red dashed line, while the distribution is filled in blue. As seen from the left figure, only half of the distribution satisfies the constraint. By applying the above theory and setting the desired constraint satisfaction to 95%, the distribution shifts leftward to ensure that 95% of the distribution meets the requirement, as shown in the pink-filled distribution.

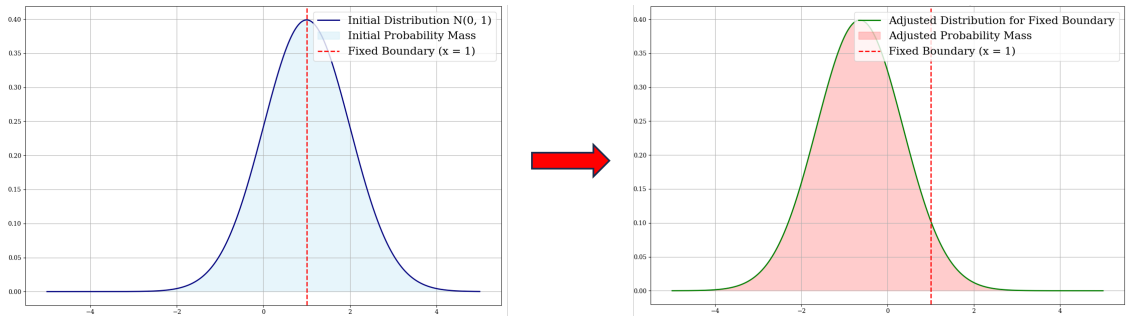


Figure 3.3: Distribution change according to the chance constraint for $p = 0.95$. The blue profile represents the initial distribution, and the red profile illustrates the distribution adjusted by the chance constraint, ensuring that 95% of the states fall within the constraints.

Additionally, the expected value of the quadratic form can be expressed in the following form:

$$\mathbb{E}(\|\mathbf{z}\|_{\mathbf{Q}}^2) = \|\mathbb{E}(\mathbf{z})\|_{\mathbf{Q}}^2 + \mathrm{tr}(\mathbf{Q}\,\mathrm{var}(\mathbf{z})) \tag{3.19}$$

In this expression, $\mathbf{z}$ represents a random vector, and $\mathbf{Q}$ is a positive semi-definite matrix. The notation $\mathbb{E}(\cdot)$ denotes the expected value and $\mathrm{tr}(\cdot)$ denotes the trace of a matrix. This decomposition allows us to separate the effect of the mean and the variability of $\mathbf{z}$ on the quadratic form.

The cost function (3.8) can be expressed as a Gaussian Linear Quadratic Control (GLQC) framework with half-space constraints by means of the equations (3.11),

(3.12) and (3.19):

$$
\begin{aligned}
\underset{\tilde{\mathbf{u}}(k),\mathbf{K},\forall k\in[0,N-1]}{\text{minimize}} \quad & \sum_{k=0}^{N-1}\left(\|\tilde{\mathbf{x}}(k)\|^2_{\mathbf{Q}_p}+\|\tilde{\mathbf{u}}(k)\|^2_{\mathbf{Q}_u}+\mathrm{tr}(\mathbf{Q}_u\mathbf{K}\hat{\mathbf{\Sigma}}(k)\mathbf{K}^T)\right) \\
& \quad +\|\tilde{\mathbf{x}}(N)\|^2_{\mathbf{Q}_f}+\mathrm{tr}(\mathbf{Q}_f\hat{\mathbf{\Sigma}}(N)) \\
\text{s.t.} \quad & \tilde{\mathbf{x}}(k+1)=\mathbf{A}_d\tilde{\mathbf{x}}(k)+\mathbf{B}_d\tilde{\mathbf{u}}(k)+\mathbf{g}_d, \\
& \hat{\mathbf{\Sigma}}(k+1)=(\mathbf{A}_d+\mathbf{B}_d\mathbf{K})\hat{\mathbf{\Sigma}}(k)(\mathbf{A}_d+\mathbf{B}_d\mathbf{K})^T+\hat{\mathbf{\Sigma}}_w, \\
& h^T_{\mathbf{x},j}\tilde{\mathbf{x}}(k)\le b_{\mathbf{x},j}-\sqrt{h^T_{\mathbf{x},j}\hat{\mathbf{\Sigma}}(k)h_{\mathbf{x},j}}\phi^{-1}(p), \quad \forall j=1,...,N_{c,x}, \\
& h^T_{\mathbf{u},j}\tilde{\mathbf{u}}(k)\le b_{\mathbf{u},j}-\sqrt{h^T_{\mathbf{u},j}\mathbf{K}\hat{\mathbf{\Sigma}}(k)\mathbf{K}^Th_{\mathbf{u},j}}\phi^{-1}(p), \quad \forall j=1,...,N_{c,u}, \\
& \tilde{\mathbf{x}}(0)=\mathbf{x}(t), \quad \hat{\mathbf{\Sigma}}(0)=\mathbf{P}
\end{aligned}
\tag{3.20}
$$

where $\mathbf{P}$ is the initial covariance matrix, which can be chosen to be zero or can be obtained from the steady state covariance matrix of the Kalman filter.

Although the untractable problem was solved by converting to the GLQC framework, the non-convexity in the cost function remains. This is because the quadratic cost function is associated with state feedback, which introduces non-linearity. In addition, the covariance matrix incorporating the state and control variables introduces a square root in quadratic form, which further complicates the optimization process. These non-convex properties greatly increase the complexity and computational requirements for solving the SMPC problem, making it difficult to ensure the global optimality and stability of the control solution.

To simplify the Gaussian linear quadratic control problem with half-space constraints, the focus is placed on optimizing the control sequence $\hat{\mathbf{u}}(k)$ while fixing the feedback gain matrix $\mathbf{K}$. By setting $\mathbf{K}$ to a constant, the control input $\hat{\mathbf{u}}$ becomes a linear function of the state deviation, thus reducing the cost function to a linear quadratic structure. This adjustment makes it possible to optimize only $\hat{\mathbf{u}}(k)$, converting the problem into a standard QP problem.

In addition, the dynamics of the system variance (represented by the covariance matrix) is designed to be independent of the control variable $\tilde{\mathbf{u}}(k)$. This separation of mean and covariance dynamics is a major advantage of our method, allowing for constraint adjustment based on pre-computed uncertainty. The uncertainty can be pre-calculated by fixing the feedback gain $\mathbf{K}$ in (3.12). In this way, these adjustments can be incorporated into the optimization problem as static constraints, thus simplifying real-time calculations and improving the predictability of control results. This approach is particularly effective for linear systems as it allows for the separation of error dynamics, which is not possible in nonlinear systems.

Through a series of simplifications and transformations above, the cost function can finally be expressed in the following form:

$$
\begin{aligned}
\underset{\tilde{\mathbf{u}}(k),\forall k\in[0,N-1]}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \|\tilde{\mathbf{x}}(k)\|^2_{\mathbf{Q}_p} + \sum_{k=0}^{N-1} \|\tilde{\mathbf{u}}(k)\|^2_{\mathbf{Q}_u} + \|\tilde{\mathbf{x}}(N)\|^2_{\mathbf{Q}_f} \\
\text{s.t.} \quad & \tilde{\mathbf{x}}(k+1) = \mathbf{A}_d\tilde{\mathbf{x}}(k) + \mathbf{B}_d\tilde{\mathbf{u}}(k) + \mathbf{g}_d, \\
& h^T_{\mathbf{x},j}\tilde{\mathbf{x}}(k) \leq b_{\mathbf{x},j} - \sqrt{h^T_{\mathbf{x},j}\hat{\mathbf{\Sigma}}(k)h_{\mathbf{x},j}}\phi^{-1}(p), \quad \forall j = 1,...,N_{c,\mathbf{x}}, \\
& h^T_{\mathbf{u},j}\tilde{\mathbf{u}}(k) \leq b_{\mathbf{u},j} - \sqrt{h^T_{\mathbf{u},j}\mathbf{K}\hat{\mathbf{\Sigma}}(k)\mathbf{K}^T h_{\mathbf{u},j}}\phi^{-1}(p), \quad \forall j = 1,...,N_{c,\mathbf{u}}, \\
& \tilde{\mathbf{x}}(0) = \mathbf{x}(t)
\end{aligned}
\tag{3.21}
$$

4

Trajectory Planning Using SMPC

This chapter updates the trajectory planning architecture outlined in Section 2.3, which primarily employs MPC and a DM to optimize trajectory and control actions as per a defined cost function given in (2.28). Unlike the deterministic approach previously described, we now incorporate SMPC, as explained in Section 3.2. It includes formulations for the ego vehicle's constraints, traffic constraints, SMPC optimization, and the SMPC framework.

4.1 Ego Vehicle Constraint Formulation

In both simulated environments such as CARLA and real-world scenarios, discrepancies invariably exist between a vehicle's theoretical model and its actual behavior. While kinematic models provide reasonable approximations of vehicle motion, inherent uncertainties persist. These uncertainties can amplify within the control system, subsequently impacting the efficacy and precision of vehicle control operations.

To model these uncertainties in the SMPC framework, Gaussian noise is employed. This choice is motivated by Gaussian noise's properties of being well-understood, having efficient computational techniques for estimation and control, and being a common assumption in system noise modeling due to the Central Limit Theorem. In this approach, the uncertainty is represented through Gaussian noise integrated into the system's linear equations, using a fixed LQR gain:

$$\hat{\mathbf{x}}^e(k+1) = \mathbf{A}_d\hat{\mathbf{x}}^e(k) + \mathbf{B}_d\hat{\mathbf{u}}^e(k) + \mathbf{g}_d \tag{4.1}$$

$$\hat{\mathbf{\Sigma}}^e(k+1) = (\mathbf{A}_d + \mathbf{B}_d\mathbf{K}_{LQR})\hat{\mathbf{\Sigma}}^e(k)(\mathbf{A}_d + \mathbf{B}_d\mathbf{K}_{LQR})^T + \mathbf{\Sigma}_w \tag{4.2}$$

$$\hat{\mathbf{x}}^e(0) = \mathbf{x}^e(0), \quad \hat{\mathbf{\Sigma}}^e(0) = \mathbf{P} \tag{4.3}$$

where, $\mathbf{P}$ denotes the stationary covariance, derived from the Kalman filter, while $\hat{\mathbf{\Sigma}}^e(k)$ is the propagated covariance at time step k, $\mathbf{K}_{LQR}$ is the LQR gain, and $\mathbf{\Sigma}_w$ represents the process noise, encapsulating the stochastic nature of the environment.

The LQR gain matrix $\mathbf{K}_{LQR}$ is a crucial component in the control design, where the goal is to minimize the quadratic cost function that takes into account both the state $\mathbf{x}$ and the control input $\mathbf{u}$. This optimization problem is solved using the discrete-time algebraic Riccati equation:

$$\mathbf{P}_{LQR} = \mathbf{A}_d^T\mathbf{P}_{LQR}\mathbf{A}_d - (\mathbf{A}_d^T\mathbf{P}_{LQR}\mathbf{B}_d)(R + \mathbf{B}_d^T\mathbf{P}_{LQR}\mathbf{B}_d)^{-1}(\mathbf{B}_d^T\mathbf{P}_{LQR}\mathbf{A}_d) + \mathbf{Q}_p \tag{4.4}$$

where, $\mathbf{P}_{LQR}$ is the solution to the Riccati equation, $\mathbf{Q}_p$ and R are the state and control weighting matrices, respectively.

Once $\mathbf{P}_{LQR}$ is determined, the optimal LQR gain matrix $\mathbf{K}_{LQR}$ is calculated as follows:

$$\mathbf{K}_{LQR} = (R + \mathbf{B}_d^T \mathbf{P}_{LQR} \mathbf{B}_d)^{-1} (\mathbf{B}_d^T \mathbf{P}_{LQR} \mathbf{A}_d), \tag{4.5}$$

Based on the above SMPC theory, state and control constraints can be tightened in the following form:

$$\mathbf{x}^e(k) \geq \mathbf{x}^e_{\min} + \sqrt{h_{\mathbf{x},j}^T \Sigma^e(k) h_{\mathbf{x},j}}\, \phi^{-1}(p) \tag{4.6a}$$

$$\mathbf{x}^e(k) \leq \mathbf{x}^e_{\max} - \sqrt{h_{\mathbf{x},j}^T \Sigma^e(k) h_{\mathbf{x},j}}\, \phi^{-1}(p) \tag{4.6b}$$

$$\mathbf{u}^e(k) \geq \mathbf{u}^e_{\min} + \sqrt{h_{\mathbf{u},j}^T \mathbf{K}_{LQR} \Sigma^e(k) \mathbf{K}_{LQR}^T h_{\mathbf{u},j}}\, \phi^{-1}(p) \tag{4.7a}$$

$$\mathbf{u}^e(k) \leq \mathbf{u}^e_{\max} - \sqrt{h_{\mathbf{u},j}^T \mathbf{K}_{LQR} \Sigma^e(k) \mathbf{K}_{LQR}^T h_{\mathbf{u},j}}\, \phi^{-1}(p) \tag{4.7b}$$

4.2 Traffic Constraints Formulation

For the trailing controller, the constraints are determined by maintaining a specified distance from the vehicle in front, as given in the deterministic MPC equation (2.25). For SMPC, an extension parameter is added to the constraints to include uncertainty in the vehicle model. The modified chance constraint also includes the slack variables, as shown in the following equation:

$$c_{\text{nc, SMPC}}\left(\hat{x}^e, \hat{x}^{(l)}, \hat{v}_x^{(l)}, \lambda^e_{\text{nc}}, \alpha_x\right) = d + L^{\text{truck}} + T\hat{v}_x^{(l)} + \hat{x}^e - \hat{x}^{(l)} + \lambda^e_{\text{nc}} + \phi(p)^{-1}\alpha_x < 0 \tag{4.8}$$

The additional term $\phi(p)^{-1}\alpha_x$ in (4.8) represents the uncertainty in the x position of the ego-vehicle,

$$\alpha_x(k) = \sqrt{h_{\mathbf{x}}^T \hat{\mathbf{\Sigma}}^e(k) h_{\mathbf{x}}} \tag{4.9}$$

where $h_{\mathbf{x}} = [1, 0, 0, 0]$. The equation (4.9) is then represented in the simplified form as,

$$\alpha_x(k) = \sqrt{\hat{\Sigma}_x(k)} \tag{4.10}$$

where, $\hat{\Sigma}_x(k)$ is an estimate of variance associated with x position at the time step k.

By doing so, the trailing constraint is now tightened. As illustrated in Figure 4.1, the tightened constraint is shown as solid yellow lines, while the constraint before tightening is presented as dashed yellow lines.

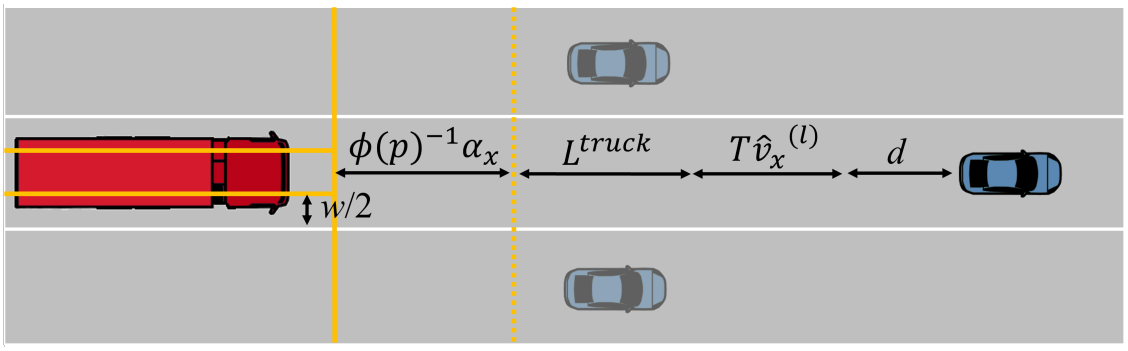


Figure 4.1: SMPC trailing constraint, where the ego vehicle is denoted by a red truck and the leading vehicle is denoted by a blue car. $\hat{v}_x^{(l)}$ is the estimated speed of the vehicle ahead, T is the minimum headway distance required for braking, and d is the minimum longitudinal distance to the vehicle ahead in the same lane, thus forming the initial trailing constraint (yellow dashed line). The SMPC trailing constraint is obtained by tightening the constraint with $\phi(p)^{-1}\alpha_x$, as shown in the yellow solid line. In addition, the lateral restriction for ego vehicles states that their COG plus half of their maximum lateral incline ($w/2$) must not exceed the edge of the lane.

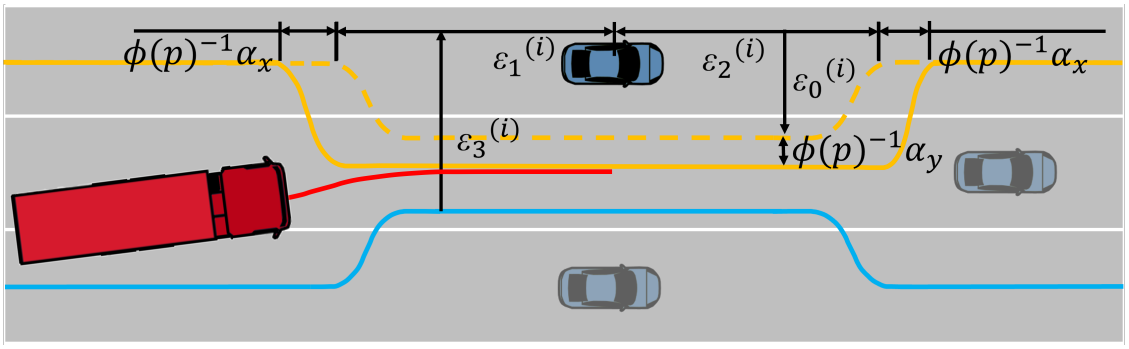
For the lane-changing scenario, (2.26) is modified similarly as (4.8). The equation is extended to include the uncertainties present in the x, y-states of the ego vehicle, as shown below:

$$\begin{aligned} b^{(i)}_{\text{SMPC}}\left(\hat{x}^e, \hat{y}^e, \hat{x}^{(i)}, \hat{y}^{(i)}, \alpha_x, \alpha_y\right) = \frac{\varepsilon_0^{(i)}\left(\hat{y}^{(i)}, \alpha_y\right)}{2} \Big[\tanh\left(\hat{x}^e - \hat{x}^{(i)} + \varepsilon_1^{(i)} + \phi(p)^{-1}\alpha_x\right) + \\ \tanh\left(\hat{x}^{(i)} - \hat{x}^e + \varepsilon_2^{(i)} + \phi(p)^{-1}\alpha_x\right)\Big] \\ + \varepsilon_3^{(i)}\left(\hat{y}^e, \hat{y}^{(i)}\right) \end{aligned} \tag{4.11}$$

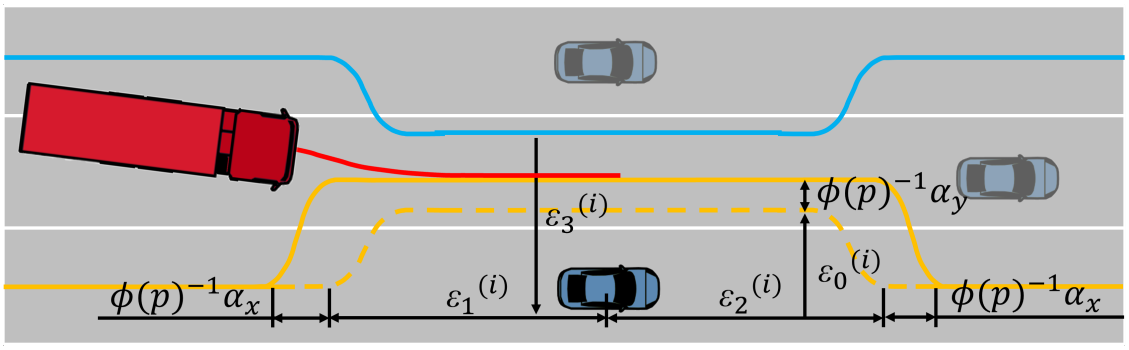
where additional term α_x, α_y are the uncertainties present in the x,y-positions of the ego-vehicle. Here $\alpha_y(k) = \sqrt{\hat{\Sigma}_y(k)}$ is calculated in the same way as (4.9) having $h_{\mathbf{x}} = [0, 1, 0, 0]$. The modified chance constraint for lane changing scenario with slack variables is as follows,

$$c_{j,\text{SMPC}}\left(\hat{x}^e, \hat{y}^e, \hat{x}^{(i)}, \hat{y}^{(i)}, \lambda_{\text{e}}^j, \alpha_x, \alpha_y\right) = F_j^{(i)}\left(b^{(i)}_{\text{SMPC}}(\cdot) - \hat{y}^e + w\right) + \lambda_{\text{e}}^j < 0 \tag{4.12}$$

The modified lane-changing scenario is illustrated in Figure 4.2



(a) SMPC Left lane change scenario constraint.



(b) SMPC Right lane change scenario constraint.

Figure 4.2: The ego vehicle is represented by the red truck, and the surrounding vehicles are represented by blue cars. The initial constraints (yellow dashed lines) for vehicle i are defined by $\varepsilon_0^{(i)}$ (for scaling the constraints laterally), two consecutive *tanh* functions (scaled by $\varepsilon_1^{(i)}$ and $\varepsilon_2^{(i)}$, respectively), and $\varepsilon_3^{(i)}$ (for shifting the constraints to correct the lateral position) are defined. The constraints around the surrounding vehicle are now more tightened (solid yellow line) by adding the terms $\phi(p)^{-1}\alpha_x$ and $\phi(p)^{-1}\alpha_y$ to the initial constraints.

4.3 Optimal Control Problem Formulation

The SMPC optimization formulation follows the same method as in Section 2.3.5. The cost function is defined as,

$$J^e_{j,\text{SMPC}}\left(\hat{\mathbf{X}}^e_j, \mathbf{U}^e_j, \mathbf{\Lambda}^e_j\right) = \sum_{k=0}^{N-1}\left(\left\|\hat{\mathbf{x}}^e_j(k) - \mathbf{x}^e_{\text{ref},j}(k)\right\|^2_{\mathbf{Q}_p} + \left\|\lambda^e_j(N)\right\|^2_{\mathbf{Q}_s}\right) \tag{4.13a}$$

$$+ \sum_{k=0}^{N-1}\left\|\mathbf{u}^e_j(k)\right\|^2_{\mathbf{R}} \tag{4.13b}$$

$$+ \left\|\hat{\mathbf{x}}^e_j(N) - \mathbf{x}^e_{\text{ref},j}(N)\right\|^2_{\mathbf{Q}_f} + \left\|\lambda^e_j\right\|^2_{\mathbf{Q}_s} \tag{4.13c}$$

$$+ \sum_{k=0}^{N-2}\left\|\mathbf{u}^e_j(k+1) - \mathbf{u}^e_j(k)\right\|^2_{\mathbf{R}_d} \tag{4.13d}$$

$$\tag{4.13e}$$

The key difference here is that the states and control inputs are now tightened as given in (4.6) and (4.7).

The general constraints equation for the ego-vehicle can be formulated in similar way as in (2.27) by combining the physical limitations of the ego-vehicle as defined in (4.6), (4.7) and traffic constraints as formulated in (4.8) and (4.12).

$$h^{\mathrm{e}}_{j,\mathrm{SMPC}}\left(\hat{\mathbf{x}}^e_j(k), \hat{\mathbf{u}}^e_j(k), \lambda^{\mathrm{e}}_j(k), \hat{\mathbf{x}}_j(k), \hat{\mathbf{u}}_j(k), \hat{\mathbf{\Sigma}}^e(k)\right) \leq 0 \tag{4.14}$$

Using the cost function defined in (4.13) and the constraints (4.14), the SMPC optimization problem can be formulated as:

$$\underset{\hat{\mathbf{X}}^e_j, \mathbf{U}^e_j, \mathbf{\Lambda}^e_j}{\text{minimize}} \quad J^e_{j,\mathrm{SMPC}}\left(\hat{\mathbf{X}}^e_j, \mathbf{U}^e_j, \mathbf{\Lambda}^e_j\right) \tag{4.15a}$$

$$\text{s.t.} \quad \hat{\mathbf{x}}^e_j(k+1) = \mathbf{A}_{\mathrm{d}}\hat{\mathbf{x}}^e_j(k) + \mathbf{B}_{\mathrm{d}}\mathbf{u}^e_j(k) + \mathbf{g}_{\mathrm{d}}, \tag{4.15b}$$

$$(\hat{x}^{(i)}(k+1), \hat{y}^{(i)}(k+1)) = \Pi_{\mathrm{CV}}(\hat{x}^{(i)}(t), \hat{y}^{(i)}(k)), \quad \forall i \in \mathbb{V}(k), \tag{4.15c}$$

$$h^{\mathrm{e}}_{j,\mathrm{SMPC}}\left(\hat{\mathbf{x}}^e_j(k), \hat{\mathbf{u}}^e_j(k), \lambda^{\mathrm{e}}_j(k), \hat{\mathbf{x}}_j(k), \hat{\mathbf{u}}_j(k), \hat{\mathbf{\Sigma}}^e(k)\right) \leq 0, \tag{4.15d}$$

$$\lambda^e_j(k) \leq 0 \tag{4.15e}$$

$$\hat{\mathbf{x}}^e_j(0) = \hat{\mathbf{x}}^e(t) \tag{4.15f}$$

The optimal trajectory among the three controllers is selected by defining the Decision Master of the SMPC in the same way as in (2.31):

$$g_j\left(\hat{\mathbf{X}}^e_j, \mathbf{U}^e_j, \mathbf{\Lambda}^e_j\right) = q_e J^e_{j,\mathrm{SMPC}}\left(\hat{\mathbf{X}}^e_j, \mathbf{U}^e_j, \mathbf{\Lambda}^e_j\right) + q_s J^s_j \tag{4.16}$$

$$\underset{\hat{\mathbf{X}}^e_j, \mathbf{U}^e_j, \mathbf{\Lambda}^e_j}{\text{argmin}} \quad g_j\left(\hat{\mathbf{X}}^e, \mathbf{U}^e, \mathbf{\Lambda}^e\right), \quad j \in \{\mathrm{nc}, \mathrm{lc}, \mathrm{rc}\} \tag{4.17}$$

4.4 SMPC Framework

Algorithm 2 outlines the prediction and planning methodology used by an Autonomous Heavy Vehicle navigating in uncertain environments utilizing SMPC Framework as described in (4.13).

The algorithm begins by linearizing and discretizing the vehicle kinematics. The LQR gain is then pre-computed as given in (4.5). This gain helps in determining the system's uncertainty $\hat{\mathbf{\Sigma}}$ for the ego vehicle over N prediction steps. Using this, the constraints are formulated as detailed in (4.6, 4.7, 4.8, and 4.12).

In the main loop, like in Algorithm 1, the states of the ego-vehicle are estimated continuously by EKF using (2.5). At every time step where $t \mod f = 0$, the states of the surrounding vehicles are estimated. E.g., if $f = 5$, the estimation happens at every 5^{th} time step. The estimation is done using a CV model for N steps ahead. The estimated states of the ego-vehicle and the surrounding vehicles are then used

to update the constraints for the next N steps.

The algorithm then sets up the DM process by initializing the cost functions to high for every controller $j = \{\text{nc, lc, rc}\}$. The cost for each controller is then computed using (2.31). The decision-making process aims to select the control strategy that minimizes the cost. Once the optimal strategy is identified, and selected, denoted by j^*, the corresponding path $\hat{\mathbf{X}}^{e*}$ becomes the target for the local controller.

The local controller then takes over to execute both lateral and longitudinal control commands using the PID controller to accurately follow the planned trajectory. This ensures the vehicle's path is optimized for safety and efficiency.

Algorithm 2 Autonomous Vehicle Control System with SMPC

Linearize vehicle kinematics
Discretize system
Compute fixed LQR gain
Compute system uncertainty for N steps
loop {Iterate over simulation time steps}
  if $t \geq T_{end}$ **then**
    exit loop
  end if
  Estimate ego state: $\hat{\mathbf{x}}^e(t) = \text{EKF}(\mathbf{y}(t))$
  if $t \mod f = 0$ **then**
    Predict states of surrounding vehicles for N steps: $\hat{\mathbf{X}}_j$
    Update constraints $h^{\text{e}}_{j,\text{SMPC}}(\cdot)$ for N steps with $\hat{\mathbf{x}}^e(t)$, $\hat{\mathbf{\Sigma}}^e$ and $\hat{\mathbf{X}}_j$,
    Initialize costs only once: $J^e_{\text{nc,SMPC}}, J^e_{\text{lc,SMPC}}, J^e_{\text{rc,SMPC}} = \infty, \quad g_{\min} = \infty$
    Solve MPC to compute optimal cost: $J^e_{\text{nc,SMPC}}, J^e_{\text{lc,SMPC}}, J^e_{\text{rc,SMPC}}$
    Compute DM cost: $g_j = q_e J^e_{j,\text{SMPC}}\left(\hat{\mathbf{X}}^e_j, \mathbf{U}^e_j, \mathbf{\Lambda}^e_j\right) + q_s J^s_j$
    Determine optimal path: $j^* = \arg\min g_j, j \in \{\text{nc}, \text{lc}, \text{rc}\}$
    Select best path: $\hat{\mathbf{X}}^{e*} = \hat{\mathbf{X}}^e_{j^*}$
  end if
  Execute lateral and longitudinal control (PID) to follow $\hat{\mathbf{X}}^{e*}$
end loop

5

Results and Analysis

In this chapter, the results of the MPC and SMPC controllers are presented. Experiments are conducted using the CARLA simulator, beginning with a detailed description of the experimental setup and test scenarios. The effectiveness of the EKF is then verified, and the realization of the SMPC constraints is demonstrated. Subsequently, the decision-making processes and actual performance of the MPC and SMPC controllers in the CARLA simulator are comparatively analyzed. Finally, the performance of both controllers is evaluated through a collision avoidance test.

5.1 CARLA Simulator

To test and validate the algorithm and vehicle behavior under realistic conditions, the CARLA simulator was used. CARLA is an open-source simulator that offers a highly realistic virtual environment with detailed urban and suburban landscapes, dynamic traffic, and weather conditions. Its modular architecture allows for easy integration of custom vehicle models and control algorithms, making it suitable for a wide range of autonomous driving research applications [29]. The installation and the documentation on how to use CARLA can be found in [30].

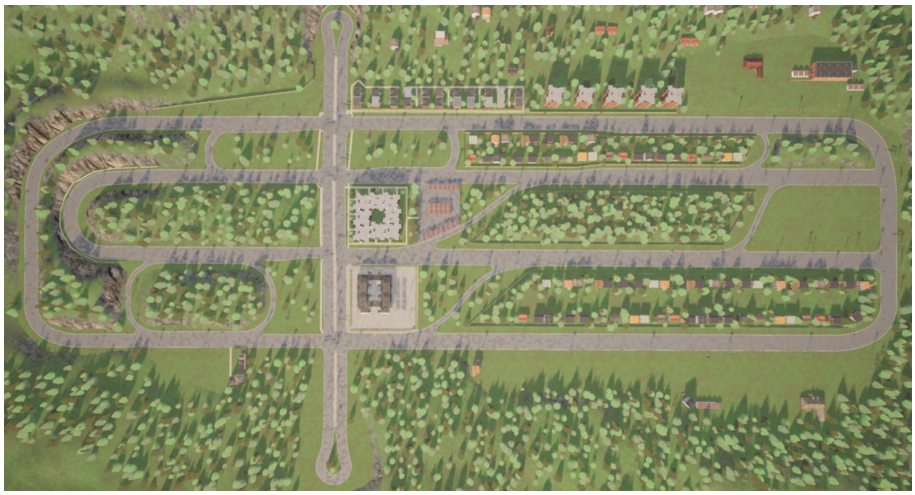


Figure 5.1: Bird eye view of highway traffic scenario on Map 06 from CARLA.

5.1.1 Experimental Setup in CARLA

In this report, CARLA version 0.9.15 is utilized, and Map06 from CARLA's selection of predefined maps has been chosen to assess and validate the algorithm and vehicle behavior, as illustrated in Figure 5.1. Map 06 provides a highway scenario with multiple lanes, offering an ideal environment to test the algorithms for highway lane changing and trailing scenarios.

The CARLA simulator offers various pre-configured vehicle models. In this study, a fire truck and a Tesla Model 3 from CARLA were used to depict the ego vehicle and surrounding vehicles, respectively, as seen in Figure 5.2. Vehicle dimensions are summarized in Table 5.1, and their expected speeds under different driving behaviors are detailed in Table 5.2.

Figure 5.2: Firetruck and Tesla Model 3 as seen in CARLA simulator.

Table 5.1: Dimensions of Vehicles Used in CARLA Simulation

Vehicle Type	Length (m)	Width (m)	Wheelbase (m)
Tesla Model 3	5.00	2.00	2.87
Fire Truck	8.46	2.89	6.00

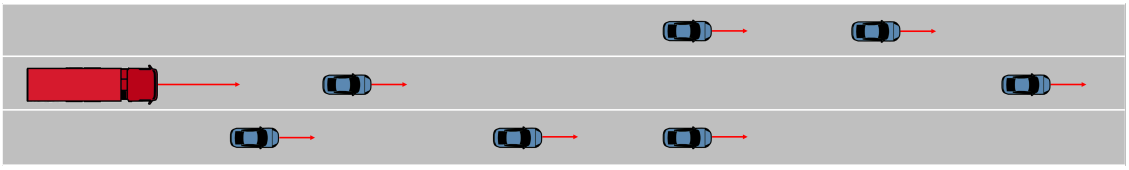
Table 5.2: Expected Speeds of Vehicles Under Different Driving Behaviors (m/s)

Vehicle Type	Aggressive(m/s)	Normal(m/s)	Passive(m/s)
Tesla Model 3	20	15	11.25
Fire Truck	20	15	11.25

5.1.2 Test Scenario Setting

To evaluate the performance of MPC and SMPC, we designed an extremely complex traffic congestion scenario in the CARLA simulation environment. Figure 5.3a shows a conceptual schematic of this scenario, while Figure 5.3b shows a real view of this scenario in CARLA. In this scenario, the red ego vehicle is surrounded by seven Model 3 vehicles operating at passive speeds. To maintain the desired traveling speed of 15 m/s and ultimately reach the destination safely, the autonomous vehicle needs to make multiple decisions on the road, including trailing, left change, and right change.

This design was very effective in helping us test the response and efficiency of the controller as well as the DM for the three different decisions. By testing this complex scenario, we can evaluate the performance of each control strategy in real traffic conditions in detail and further validate our control algorithm.



(a) Conceptual diagram of the traffic congestion scenario used to evaluate the performance of MPC and SMPC controllers in CARLA. This schematic illustrates the red ego vehicle behind seven vehicles, each operating under a passive velocity model.

(b) Representation of the actual test environment in CARLA, featuring the red ego vehicle behind seven vehicles in urban traffic conditions.

Figure 5.3: Illustration of the complex environment utilized to evaluate the performance of MPC and SMPC.

5.2 EKF Validation and Results

Additionally, the CARLA API provides high-precision data about the vehicle's state, including position, velocity, acceleration, and heading angle. These data points accurately reflect the actual state of the vehicle within the CARLA simulation environment. To enhance the simulation's realism, it is necessary to introduce measurement noise into these data points. Following the addition of noise, an EKF is employed to estimate and feed data into both the longitudinal and lateral control systems.

Measurement errors are mainly inaccuracies in the measurement of x and y positions by the LiDAR system. Similarly, velocity measurement errors stem from the limitations of the speedometer, while angular errors are detected by the inertial measurement unit (IMU). Table 5.3 shows these measurement errors and their variances, with data from [31][32]. The original CARLA data is error-free, but to fit the actual situation, we added measurement errors to the data in the program based on parameters in table 5.3.

Table 5.3: Measurement Error Specifications for Kalman Filter

Parameters	Error Variance	Description
x	0.01	Position in the x direction (m^2)
y	0.01	Position in the y direction (m^2)
v	0.01	Velocity $(m/s)^2$
ψ	$\left(\frac{1}{180}\pi\right)^2$	Orientation angle (rad^2)

Selecting appropriate process noise parameters is crucial for accurate vehicle dynamics simulations using the Extended Kalman Filter. We adjust these parameters based on the maximum expected deviations of each state variable to effectively model the system's uncertainty and dynamics. Table 5.4 provides the process noise values for each state variable.

Table 5.4: Process Noise Specifications for Kalman Filter

Parameters	Error Variance	Description
x	0.3	Position in the x direction (m^2)
y	0.05	Position in the y direction (m^2)
v	0.5	Velocity $(m/s)^2$
ψ	0.01^2	Orientation angle (rad^2)

The effectiveness of the EKF is demonstrated through visual results. Figures 5.4, 5.5, and 5.6 compare estimated trajectories and positional errors against actual measurements.

Figure 5.4 provides a comparative analysis between the Kalman filter estimations and the actual vehicle trajectory in the CARLA simulation environment. Figure 5.5

displays the x direction error, reducing and stabilizing within acceptable limits over time. In Figure 5.6, the error in the y direction also narrows significantly, maintaining tight bounds. These results confirm that our choice of process noise parameters is effective. The graphs demonstrate that the Kalman filter accurately tracks vehicle movements with minimal error, proving its value in real-world applications. These results confirm the process noise parameter selections and illustrate the Kalman filter's precision in tracking vehicle movements with minimal error.

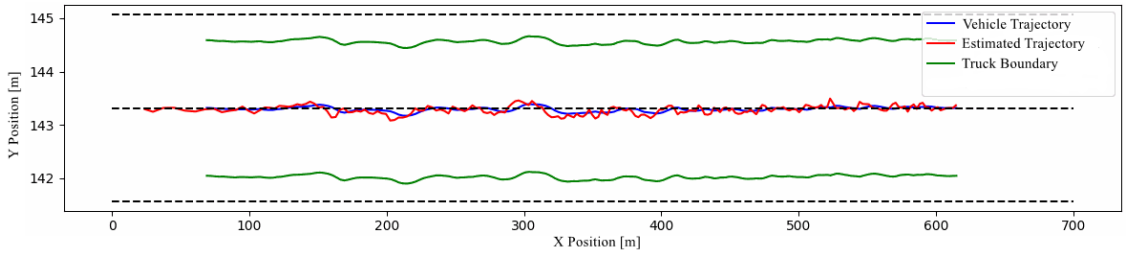


Figure 5.4: Comparison of estimation and vehicle trajectory. The red curve represents estimation results from the Kalman filter; the blue curve depicts the actual vehicle trajectory; the green outlines the vehicle boundaries.

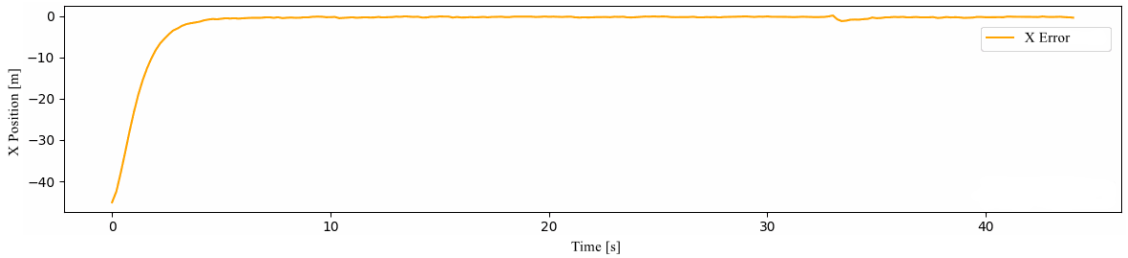


Figure 5.5: Kalman filter error in the x direction. Initially showing deviation, the error converges significantly, maintaining within 6 σ.

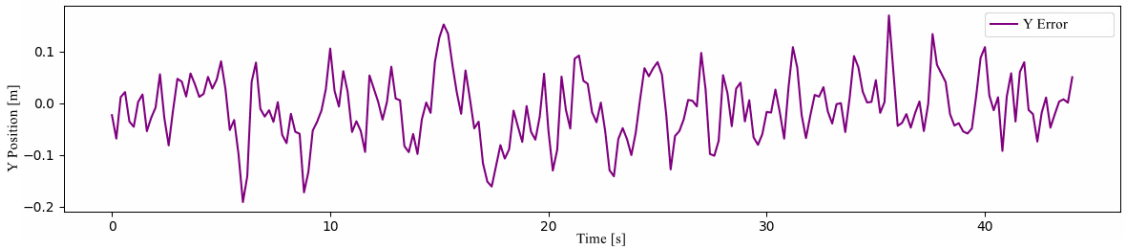


Figure 5.6: Kalman filter error in the y direction. The error narrows to within [-0.1, 0.1] (within 6 σ), demonstrating the filter's precision.

5.3 SMPC Constraints Implementation

In the context of the linearization and discretization process based on equations (2.14) to (2.16), the dynamic system around the fixed operating point $\bar{\mathbf{x}} = [0, 0, 15, 0]$

and $\bar{\mathbf{u}} = [0, 0]$ is considered. The matrices $\mathbf{A}_d$ and $\mathbf{B}_d$ are computed at the specified operating point with a sampling time of 0.3 s. Although this method of linearizing the dynamic equations at a fixed operating point may have inaccuracies, this method helps to precompute the fixed LQR gains and uncertainties, which is critical to reducing computation time.

$$\mathbf{A}_d = \begin{bmatrix} 1 & 0 & 0.3 & 0 \\ 0 & 1 & 0 & 4.5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

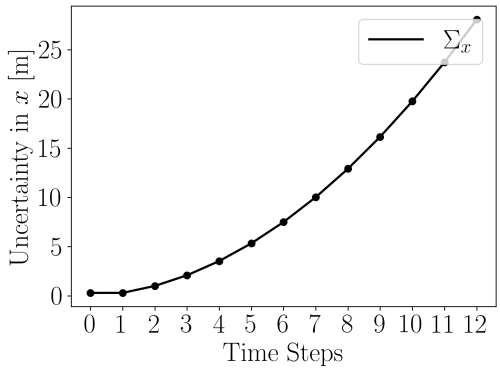
$$\mathbf{B}_d = \begin{bmatrix} 0 & 0 \\ 2.25 & 0 \\ 0 & 0.3 \\ 1.5 & 0 \end{bmatrix}$$

For the specific system, applying equations mentioned in Section 4 results in the LQR gain matrix:

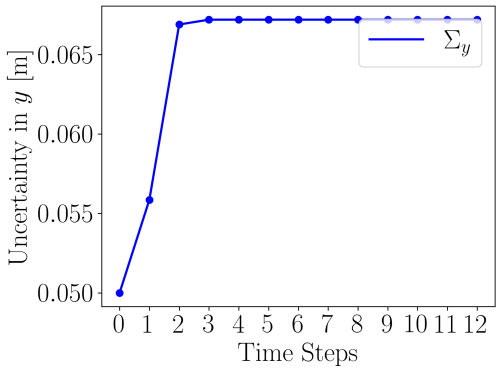
$$\mathbf{K}_{LQR} = \begin{bmatrix} 0 & -0.2092 & 0 & -1.2907 \\ 0 & 0 & -2.8743 & 0 \end{bmatrix}$$

From (3.11) and (3.12), the variance of the state of the system increases over time, which can have a significant impact on the dynamics of the system. Figures 5.7a to 5.7d visually illustrate the increase in variance. In the graph 5.7a, the black curve shows that the variance (Σ_x) continues to increase over successive time intervals, highlighting the growth of uncertainty in the x direction. Figure 5.7b shows a similar increase in Σ_y, illustrating the continued growth of uncertainty. The green curve in Figure 5.7c shows the variance Σ_v, which flattens out after reaching 0.51. The red curve in the graph 5.7d represents the variance Σ_ψ, which stabilizes after a slight increase to 0.006. In addition, these variance results serve as pre-compute constraint adjustment to tighten the SMPC constraints.

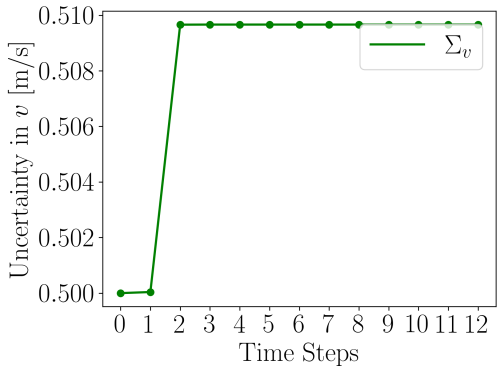
Figure 5.8 illustrates the growth of the position uncertainty of the ego vehicle over time when using the LQR controller during straight-line driving. The orange ellipse in the figure represents the uncertainty region of the vehicle's COG, which gradually grows over time. If this increase in uncertainty is not adequately accounted for in the planning and control process, it will seriously affect the control performance of the autonomous vehicle when maintaining a trailing or lane-changing maneuver, increasing the risk of collision with surrounding vehicles.



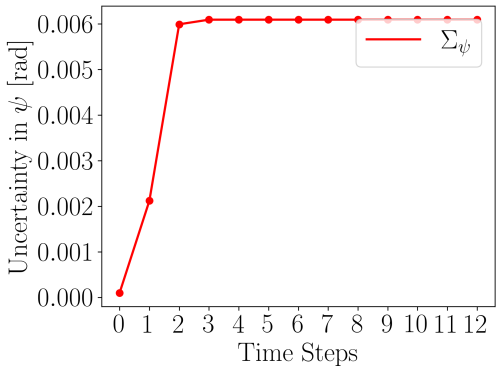
(a) Increase in Σ_x overtime



(b) Increase in Σ_y over time



(c) Increase in Σ_v overtime



(d) Increase in Σ_ψ overtime

Figure 5.7: Increases in uncertainty values over time for different variables: Σ_x, Σ_y, Σ_v, and Σ_ψ.

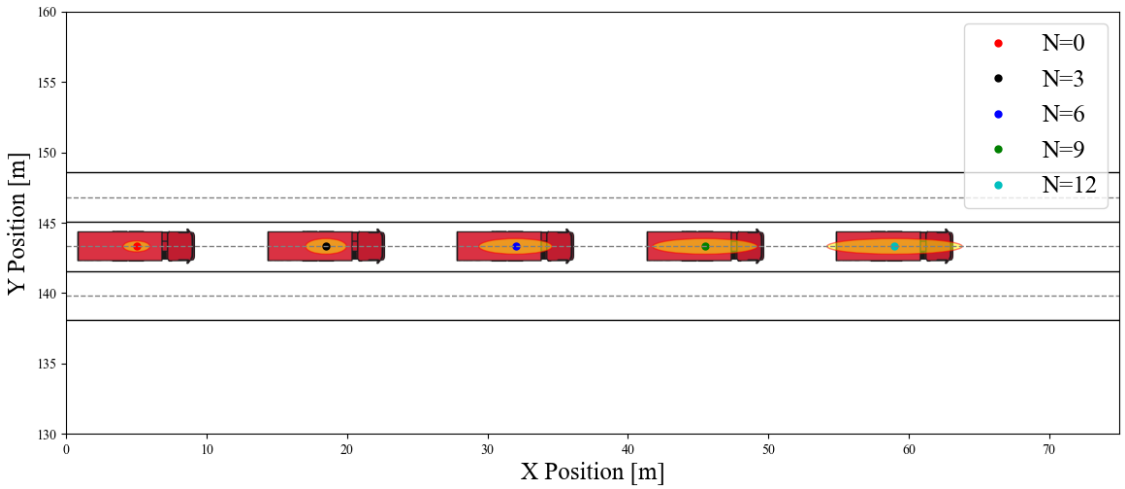


Figure 5.8: Uncertainty Propagation of the Ego Vehicle under LQR Control during Straight-Line Driving. The red truck represents the ego vehicle, and the orange ellipse illustrates the spatial uncertainty in the x and y directions at various time horizons.

Table 5.5 outlines the operational limits for the system's state variables to ensure

safety and compliance. The table sets the permissible range for each variable. For example, the x direction position ranges from 0 to 5000 meters, accommodating broad operational areas. This value is indicative and can be any large number, as we do not impose a specific upper limit on the x direction to ensure flexibility in various scenarios. In contrast, the y direction has narrower limits from 138.05 to 148.55 meters, reflecting constraints due to proximity to road edges. The velocity variable v^e is limited to 30 m/s and does not permit reverse movement, thus ranging only from 0 to 30 m/s. The orientation angle ψ^e and the steering angle δ^e are both limited to $-\frac{\pi}{8}$ to $\frac{\pi}{8}$ radians, matching typical steering capabilities. Similarly, the acceleration is restricted from -0.5g to 0.5g, ensuring a balance between agility and safety.

Table 5.5: Defined constraints for system states and inputs

Variable	Lower Bound	Upper Bound	Description
x^e	0	5000	Position in x-direction (m)
y^e	138.05	148.55	Position in y-direction (m)
v^e	0	30	Velocity (m/s)
ψ^e	$-\frac{\pi}{8}$	$\frac{\pi}{8}$	Orientation angle (rad)
a^e	$-0.5g$	$0.5g$	Acceleration (m/s^2)
δ^e	$-\frac{\pi}{8}$	$\frac{\pi}{8}$	Steering angle (rad)

Figures 5.9 to 5.14 display the dynamic adaptation of constraints for different state variables over various time steps, visually detailing how the chance constraints adjust bounds to meet evolving operational demands and conditions. These figures vividly represent the practical implementation of constraint tightening to enhance system safety and performance. The visual analysis provided by these figures helps in understanding how theoretical adjustments are applied in real-world scenarios, illustrating the adaptive nature of constraint management in response to dynamic conditions and requirements.

This constraint tightening is based on the SMPC theory described in Section 4 with a selection probability of 95%. Due to the fixed LQR gain and matrices $\mathbf{A}_d$ and $\mathbf{B}_d$, the value of the uncertainty is constant, and therefore there is no need to recalculate Σ. Therefore, based on the probabilities and uncertainties, we can derive progressively tighter states and input constraints, as shown in Figure 5.9 to 5.14.

As illustrated in Figure 5.9, since we do not impose strict constraints in the x direction, the tightening does not significantly impact the constraints. In contrast, the other figures clearly show that the state and input constraints are progressively tightened and become more stringent as the time step increases.

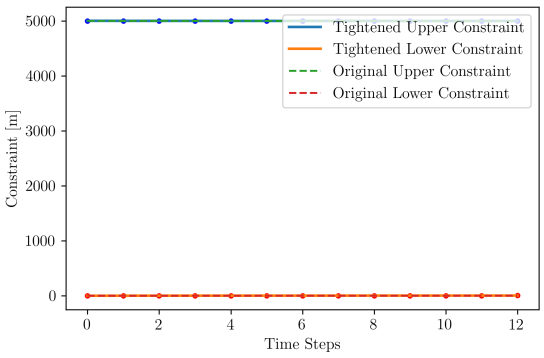


Figure 5.9: Comparison of the initial constraints and the constraints of tightening for x direction state quantities at different time steps. Blue represents the upper bound after tightening, orange represents the lower bound, and the initial constraints are shown with green dashed (upper) and red dashed (lower) lines.

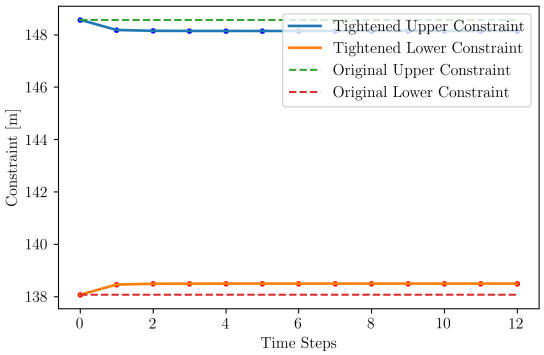


Figure 5.10: Comparison of the initial constraints and the constraints of tightening for y direction state quantities at different time steps. The blue and orange lines indicate the upper and lower bounds after tightening, respectively. Initial bounds are depicted as green and red dashed lines.

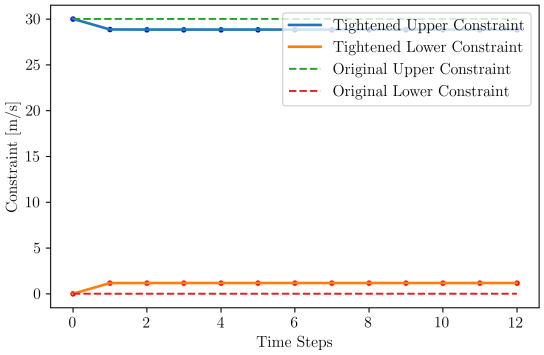


Figure 5.11: Dynamic adjustment of velocity constraints (v^e) over time. The constraints shift from initial bounds (green upper, red lower dashed lines) to tightened bounds (blue upper, orange lower). The tightened constraints ensure velocities do not exceed the maximum limit.

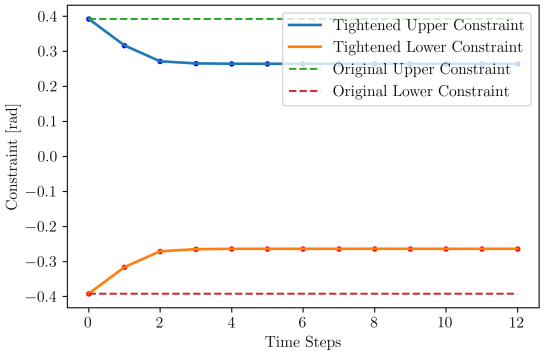


Figure 5.12: Adjustment of heading angle (ψ^e) constraints across various time steps. The blue and orange lines represent the tightened upper and lower bounds, respectively, with initial conditions shown as green and red dashed lines.

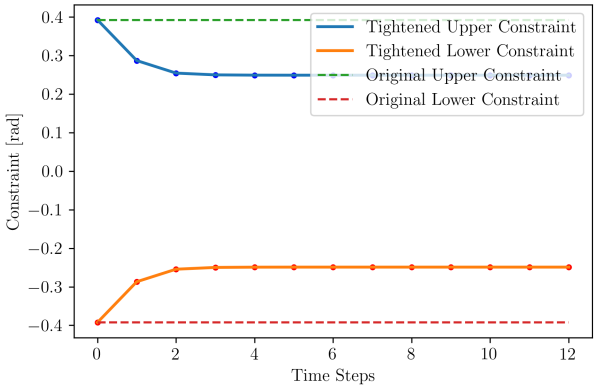


Figure 5.13: The steering angle constraint (δ^e) adjusts dynamically over time. The constraint changes from an initial constraint (green upper dashed line, red lower dashed line) to a tightened constraint (blue upper dashed line, orange lower dashed line).

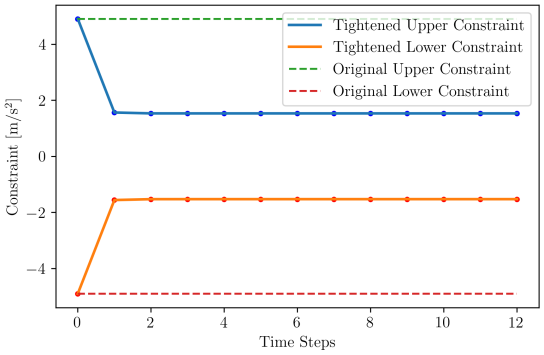
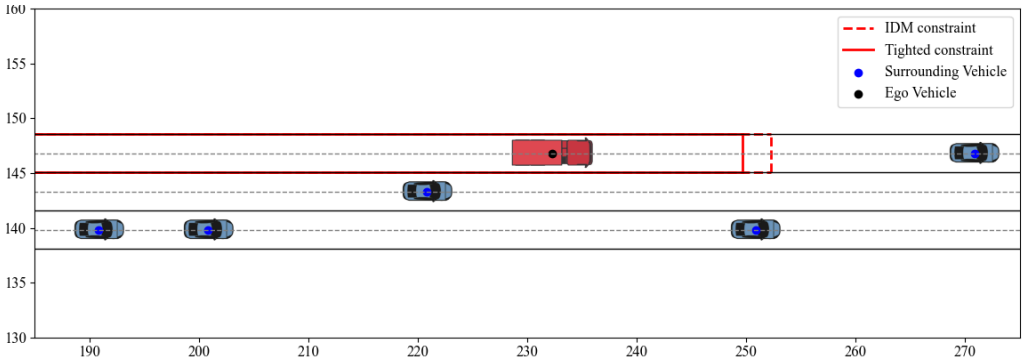


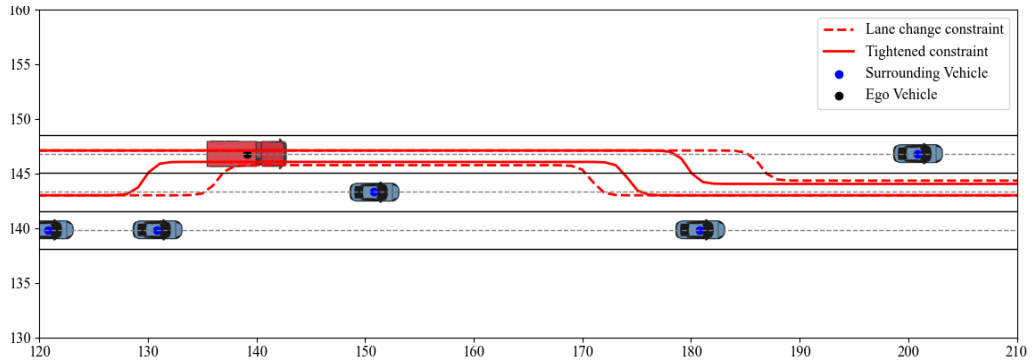
Figure 5.14: Acceleration (a^e) limit adjustments for different time steps. The blue and orange lines represent the tightened upper and lower limits, respectively, while the green and red dashed lines represent the initial conditions.

Figure 5.15a demonstrates how SMPC adjusts the constraints (red solid lines) at step $N = 1$ due to the uncertainty. Compared to the original MPC-defined trailing constraints (red dashed line), these tightened constraints provide additional redundancy space, enhancing the system's resilience against disturbances and ensuring safety performance.

Similarly, as depicted in Figure 5.15b, the lane changing constraints affect both the lane occupied by the ego vehicle and the adjacent right lane. These constraints, defined by the nonlinear functions in (4.11) for the solid lines and (2.23) for the dashed lines, encapsulate the maneuvering vehicle. The tightened constraints (red solid lines), as opposed to the initial constraints (red dashed lines), are expanded in the x and y directions. This expansion ensures that the vehicle avoids collisions with surrounding vehicles during lane-changing maneuvers.



(a) Comparison of MPC and SMPC Constraints in the Trailing Scenario



(b) Comparison of MPC and SMPC Constraints in the Overtaking Scenario

Figure 5.15: Illustrations of adaptive constraints under SMPC. The ego vehicle (red truck) and the leading vehicle (blue car) are shown. The red dashed lines represent the original traffic constraints under standard MPC, while the red solid lines show the constraints after being tightened based on SMPC theory.

5.4 Optimization Solver

In the implementation of MPC and SMPC, the CasADi [33] and the optimization solver IPOPT are crucial. CasADi helps by providing tools for symbolic representation and algorithmic differentiation. These tools are important for setting up the dynamic models and constraints used in SMPC, enabling the efficient setup of optimization problems critical for simulating control strategies effectively.

IPOPT, which works well with CasADi, tackles these optimization challenges efficiently. As a solver designed for large-scale nonlinear optimization problems, IPOPT is well-suited for the complex and nonlinear aspects of our MPC problems.

Additionally, the nonlinear problem-solving involved in this study was performed on a Dell Precision 7750 workstation configured with an Intel(R) Core(TM) i7-10850H CPU and 32 GB of RAM.

5.5 Analysis of Decision-Making in MPC and SMPC

In this section, the decision-making process under the SMPC framework is analyzed and compared with the MPC framework. The parameters used by these frameworks in the simulation are summarized in Table 5.6, which are crucial for the setup and evaluation of our control strategies.

Table 5.6 provides detailed simulation parameters under both MPC and SMPC control strategies, including time step, sampling time, reference speed, and the reference position of each lane. In addition, the table lists the state weight matrices, slack variable weight matrices, input weight matrices, terminal state weight matrices, and input change weight matrices, which are the key parameters for the performance of the control strategies.

5.5.1 Decision-Making Process using MPC and SMPC

The decision-making process and the trajectory predictions of the controllers for the ego vehicle are shown in Figure 5.16. Figures 5.16a to 5.16d show the results of four iterations, namely Iteration 10, 20, 30 and 40. The controller and slack costs for SMPC are shown in Table 5.7 and Table 5.8.

In the figure 5.16a, the ego vehicle is positioned in the center lane, and the two surrounding vehicles are at the center and right lane during the first iteration, Iteration 10. As the ego vehicle is in the center lane, the DM predicts trajectories for all three controllers $j = \{\text{nc, lc, rc}\}$. In Table 5.7, the left lane change controller has the lowest controller cost. The slack cost, shown in Table 5.8, is the lowest for the left lane change controller. Here, the DM chooses the left lane change controller and the ego vehicle turns to the left lane.

Table 5.6: Simulation Parameters for MPC and SMPC

Parameter	Description	MPC	SMPC
N	Time Steps	12	12
T_s	Sampling time (s)	0.3	0.3
v^e_{ref}	Reference velocity (m/s)	15	15
$y^e_{\text{ref},l}$	Ref y pos at left lane (m)	139.8	139.8
$y^e_{\text{ref},m}$	Ref y pos at middle lane (m)	143.3	143.3
$y^e_{\text{ref},r}$	Ref y pos at right lane (m)	146.8	146.8
$\mathbf{Q}_p$	State weighting matrix	diag(0,40,300,5)	diag(0,80,300,5)
$\mathbf{Q}_s$	Slack var weighting matrix	10^5	10^5
$\mathbf{R}$	Input weighting matrix	diag(5,5)	diag(5,5)
$\mathbf{Q}_f$	Terminal state weighting matrix	diag(0,40,300,5)	diag(0,40,300,5)
$\mathbf{R}_d$	Input change weighting matrix	diag(1000,1000)	diag(1000,1000)
q_e	Cost function significance	1	1
q_s	Controller switching parameter	100	100

In Iteration 20, the ego vehicle is now approaching the left lane, shown in figure 5.16b. The DM predicts the trajectories for the trailing and left lane change controller. The right lane change controller is turned off. The controller cost for this iteration is 31.30 for left lane change and 636.10 for trailing, as seen in Table 5.7. The lowest slack cost in Iteration 20 is for the trailing controller. The DM chooses the left lane controller trajectory and the ego vehicle continues to merge into the left lane.

In Iteration 30, depicted in Figure 5.16c, the ego vehicle is now in the left lane. The DM continues to predict the trajectories for trailing a left change controller. The controller costs are 1.78 for a left lane change, and 1.78 for trailing, as presented in Table 5.7. The lowest slack cost is for the trailing controller with a cost value of 2.90×10^{-21}. DM decides to go for the optimal trajectory from the trailing controller.

In Iteration 40, shown in Figure 5.16d, the ego vehicle continues to drive in the left lane. The trajectory predictions from the trailing and left change controller are evaluated. From the Table 5.7 and 5.8, the controller cost and slack cost for the trailing controller are the lowest in this iteration. The DM chooses the trajectory predictions from the trailing controller.

The decision-making process for MPC is similar to that explained for SMPC. The trajectory predictions and decision-making process of the ego vehicle for the MPC framework are shown in Figure 5.16e to 5.16h for the four iterations. The controller and slack costs for MPC are presented in Table 5.9 and Table 5.10.

In Iteration 10, depicted in Figure 5.16e, the ego vehicle, positioned in the middle lane, decides to trail. The trailing controller has the lowest controller cost of 2.77 (Table 5.9) and the lowest slack cost of 8.33×10^{-19} (Table 5.10). In Iteration 20,

the DM decides to do a left change and the left change controller has the lowest controller cost of 929.61 (Table 5.9) and lowest slack cost of 3.75×10^{-17} (Table 5.10). In Iteration 30, depicted in 5.16g, the ego vehicle is changing to the left lane, and the trajectory predictions for left change and trailing controller are evaluated. The lowest controller cost of 32.93 (Table 5.9) is for the left change controller, and the lowest slack cost 2.35×10^{-21} is for the trailing controller. In Iteration 40, the ego vehicle is on the left lane and the trajectory predictions are done for left change and trailing controllers.

Table 5.7: Controller costs and decisions at different steps for SMPC framework

Step	Left Change	Right Change	Trailing	Decision
10	935.41	4446.31	2565.15	Change Left
20	31.30	-	636.10	Change Left
30	1.78	-	1.78	Trailing
40	4441.35	-	2.55	Trailing

Table 5.8: Slack values at different steps for SMPC framework

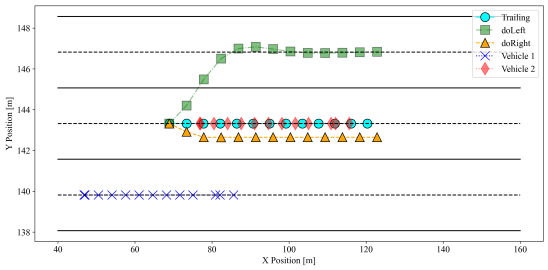
Step	Slack Left	Slack Right	Slack Trailing
10	3.75×10^{-17}	1.58	6.81
20	6.10×10^{-17}	-	1.91×10^{-22}
30	8.39×10^{-18}	-	2.90×10^{-21}
40	1.57	-	7.85×10^{-21}

Table 5.9: Controller costs and decisions at different steps for MPC Framework

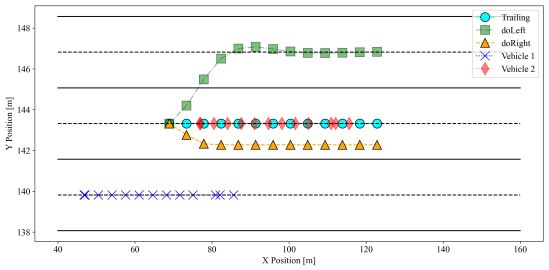
Step	Left Change	Right Change	Trailing	Decision
10	935.91	3589.43	8.77	Trailing
20	929.61	5721.91	31886.70	Change Left
30	32.93	-	630.95	Change Left
40	1.39	-	1.39	Trailing

Table 5.10: Slack values at different iterations for MPC framework

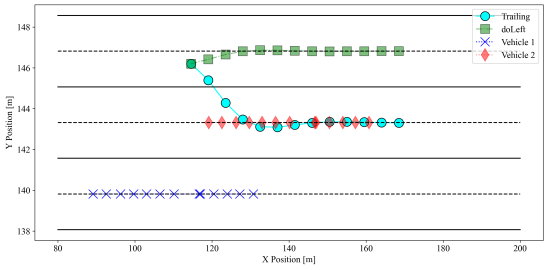
Step	Slack Left	Slack Right	Slack Trailing
10	3.75×10^{-17}	1.31	8.33×10^{-19}
20	3.75×10^{-17}	10.42	60.86
30	6.03×10^{-17}	-	2.35×10^{-21}
40	6.46×10^{-17}	-	5.79×10^{-21}



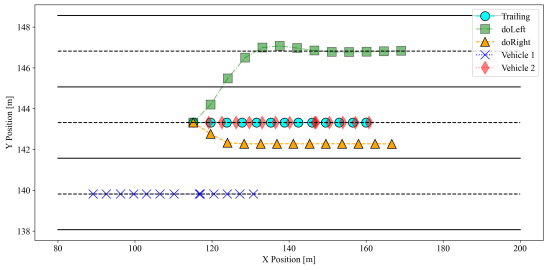
(a) Trajectory predictions at Iteration 10 using SMPC



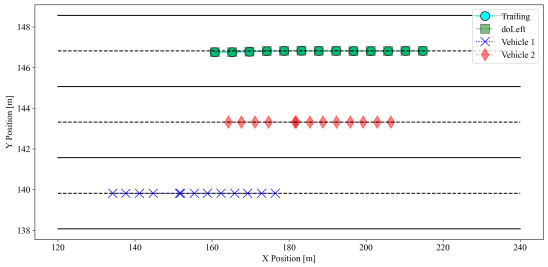
(e) Trajectory predictions at Iteration 10 using MPC



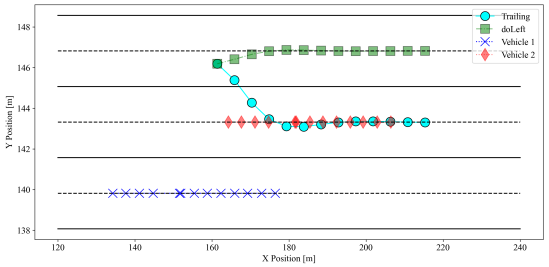
(b) Trajectory predictions at Iteration 20 using SMPC



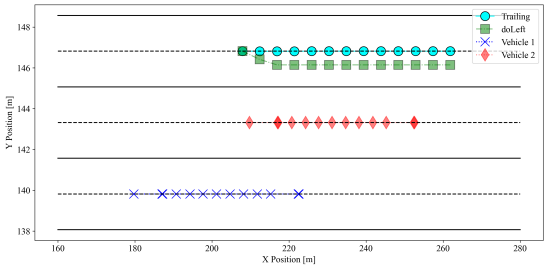
(f) Trajectory predictions at Iteration 20 using MPC



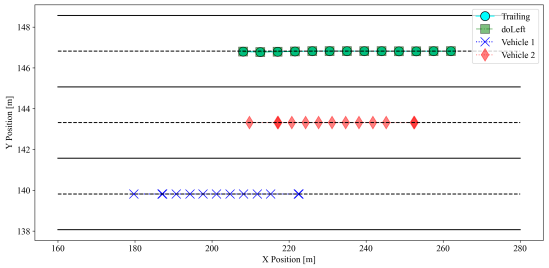
(c) Trajectory predictions at Iteration 30 using SMPC



(g) Trajectory predictions at Iteration 30 using MPC



(d) Trajectory predictions at Iteration 40 using SMPC



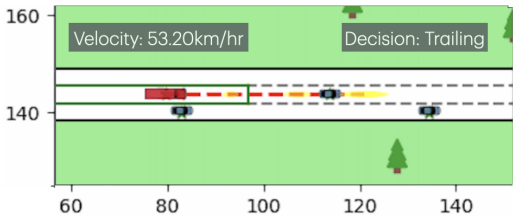
(h) Trajectory predictions at Iteration 40 using MPC

Figure 5.16: Trajectory predictions and decision-making plots for the SMPC framework (left) and MPC framework (right) across four iterations.

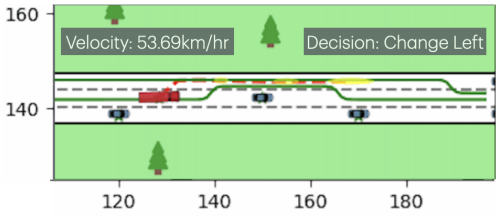
5.5.2 Decision-Making Results from CARLA

Simulation plots based on the results from Figure 5.16are illustrated and compared in Figure 5.17. Figure 5.17 is plotted based on CARLA real-time data, showing

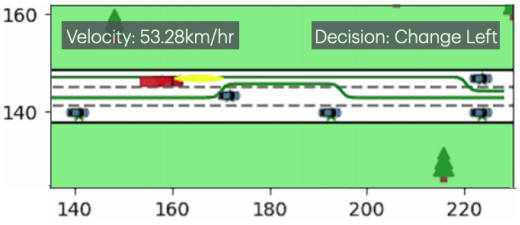
the speed, decision state, and constraints of the ego vehicle in the SMPC and MPC framework.



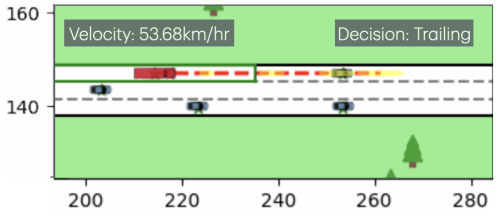
(a) Simulation results at Iteration 10 using MPC framework



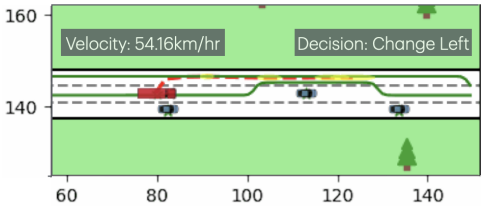
(b) Simulation results at Iteration 20 using MPC framework



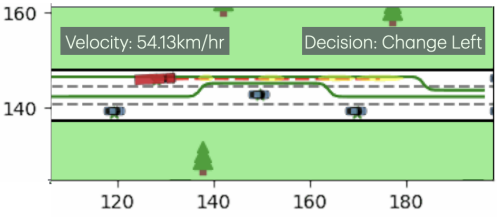
(c) Simulation results at Iteration 30 using MPC framework



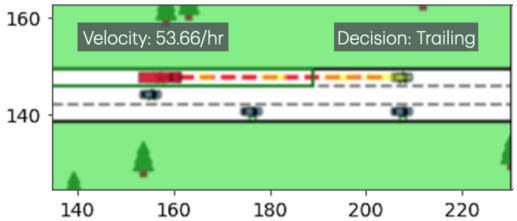
(d) Simulation results at Iteration 40 using MPC framework



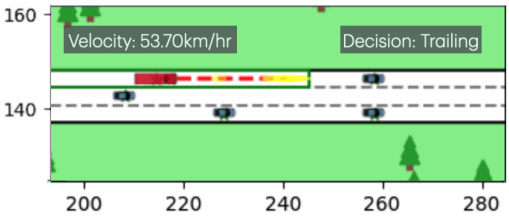
(e) Simulation results at Iteration 10 using SMPC framework



(f) Simulation results at Iteration 20 using SMPC framework



(g) Simulation results at Iteration 30 using SMPC framework



(h) Simulation results at Iteration 40 using SMPC framework

Figure 5.17: Simulation results of trajectory prediction of the ego vehicle at Iterations 10, 20, 30 and 40 for MPC and SMPC frameworks. The red vehicle is the ego vehicle. The green lines around the ego vehicle are the constraints. The predicted trajectory is shown in red dashed lines. The yellow ellipses on the predicted trajectories are the uncertainty propagation of the ego vehicle.

For the MPC framework, in Iteration 10, as shown in Figure 5.17a, the ego vehicle initially follows the traffic car in front of it, its traffic constraints are shown as green

lines, yellow ellipses mark the positional uncertainty, and the future trajectory predicted by the MPC is shown as a red dashed line. In Iteration 20, shown in 5.17b, the DM recalculates and determines the change lanes to the left. In Iteration 30, shown in 5.17c, the ego vehicle is approaching the left lane, and the DM continues to predict the trajectories and chooses to continue to do a left lane change. In Iteration 40, shown in Figure 5.17d, the ego vehicle is now in the left lane and DM chooses to do trailing.

For SMPC Framework, in Iteration 10, shown in Figure 5.17e, the ego vehicle in the middle lane chooses to do a left change. In Iteration 20, shown in Figure 5.17f, the ego vehicle is approaching the left lane and the DM chooses to continue doing the left change. In Iteration 30 and Iteration 40, shown in Figure 5.17c and 5.17d respectively, the ego vehicle is in the left lane and the DM decides to do trailing.

5.6 Performance Evaluation of MPC and SMPC

Figure 5.18 shows the trajectory of SMPC compared to MPC in the y axis direction. The graph shows that SMPC makes a lane change decision about 3 s into the simulation and instructs the local controller to change lanes to the left. This early action is based on the SMPC's tightened constraints, considering the system uncertainty and its prediction of a higher risk of a collision (the possibility of collision should be less than p), resulting in the choice to change lanes earlier to avoid potential risks. In contrast, the conventional MPC made the lane change decision after about 6 seconds.

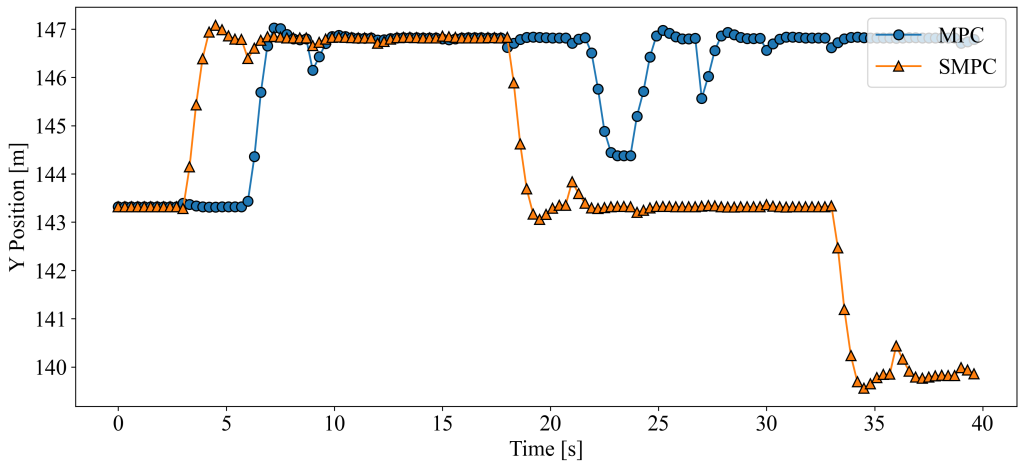


Figure 5.18: Trajectories comparison of SMPC with MPC in the y direction.

Further observations show that the SMPC starts the lane change at 17 seconds and successfully completes it at 19 seconds. In contrast, the MPC does not begin changing lanes until 22 seconds because the controller does not account for system uncertainty. During the lane change process, the vehicle controlled by the MPC collides with surrounding vehicles. Despite the collision, the slack variables introduced in the MPC framework play a crucial role in maintaining the operation. These slack

variables prevent the solver from encountering 'infeasible' errors, which would otherwise cause the control system to fail. By allowing minimal violations of constraints, the slack variables enable the MPC to propose trajectories even when the strict problem is infeasible. Consequently, the ego vehicle is able to continue driving, as the system is not designed to stop upon collision.

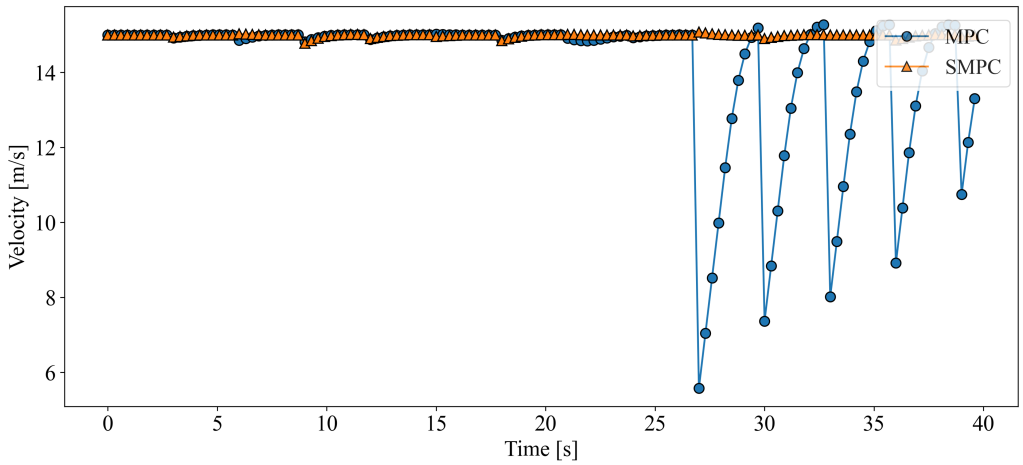


Figure 5.19: Velocity comparison of SMPC with MPC in the x direction.

Figure 5.19 shows the speed comparison between MPC and SMPC. Before the collision, the average speeds of the two control strategies are 14.94 m/s and 14.93 m/s, respectively, which shows that both of them can effectively track the set target speed of 15 m/s and exhibit excellent control performance. However, the figure also shows that at approximately 25 seconds, the speed of the MPC suddenly drops, a phenomenon that is attributed to the occurrence of a collision. This collision causes an urgent need for the system to switch to the left change controller in an attempt to quickly return to the desired speed, which triggers a sharp oscillation in the MPC speed.

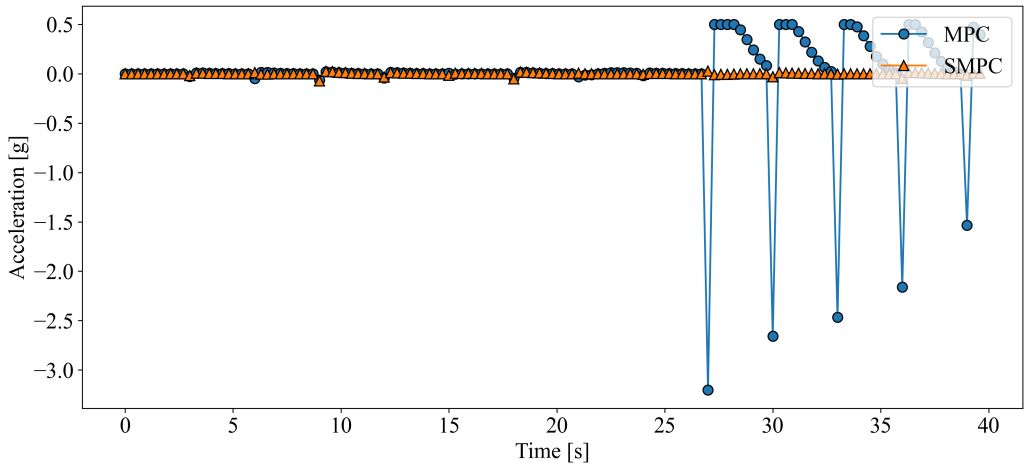


Figure 5.20: Acceleration comparison of SMPC and MPC in the x direction.

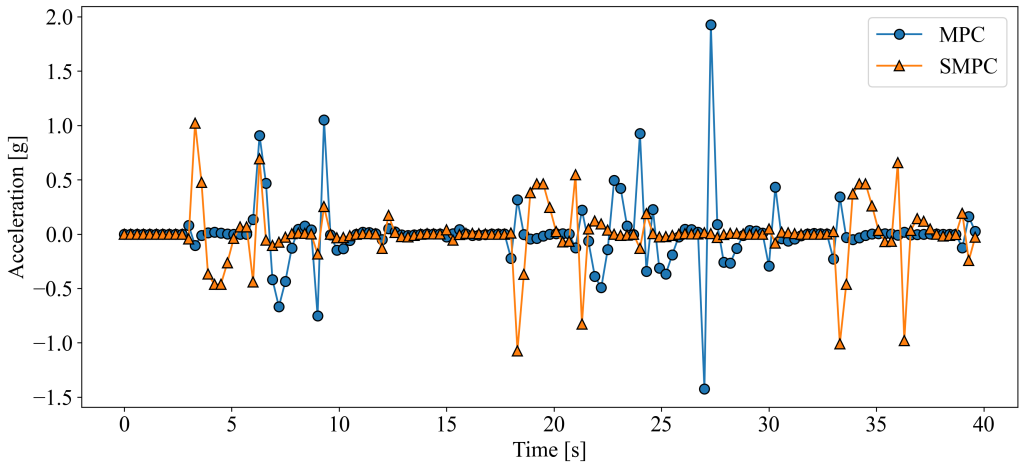


Figure 5.21: Acceleration comparison of SMPC and MPC in the y direction.

For the acceleration performance in the x direction, as shown in Figure 5.20 both control strategies perform very well up to 25 seconds, with the acceleration values remaining almost near 0, indicating that the velocity is very well maintained. However, after 25 seconds, due to the collision, the acceleration of the MPC in the x direction changes drastically, especially at 26 seconds and 30 seconds, where the acceleration values even exceed the constraint limits of the system.

For the lateral acceleration, as shown in Figure 5.21, the MPC shows a sharp acceleration change due to the collision at 25 seconds. The MPC has a strong acceleration change in the y direction. However, even without the collision, the acceleration change of SMPC in the y direction is quite strong.

5.7 Collision Avoidance Testing

The primary purpose of this section is to evaluate the collision avoidance performance of the MPC and the SMPC framework in complex traffic scenarios. The test environment features three lanes, each 170 meters long (spanning from 40 m to 210 m in CARLA Map06). Within this environment, eight vehicles, including the ego vehicle, were randomly generated with speeds ranging from 11.25 m/s to 20 m/s, as detailed in Table 5.2. The initial speed of the ego vehicle was set at 15 m/s. To ensure safety, the distance between any two vehicles in the same lane was maintained at no less than 30 meters.

According to the algorithm 3, all necessary variables are first initialized (Step 1). Next, the surrounding vehicles are randomly generated and their attributes (lane, position, and speed) are stored in the vehicle set $\mathbb{V}$ (Step 2-3). For the y position, it is chosen from the lane set $\mathbb{L}$, where -1 represents the left lane, 0 represents the middle lane, and 1 represents the right lane. For the x position, it is chosen from l_{start} to l_{end} randomly. Afterward, other vehicles are generated cyclically, and the same operation is performed (Step 4-6). To avoid insufficient vehicle spacing during the initial generation, we ensure a minimum distance between all vehicles by an

adjustment loop (Step 7).

Collision detection is performed through a continuous MPC or SMPC loop. In each loop, the speed of each vehicle is checked (Steps 9-14). If the speed of either vehicle is less than 5 m/s, we consider that the vehicle has collided (this speed threshold can be adjusted to reflect different collision determination criteria). At this point, the algorithm sets the collision flag C to true and returns the list of vehicles and the collision status for further analysis.

Algorithm 3 Test Environment and Collision Detection

1: Initialize variables: $\mathbb{L} = \{-1, 0, 1\}$, $l_{\text{end}} = 210$, $l_{\text{start}} = 40$, $d_{\min} = 30$, $v_{\min} = 11.5$, $v_{\max} = 20$, $n_{\text{vehicles}} = 8$, $\mathbb{V} = []$, $C =$ False
2: Generate ego vehicle in random lane and position, set speed to 15 m/s
3: Append ego vehicle to $\mathbb{V}$
4: **for** $i = 1$ to $n_{\text{vehicles}} - 1$ **do**
5: Generate and append other vehicles to $\mathbb{V}$
6: **end for**
7: Ensure minimum distance $d_{\min}$ between vehicles on the same lane
8: **loop** {Iterate over simulation time steps}
9: **if** $t \geq T_{end}$ **then**
10: **return** $\mathbb{V}$, C
11: **end if**
12: **for** v in $\mathbb{V}$ **do**
13: **if** vehicle.speed < 5 **then**
14: $C \leftarrow$ True
15: **return** v, C
16: **end if**
17: **end for**
18: **end loop**

In each operation of the DM, three controllers $j = \{\text{nc, lc, rc}\}$ are solved independently. Due to the large amount of computation involved in the solving process, the solving time is relatively long. In the standard MPC strategy, we find that its average solution time is 0.31 seconds. In contrast, the solution time of SMPC is slightly increased. Especially in complex road conditions, the constraints of SMPC are further tightened, which makes the solution process more complicated and time-consuming, and its average solution time reaches 0.35 seconds. This is because, in the SMPC, the problem becomes more difficult due to the shrinking feasible space, which increases the computational complexity and time. The related data and comparison results are shown in Table 5.11.

The above results show that although SMPC provides tighter control when dealing with complex environments, this correspondingly increases the computational load and the time required for solving.

Table 5.11: Comparison of MPC and SMPC under 100 Randomized Scenarios

Strategy	Collision Count	Success Probability (%)	Average Solution Time (s)
MPC	5	95	0.31
SMPC	1	99	0.35

In the 100 randomized tests we conducted, the self-vehicle with the SMPC strategy collided in very few cases, with a collision rate of only 1% (only 1 collision). In contrast, the self-vehicle with the standard MPC strategy had 5 collisions, with a collision rate of 5%. This data clearly shows that despite the increased solution time, SMPC significantly outperforms standard MPC in terms of safety, especially when dealing with complex and variable road conditions.

6
Discussion

In this chapter, we will discuss the main findings and implications of the research in this paper, focusing on analyzing the differences in the performance of MPC and SMPC in decision-making, performance evaluation, and collision avoidance testing. By comparing the decision-making behaviors and control effects of these two control frameworks in different traffic scenarios, we can identify the specific challenges and room for improvement of MPC and SMPC when dealing with complex traffic environments. This will help to select more appropriate optimization tools and algorithms in future research to further enhance the reliability and safety of autonomous driving systems.

6.1 Decision-Making in MPC and SMPC

The scenarios shown in the Figure 5.16 clearly depict the differences of SMPC and MPC frameworks in the decision-making, trajectory predictions and the constraint formulation. In Iteration 10, for SMPC and MPC framework, shown in Figure 5.16a and 5.16e respectively, the DM evaluates the trajectory predictions for all the three controllers as the ego vehicle is in the middle lane, as explained in Section 2.4. From the Table 5.7 and 5.9, the trailing controller cost at Iteration 10 for the SMPC framework is 2565.15, while it is only 8.77 for the MPC framework. This higher trailing controller cost for SMPC is due to the tightening of the trailing constraints to account for the uncertainty.

Although a left lane change requires switching controllers and is more expensive, the fact that the maximum speed can be maintained given that there are no vehicles in the left lane makes the left turn a much less costly option. In the SMPC framework, the cost of changing lanes to the left is 935.41, which makes the left turn a more economical decision in comparison. This cost-based decision-making is a result of SMPC's ability to excel at handling uncertainty and managing risk, resulting in the decision to perform a left lane change in the Iteration 10, rather than choosing to follow the car as observed in the MPC framework.

Comparing the trajectory predictions for the right change controller in Iteration 10 for SMPC and MPC frameworks, the SMPC one (doRight in Figure 5.16a) is closer to the center line of the middle lane than the MPC one (doRight in Figure 5.16e). This is due to the tightening of the constraints for SMPC controllers, which is also

evident from their corresponding controller cost. From Table 5.7 and 5.9, the controller cost for SMPC is 4446.31, while for MPC it is only 3589.43.

As explained in Section 2.4, the right change controller is inactive for the Iteration 20 for the SMPC framework. With a traffic vehicle always present in the middle lane, the constraints for the right change controller are guaranteed to be violated, resulting in a high slack cost if the controller were active. Therefore, the right lane change controller will be inactive and thereby also improving the computational time. This is also seen in Iteration 30 for the MPC framework, shown in Figure 5.16g.

In Iteration 20 for the SMPC framework, the ego vehicle is still approaching the left lane, and the conditions for the reference update, as depicted in Figure 2.11, have not yet been met. Thus, the left change and trailing controllers continue to predict trajectories with the center line of the left and middle lanes as their respective references. Similar decision behavior is observed in Iteration 30 for MPC framework, as shown in Figure 5.16g.

In Iteration 30 for SMPC framework, as shown in Figure 5.16c, the trajectory predictions from the left change and trailing controllers both target the center line of the left lane. Here, the reference update for the trailing controller has been completed, while the left change controller still follows the previous reference. Notably, the ego vehicle has not yet crossed the center line of the left lane, which is necessary for the reference update for the left change controller, as explained in Figure 2.11. A similar behavior is also seen in MPC framework, as shown in the Figure 5.16h.

By Iteration 40 in the SMPC framework, the conditions for the reference update are met and applied to both the left change and trailing controllers. Given that the controller cost for the trailing controller is significantly lower compared to the left change controller, the ego vehicle decides to follow the trailing controller in the left lane.

6.2 Performance Evaluation of MPC and SMPC

As shown in Figures 5.18 and 5.19, it is clear that SMPC has a significant advantage over MPC in improving driving safety, especially in complex traffic scenarios. This is not only reflected in the vehicle's speed maintenance and collision avoidance capabilities but also supported by the longitudinal acceleration stability performance (Figure 5.20).

In terms of acceleration performance in the x direction, the SMPC is always very stable because its control algorithm tightens the constraints strictly, thus effectively avoiding collisions. In contrast, the acceleration in the x direction of the MPC shows serious abrupt changes, even exceeding the system-limited acceleration value. The frequency of this acceleration anomaly is consistent with the operating frequency of the MPC. The reasons for this phenomenon are the drastic changes in the traffic

environment after the collision and the controller forcing the ego vehicle to adopt extreme acceleration or deceleration strategies. In addition, the real-time CARLA simulation is still performing acceleration and deceleration motions during the MPC computation. This portion of the MPC computation time is not taken into account when calculating the acceleration in the x direction, which results in acceleration values that exceed the system limits.

For acceleration in the y direction, both MPC and SMPC do not perform well, and even when SMPC avoids collisions, its performance in the y direction is not very good. This is mainly due to the fact that the lateral control essentially employs a PI controller. The long sampling time in this paper leads to a lag in the response of the controller, and the integrator accumulates a large error, resulting in too large an output, thus affecting the stability and control accuracy of the system. Another reason is that with frequent changes in the set point, the integrator term accumulates large historical errors, which do not match the actual control requirements, resulting in a control output that is off target.

In addition, if the parameters of the local controllers, especially the lateral controller, are adjusted, the aggressiveness of the MPC during lane changing can be enhanced, leading to effective obstacle avoidance. Although this approach is effective in specific scenarios, it is only applicable to known specific scenarios, and this scenario-dependent tuning deviates from the original intent of the design of the autonomous driving system. In fact, this approach is more of a stopgap measure for a specific situation than an ideal systematic strategy.

6.3 Collision Avoidance Testing of MPC and SMPC

According to table 5.11, although SMPC requires longer computation time in dealing with nonlinear problems, which may lead to a slightly delayed response time of the control system. This increase in time may affect the responsiveness and efficiency of the control system in real-time application scenarios. However, thanks to its tightened constraints, the ego vehicle is still able to maintain a sufficiently safe distance in more complex road environments, thus effectively avoiding most potential collisions. In addition, this strategy shows significant advantages, especially in the management of system uncertainty. System uncertainty itself is a major challenge for the ego vehicle to avoid collisions. With the implementation of the SMPC strategy, these uncertainties are more effectively approached, thus enhancing the stability and safety of the overall system.

6.4 Future Work

In the study of this paper, the MPC-based framework is severely limited by the solution time. This is due to the fact that in complex traffic scenarios, the constraints

become very complex and multiple optimal control problems often need to be solved, while the currently used CasADi solver is not very fast. If possible, in the future, it is desired to solve such optimization problems using ACADOS [34], which has a higher computational efficiency and can solve complex optimization problems in a shorter time.

Another possible approach is to enable the model to learn and handle these complex traffic situations by means of reinforcement learning [35]. Reinforcement learning has advantages in dealing with dynamic environments and complex decision-making problems, and it is expected to significantly improve the system's adaptive ability and responsiveness by training the model to cope with different traffic conditions.

In addition, due to the constraints of the solution time, this paper can only design the surrounding vehicles to move in a straight line, otherwise the increase of the solution time will likely lead to collisions. Therefore, this paper has not been able to realistically simulate the actual road conditions. If ACADOS or reinforcement learning methods are applied, it is expected that in the future, the surrounding vehicles can be designed to drive according to actual road conditions, including complex behaviors such as lane changing, so as to more realistically simulate the actual driving environment.

Finally, since only truck models such as fire trucks are provided in CARLA and not trailer models, the vehicle modeling in this paper is only based on a single-degree-of-freedom kinematic model. If the trailer model in CARLA can be obtained in the future, the model can be extended based on this paper to simulate and validate the trailer in CARLA. This will help to more comprehensively evaluate and optimize the performance of the trailer autopilot system in complex traffic environments.

7
Conclusion

This thesis presented a comprehensive framework for autonomous vehicles using SMPC. The purpose of the thesis was to design and implement a robust control system that is capable of effective decision-making, trajectory prediction and collision avoidance in dense traffic scenarios. The research incorporated detailed vehicle dynamics modeling, constraint formulation, and optimization solver integration within the CARLA simulation environment. The main research questions of this thesis consist of the following:

1. How to build a safety-critical scenario for the heavy vehicle in CARLA and make it fit the realistic scenario?
2. How can the challenges associated with the uncertainty in the vehicle's propagation between the vehicle model and the CARLA simulation environment be addressed?
3. How can the integration of a stochastic control with the traditional MPC improve the robustness of the autonomous heavy vehicle operating in uncertain environments?
4. How does the performance of SMPC compare to traditional MPC in terms of safety and efficiency in high-fidelity simulation environments like CARLA?

An appropriate safety-critical scenario was presented in CARLA to fit the realistic scenario. Using CARLA as a simulation environment proved to be highly effective in doing research in the autonomous driving field. In the CARLA environment, complex traffic conditions are created on three lanes of a highway to validate the algorithms.

In the CARLA scenario, information about the ego vehicle is obtained by using the Kalman Filter as an observer. The steady state covariance of the Kalman Filter is utilized as the initial uncertainty to analyze the propagation of uncertainty with time in a closed-loop system. Based on SMPC theory, the constraints of the system are tightened to approach the effect of uncertainty on the system. Based on the tightened constraints, the optimal control problem is reconstructed, and the control architecture is updated to output a new optimal trajectory.

Furthermore, the performance of the SMPC with the traditional MPC was evaluated in CARLA. It is shown that by tightening the constraints, the SMPC-based approach can significantly improve the safety and stability of autonomous heavy vehicles, ensuring that a safe distance is maintained and collisions are effectively avoided. The simulation results confirm that the SMPC outperforms the traditional MPC in terms of driving safety.

The study also identified several limitations related to the CARLA simulation environment and computational efficiency. Even though the current use of integration solver CasADi has proved to be effective, it poses several limitations in computation time, especially in a dense traffic scenario. Future work to explore the integration of faster solvers like ACADOS to enhance computational efficiency was laid out. Additionally, machine learning techniques present a promising avenue to improve the system's adaptive capabilities in handling diverse traffic conditions. This thesis thus provides a good groundwork for further research in the field of autonomous driving.

By addressing these research questions and challenges, this thesis contributes to advancing the robustness, safety, and efficiency of autonomous vehicles, particularly in uncertain and dynamic environments.

Bibliography

[1] JI Yangjie, ZHOU Zewei, YANG Ziru, Yanjun HUANG, Yuanjian ZHANG, Wanting ZHANG, Lu XIONG, and YU Zhuoping. Towards autonomous vehicles: a survey on cooperative vehicle-infrastructure system. *iScience*, 2024.

[2] Heather E Rosen, Imran Bari, Nino Paichadze, Margaret Peden, Meleckidzedeck Khayesi, Jesús Monclús, and Adnan A Hyder. Global road safety 2010–18: an analysis of global status reports. *Injury*, 2022.

[3] A Haghi, D Ketabi, M Ghanbari, and H Rajabi. Assessment of human errors in driving accidents; analysis of the causes based on aberrant behaviors. *Life Science Journal*, 11(9):414–420, 2014.

[4] Đorđe Petrović, Radomir Mijailović, and Dalibor Pešić. Traffic accidents with autonomous vehicles: type of collisions, manoeuvres and errors of conventional vehicles' drivers. *Transportation research procedia*, 45:161–168, 2020.

[5] Stefano Arrigoni, Francesco Braghin, and Federico Cheli. Mpc trajectory planner for autonomous driving solved by genetic algorithm technique. *Vehicle system dynamics*, 60(12):4118–4143, 2022.

[6] Álvaro Carrizosa-Rendón, Vicenç Puig, and Fatiha Nejjari. Safe motion planner for autonomous driving based on lpv mpc and reachability analysis. *Control Engineering Practice*, 147:105932, 2024.

[7] Ivo Batkovic, Ankit Gupta, Mario Zanon, and Paolo Falcone. Experimental validation of safe mpc for autonomous driving in uncertain environments. *IEEE Transactions on Control Systems Technology*, 2023.

[8] Xinchen Li, Levent Guvenc, and Bilin Aksun-Guvenc. Vehicle state estimation and prediction for autonomous driving in a round intersection. *Vehicles*, 5(4):1328–1352, 2023.

[9] Juraj Kabzan, Miguel I Valls, Victor JF Reijgwart, Hubertus FC Hendrikx, Claas Ehmke, Manish Prajapat, Andreas Bühler, Nikhil Gosala, Mehak Gupta, Ramya Sivanesan, et al. Amz driverless: The full autonomous racing system. *Journal of Field Robotics*, 37(7):1267–1294, 2020.

[10] Keisuke Fujii. Extended kalman filter. *Refernce Manual*, 14:41, 2013.

[11] Hugh Durrant-Whyte et al. Introduction to estimation and the kalman filter. *Australian Centre for Field Robotics*, 28(3):65–94, 2001.

[12] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016.

[13] Guoqiang Li, Xudong Zhang, Hongliang Guo, Basilio Lenzo, and Ningyuan Guo. Real-time optimal trajectory planning for autonomous driving with col-

lision avoidance using convex optimization. *Automotive Innovation*, 6(3):481–491, 2023.

[14] Zhenfeng Li, Xuncheng Wu, Weiwei Zhang, and Wangpengfei Yu. Obstacle avoidance trajectory planning for autonomous vehicles on urban roads based on gaussian pseudo-spectral method. *World Electric Vehicle Journal*, 15(1):7, 2023.

[15] David Q Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014.

[16] KS Holkar and Laxman M Waghmare. An overview of model predictive control. *International Journal of control and automation*, 3(4):47–63, 2010.

[17] Erik Börve. Autonomous truck sim. `https://github.com/BorveErik/Autonomous-Truck-Sim`, 2023.

[18] D. Limon, T. Alamo, F. Salas, and E.F. Camacho. On the stability of constrained mpc without terminal constraint. *IEEE Transactions on Automatic Control*, 51(5):832–836, 2006.

[19] Johan Karlsson, Nikolce Murgovski, and Jonas Sjöberg. Optimal trajectory planning and decision making in lane change maneuvers near a highway exit. In *2019 18th European Control Conference (ECC)*, pages 3254–3260. IEEE, 2019.

[20] Pan Zhao, Jiajia Chen, Yan Song, Xiang Tao, Tiejuan Xu, and Tao Mei. Design of a control system for an autonomous vehicle based on adaptive-pid. *International Journal of Advanced Robotic Systems*, 9(2):44, 2012.

[21] Dario D Salvucci and Rob Gray. A two-point visual control model of steering. *Perception*, 33(10):1233–1248, 2004.

[22] Davide Martino Raimondo, Daniel Limon, Mircea Lazar, Lalo Magni, and Eduardo Ferná ndez Camacho. Min-max model predictive control of nonlinear systems: A unifying overview on stability. *European Journal of Control*, 15(1):5–21, 2009.

[23] Jianshan Zhou, Daxin Tian, Zhengguo Sheng, Xuting Duan, Guixian Qu, Dezong Zhao, Dongpu Cao, and Xuemin Shen. Robust min-max model predictive vehicle platooning with causal disturbance feedback. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):15878–15897, 2022.

[24] Yong Dai and Duo Wang. A tube model predictive control method for autonomous lateral vehicle control based on sliding mode control. *Sensors*, 23(8):3844, 2023.

[25] James Blake Rawlings, David Q Mayne, Moritz Diehl, et al. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI, 2017.

[26] Tim Brüdigam, Michael Olbrich, Dirk Wollherr, and Marion Leibold. Stochastic model predictive control with a safety guarantee for automated driving. *IEEE Transactions on Intelligent Vehicles*, 8(1):22–36, 2021.

[27] Marcello Farina, Luca Giulioni, and Riccardo Scattolini. Stochastic linear model predictive control with chance constraints–a review. *Journal of Process Control*, 44:53–67, 2016.

[28] Ashwin Carvalho, Yiqi Gao, Stéphanie Lefevre, and Francesco Borrelli. Stochastic predictive control of autonomous vehicles in uncertain environments. In *12th international symposium on advanced vehicle control*, volume 9, 2014.

[29] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[30] CARLA Team. CARLA simulator documentation. `https://carla.readthedocs.io/en/latest/`. Accessed: May 4, 2024.

[31] Arsalan Haider, Yongjae Cho, Marcell Pigniczki, Michael H Köhler, Lukas Haas, Ludwig Kastner, Maximilian Fink, Michael Schardt, Yannik Cichy, Shotaro Koyama, et al. Performance evaluation of mems-based automotive lidar sensor and its simulation model as per astm e3125-17 standard. *Sensors*, 23(6):3113, 2023.

[32] Vladimir Suvorkin, Miquel Garcia-Fernandez, Guillermo González-Casado, Mowen Li, and Adria Rovira-Garcia. Assessment of noise of mems imu sensors of different grades for gnss/imu navigation. *Sensors*, 24(6):1953, 2024.

[33] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[34] Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. acados – a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 2021.

[35] Ahmad Reda and József Vásárhelyi. Design and implementation of reinforcement learning for automated driving compared to classical mpc control. *Designs*, 7(1):18, 2023.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY