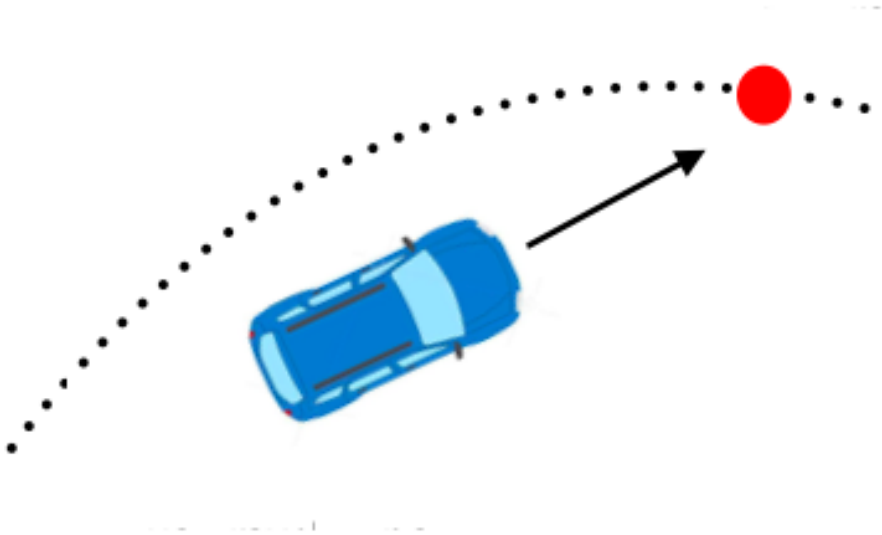
CHALMERS
UNIVERSITY OF TECHNOLOGY



Adaptive Path Following Driver Model

Master's thesis in Mobility Engineering

Balaji Sathiya Venkata Narayanan
Muralikrishna Manickam

DEPARTMENT OF MECHNICS AND MARITIME SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

Adaptive Path Following Driver Model

Balaji Sathiya Venkata Narayanan
Muralikrishna Manickam

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Chalmers University of Technology
Gothenburg, Sweden 2025

Adaptive Path Following Driver Model

Balaji Sathiya Venkata Narayanan, Muralikrishna Manickam

Supervisor: Holger lindstrom, Volvo Car Corporation

Examiner: Fredrik Bruzelius, Department of Mechanics and Maritime Sciences, Chalmers University of Technology

Master's Thesis 2025
Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of trajectory tracking: Highlighting deviation between the desired trajectory and the actual traveled path.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

Adaptive Path Following Driver Model
Balaji Sathiya Venkata Narayanan
Muralikrishna Manickam
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

The evolution of advanced driver assistance systems (ADAS) and autonomous driving technologies has heightened the need for robust and adaptive driver models. This thesis focuses on developing an adaptive driver model within a Software-in-the-Loop (SIL) framework, designed to handle dynamic environments, complex scenarios, and disturbances with high precision.

A state-space model is formulated to capture vehicle dynamics and error dynamics, essential for precise trajectory tracking. The error dynamics state-space model updates in real time, accounting for deviations in lateral position, yaw angle, and other key variables. This real-time updation enables the model to compute optimal control inputs using both a Linear Quadratic Regulator (LQR)-based controller and a Model Predictive Control (MPC)-based approach. MPC, with its ability to anticipate future states and optimize control inputs over a finite horizon, complements LQR by providing enhanced performance in managing constraints and nonlinearities, especially in dynamic environments. The SIL framework integrates real-time data exchange between components, leveraging middleware to maintain simulation fidelity and responsiveness.

By iteratively refining error dynamics, adapting to changes in each simulation setup, and leveraging both LQR and MPC for trajectory tracking, the proposed driver model enhances precision and adaptability. This research contributes to advancing SIL frameworks, supporting safer and more reliable autonomous driving technologies while meeting industry standards.

Keywords: AD & ADAS, Adaptive driver model, Path following, Software-in-the-Loop, Error Dynamics, State-space model, Model Predictive Control, Linear Quadratic Regulator, Trajectory Tracking, Lateral control, Real-time Control, LQR Controller Tuning, Dynamic Environments, Vehicle Dynamics, Middleware Integration, Tracking accuracy, MPC.

Acknowledgements

We would like to express our sincere gratitude to Holger Lindstrom, our supervisor at Volvo Cars Corporation, for his expert advice and unwavering support throughout this project. We are also deeply thankful to Fredrik Bruzelius, our supervisor and examiner at Chalmers University of Technology, for his valuable feedback during our meetings and for guiding us through the challenges of this work.

Our heartfelt thanks go to Francesco Costagliola, our manager at Volvo Cars, for his flexibility in accommodating our requirements and for his continuous support during the project. We are immensely grateful for all the academic support, resources, and opportunities provided by Chalmers University of Technology and Volvo Cars Corporation, which were instrumental in completing this thesis.

Finally, we extend special thanks to our family and friends for their love and encouragement, which have been invaluable throughout this journey. This project would not have been the same without the guidance and support we received from all those involved.

Balaji Sathiya Venkata Narayanan & Muralikrishna Mancikam, Gothenburg, February, 2025

List of Acronyms

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

AV	Autonomous Vehicles
SAE	Society of Automotive Engineers
AD	Autonomous Driving
ADAS	Advanced Driver Assistance Systems
SIL	Software-in-the-Loop
PID	Proportional-Integral-Derivative
LQR	Linear Quadratic Regulator
MPC	Model Predictive Control
AMPC	Adaptive Model Predictive Control
RL	Reinforcement Learning
SMC	Sliding Mode Control
IMC	Internal Model Control
ART	Advanced Research & Technology
SVA	Solution of Safe Vehicle Automation
VCC	Volvo Cars Corporation
CoG	Center of Gravity
DDS	Data Distribution Service
DOF	Degrees of Freedom
ARE	Algebraic Riccati Equation
DARE	Discrete Algebraic Riccati Equation
DLC	Double Lane Change
CC	Constant Curvature

Nomenclature

Below is the nomenclature of variables that have been used throughout this thesis.

Variables

Ego Vehicle	The vehicle of Interest to which the controller input is fed for real time feedback
Ghost Vehicle	The vehicle serve as reference by which the reference trajectory is generated.
x	Longitudinal position of the Ego Vehicle
y	Lateral position of the Ego Vehicle
θ	Heading angle of the Ego Vehicle
v	Longitudnal Velocity
L	wheel base
v_x	Longitudinal velocity of the Ego Vehicle
v_y	Lateral velocity of the Ego Vehicle
ψ	Yaw angle of the Ego Vehicle
$\dot{\psi}$	Yaw rate of the Ego Vehicle
δ	Steering angle
e_{ct}	Cross-track error
e_{ψ}	Heading error
$\dot{e}_{ct}$	Rate of change of cross-track error
$\dot{e}_{\psi}$	Rate of change of heading error
F_y	Lateral force acting on tires
F_{yf}	Lateral force acting on the front tires
F_{yr}	Lateral force acting on the rear tires
m	Mass of the vehicle
I_z	Moment of inertia about the vertical axis

L_f	Distance from the center of gravity to the front axle
L_r	Distance from the center of gravity to the rear axle
C_f	Cornering stiffness of the front tires
C_r	Cornering stiffness of the rear tires
α_f	Wheel Slip Angle of the front tires
α_r	Wheel Slip Angle of the rear tires
u	Control input (steering or acceleration)
Δt	Simulation time step
e_y	Lateral deviation from the path (cross-track error)
f_e	Front axle error
k_v	Speed-dependent gain
θ_e	Heading error
$\dot{\theta}_r$	Reference Yaw rate k_s
Stanley controller gain for cross-track error response	
ϵ	Small constant to prevent division by zero in Stanley control law
Q	State weighting matrix in LQR cost function
R	Control input weighting matrix in LQR cost function
K	Optimal gain matrix in LQR
J	Cost function minimized by LQR
$u(t)$	Control output at time t (e.g., throttle or steering)
$e(t)$	Error signal at time t
K_p	Proportional gain in PID controller
K_i	Integral gain in PID controller
K_d	Derivative gain in PID controller
q_1, q_2, q_3, q_4	Weighting factors in the LQR cost function
q_5	Control effort weight in LQR
$\mathbf{A}, \mathbf{B}, \mathbf{C}$	State-space matrices
$\mathbf{P}$	Solution to the Riccati equation
$\mathbf{K}$	Feedback gain matrix in LQR
ρ	Path curvature
δ_{ff}	Feedforward steering angle
δ_f	Total steering angle

T_{sam}	Sampling time
dla	distancle look ahead

Contents

List of Figures

List of Tables

1
Introduction

Autonomous vehicles (AVs), commonly referred to as self-driving cars, are reshaping the transportation sector with their potential to enhance safety, improve traffic efficiency, and provide accessible mobility solutions. These vehicles operate without human intervention by leveraging technologies such as sensor fusion, computer vision, and artificial intelligence. Significant progress in machine learning, robotics, and computational power has driven advancements in this domain.[1].

The Society of Automotive Engineers (SAE) categorizes autonomous driving into six levels, ranging from Level 0 (no automation) to Level 5 (full automation). Autonomous vehicles employ an array of sensors—such as cameras, radar, and lidar—working in tandem with sophisticated software algorithms. These enable AVs to perceive their surroundings, make decisions, and execute driving tasks with remarkable precision, ensuring operational safety and efficiency.[2].

1.1 Autonomous Vehicles

Autonomous vehicles (AVs) are rapidly transforming the landscape of modern transportation by leveraging cutting-edge technologies to navigate safely and efficiently. This section explores the advancements, challenges, and models associated with autonomous vehicles, particularly focusing on path-following techniques crucial for their functionality.

1.1.1 The Significance of Path Following for Autonomous Vehicles

Path following is an essential component of autonomous navigation, ensuring that vehicles adhere to predefined or dynamically generated trajectories. This ability significantly contributes to passenger safety and vehicle performance by mitigating risks such as collisions, road departures, and inefficient navigation. Reliable path-following systems play a pivotal role in achieving operational efficiency and safety in autonomous vehicles.

Despite its importance, path following poses considerable challenges. These include handling uncertainties in sensor data, adapting to changing road conditions, and responding promptly to dynamic obstacles. Effective path-following systems not only ensure technical robustness but also address environmental concerns by optimizing

fuel consumption and reducing emissions and passenger discomfort.[3].

1.1.2 Path Following Driver Models

Driver models simulate human behavior to enable autonomous vehicles to track desired trajectories accurately. These models are classified into three primary categories:[3]

- **Geometric Models:** Focus on spatial relationships between the vehicle and the path.
- **Kinematic Models:** Incorporate vehicle motion characteristics such as velocity and acceleration.
- **Dynamic Models:** Account for tire forces, suspension systems, and the interaction between the vehicle and the road.

Dynamic models are particularly well-suited for advanced applications due to their comprehensive nature, making them essential for replicating complex real-world conditions[4].

1.1.3 Adaptive Path Following Driver Models

While traditional models provide adequate solutions under controlled conditions, real-world scenarios demand adaptive systems that can respond dynamically to complex environments. Adaptive path-following driver models integrate real-time sensor feedback and environmental data to adjust vehicle control parameters dynamically. These models employ online learning and optimization techniques to handle obstacles, uncertainties, and rapid environmental changes effectively.[5][6].

Such adaptability is crucial for ensuring operational safety and efficiency, especially in unpredictable scenarios where static models fail to perform adequately. Adaptive models represent a significant advancement in the field, paving the way for safer and more reliable autonomous vehicles.

In the subsequent sections of this report, we will delve deeper into the technical aspects of adaptive path following driver models, exploring their mathematical formulations, implementation strategies, and experimental evaluations.

1.2 Intro to Path Following Driver Model

The Path Following Driver Model developed in this thesis is designed to enable precise trajectory tracking for an autonomous vehicle, referred to as the Ego Vehicle. The Ego Vehicle represents the primary vehicle of interest in this research, tasked with following a trajectory generated by another entity termed the Ghost Vehicle.

The Ghost Vehicle serves as a virtual guide or reference, simulating a predefined or dynamically generated path. This trajectory may be produced based on a variety of inputs, such as planned routes, real-time road conditions, or desired driving behaviors. By doing so, the Ghost Vehicle acts as a benchmark for the Ego Vehicle to follow, ensuring adherence to the intended path. The trajectory provided by the Ghost Vehicle includes spatial and temporal data, such as position, heading, and speed, which the Ego Vehicle continuously tracks and adjusts to in real time.

To test and refine the Path Following Driver Model, a controlled simulation environment has been employed. This setup enables iterative development, allowing for the evaluation of the Ego Vehicle's performance under diverse and complex scenarios. The simulation environment not only facilitates safe testing but also provides flexibility in adjusting parameters and configurations, thereby supporting the optimization of the driver model.

This approach allows the Path Following Driver Model to be evaluated for accuracy, adaptability, and computational efficiency, ensuring its suitability for real-world applications in autonomous vehicles.

1.2.1 Simulation Environment Configuration

To simulate the Path Following Driver Model, a comprehensive simulation environment is set up, which includes the following components:

1.2.1.1 Simulation Software: esmini OpenSCENARIO Player

The simulation employs **esmini**, a lightweight, open-source platform for testing and validating driving scenarios. esmini enables the creation of complex traffic situations, incorporating diverse environmental conditions and sensor behaviors to evaluate autonomous driving algorithms comprehensively[7].

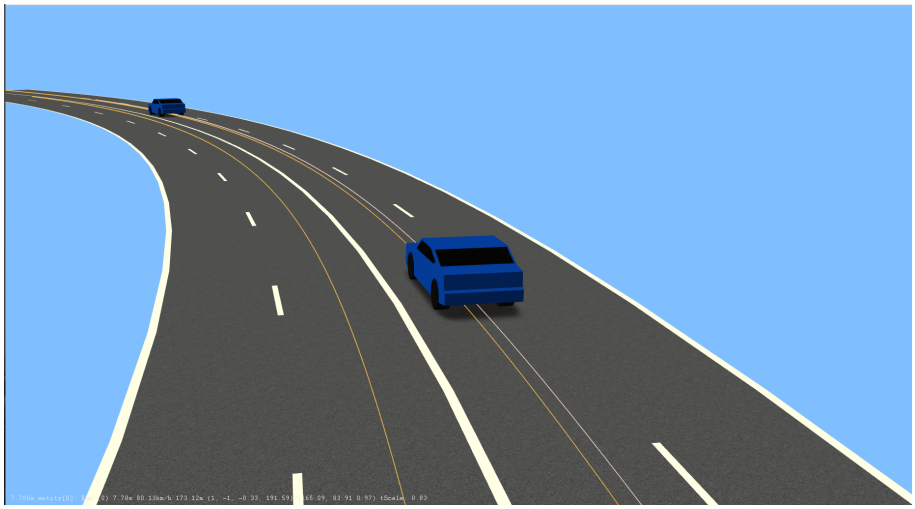


Figure 1.1: Simulation environment using esmini OpenSCENARIO Player showcasing Ego and Ghost vehicles on a predefined trajectory.

As shown in Figure 1.1, the simulation environment illustrates the Ego Vehicle (vehicle of interest) following a trajectory provided by the Ghost Vehicle. The trajectory lines demonstrate the desired path, enabling performance evaluation of the Path Following Driver Model.

esmini will be discussed more in detail in 2.3.

1.2.1.2 Vehicle Model

A Dyanmic model with 7-DOF is employed to represent the vehicle's dynamics and kinematics.[4] This includes vehicle's motion equations to focus on lateral dynamics, such as steering behavior and side-slip angles. The model captures essential aspects of vehicle behavior, such as:

- **Dynamics**: The forces and moments acting on the vehicle, including those generated by steering, acceleration, and braking.
- **Kinematics**: The vehicle's motion in terms of position, velocity, and acceleration, both longitudinally and laterally.

By utilizing this model, the simulation can accurately predict the vehicle's response to various control inputs and external conditions, facilitating the development and testing of control algorithms. Incorporating additional degrees of freedom into the vehicle model enhances its realism, allowing for more comprehensive and accurate simulations.

1.2.1.3 Driver Models

Driver models play a pivotal role in enabling autonomous vehicles to follow a desired path by generating real-time control inputs for steering, acceleration, and braking. These models process trajectory information, such as position, heading, and speed, to ensure the vehicle adheres to the path accurately. Path-following driver models can be broadly categorized into classical control models, optimization-based models, and adaptive models. Each model addresses specific challenges related to accuracy, robustness, and computational efficiency [4, 13].

- **Classical Control Models:** Classical controllers provide straightforward solutions for path-following and are widely used due to their simplicity and computational efficiency:
  - **Proportional-Integral-Derivative (PID) Controller:** The PID controller is employed for longitudinal control by adjusting the vehicle's acceleration and braking to minimize deviations from the target velocity. Its simplicity makes it ideal for structured environments, though it can struggle with external disturbances [14].
  - **Stanley Controller:** The Stanley Controller is a geometric-based lateral control algorithm that minimizes cross-track error and heading error

by adjusting the steering angle. It is particularly effective in structured environments with predictable curvatures and is used as a baseline in this thesis [15].

- **Optimization-Based Models:** Optimization-based models compute control inputs by minimizing a cost function, ensuring smooth and efficient path-following while considering system constraints:

  - **Linear Quadratic Regulator (LQR):** LQR uses a cost function to optimize steering control by minimizing cross-track error, heading deviation, and control effort. Its robustness and simplicity make it suitable for vehicle dynamics control in structured environments [4].

  - **Model Predictive Control (MPC):** MPC predicts the future states of the vehicle over a receding time horizon and generates control inputs that minimize a defined cost function. Its ability to handle dynamic constraints and optimize control actions makes it well-suited for complex and dynamic environments [13].

- **Adaptive and Learning-Based Models:** Adaptive models dynamically adjust their parameters in real time to improve performance under uncertain and varying conditions. Learning-based methods provide additional flexibility in handling complex scenarios:

  - **Adaptive Model Predictive Control (AMPC):** AMPC extends traditional MPC by continuously updating its internal model to account for changes in vehicle dynamics and environmental conditions, improving adaptability and accuracy [16].

  - **Reinforcement Learning-Based Models:** Reinforcement learning (RL) algorithms enable autonomous vehicles to learn optimal control policies through interaction with their environment. RL-based models can handle complex and unpredictable scenarios that traditional methods may struggle with [17].

  - **Robust Control Models:** Robust controllers, such as Sliding Mode Control (SMC) and Internal Model Control (IMC), are designed to maintain performance in the presence of model uncertainties and external disturbances, ensuring reliable path following [18].

Each of these driver models offers unique advantages depending on the operating conditions and complexity of the driving environment. Classical control models like PID and Stanley provide computational efficiency and ease of implementation, making them ideal for simple, structured scenarios. In contrast, optimization-based methods such as LQR and MPC offer improved accuracy and robustness, particularly in dynamic environments. Adaptive and learning-based models provide ad-

ditional flexibility, enabling the vehicle to adapt to uncertainties and unforeseen conditions in real time.
The Selection of driver models will be further discussed in section 2.4

1.2.1.4 Benefits of Simulation

Simulating the Path Following Driver Model offers several advantages:

- **Controlled Environment**: Allows testing under a wide range of conditions without the risks associated with real-world testing.
- **Flexibility**: Allows for easy modification of the vehicle model, control algorithms, and test scenarios.
- **Cost-Effectiveness**: Reduces the need for physical prototypes and extensive field tests.

1.3 Motivation

Autonomous vehicles are poised to revolutionize the transportation industry, promising enhanced safety, efficiency, and convenience. However, for these vehicles to operate reliably and gain public trust, they must be capable of navigating complex and dynamic environments with utmost precision and adaptability.

One of the critical challenges in autonomous vehicle control is accurate path following. Precise adherence to a predefined or dynamically generated path is essential for ensuring safe and efficient navigation. Even minor deviations from the intended path can have severe consequences, such as collisions with obstacles, veering off the road, or causing discomfort to passengers.

The ability to adapt to sudden changes in the environment is equally crucial. Autonomous vehicles must be able to react promptly and appropriately to unexpected obstacles, varying road conditions, and the presence of other vehicles. Failure to do so can compromise safety and undermine the overall performance of the autonomous system.

Existing path-following algorithms have shown promising results in controlled environments, but their performance may be inadequate when faced with the complexities and uncertainties of real-world scenarios. These algorithms often struggle to handle dynamic obstacles and rapidly changing road conditions, limiting their applicability in autonomous vehicle systems.

To address this challenge, there is a pressing need for an adaptive and robust path-following driver model that can excel in diverse and unpredictable testing scenarios. Such a model would not only ensure the safety and efficiency of autonomous vehicles but also contribute to building public confidence in this emerging technology.

By exploring and evaluating different path-following algorithms through comprehensive simulations and testing, our research aims to identify the most suitable approach for developing an adaptive driver model. The insights gained from this analysis will contribute to advancing the state-of-the-art in autonomous vehicle control, paving the way for safer and more reliable self-driving systems.

1.4 Problem Statement

In the rapidly advancing field of AD/ADAS, creating a robust and reliable driver model is absolutely critical. Our main goal in this research is to develop a driver model that excels in following paths and can quickly adapt to sudden changes in diverse testing scenarios. This capability is vital to ensure the safety and efficiency of autonomous vehicles, which must navigate dynamic and unpredictable environments.

To achieve our objective, it is crucial to explore different path-following algorithms that can be integrated into the driver model. Path-following algorithms play a crucial role in autonomous vehicle control systems, enabling them to accurately track a predefined path while reacting to real-time changes in the environment. These changes may include unexpected obstacles, variations in road conditions, and sudden maneuvers by other vehicles.

Through a comprehensive comparative analysis, our research aims to identify the most suitable path-following algorithm for the driver model. This involves a thorough testing and validation of various algorithms in simulated scenarios to evaluate their effectiveness. The outcomes of this analysis will offer valuable insights into the strengths and weaknesses of each algorithm, guiding us in selecting the optimal approach for developing the driver model.

1.5 Goal and Challenges

The overarching objective of this project is to develop an adaptive driver model that excels in path-following capabilities, particularly in Complex test scenarios. The primary aim is to ensure that the model can effectively address three key requirements:

- **Accurate Path Tracking:** The model should be adept at accurately tracking a predefined path generated by a "ghost vehicle," even when faced with complex test scenarios. This involves maintaining precise spatial alignment and trajectory following, regardless of test scenarios.

- **Dynamic Adaptability:** Another crucial aspect is the model's ability to quickly and safely adapt to changes in the road environment and traffic conditions. This adaptability ensures that the vehicle can respond effectively to changes in road conditions, thereby enhancing overall safety and performance.

- **Real-time Efficiency:** Furthermore, it is imperative for the model to exhibit high computational efficiency, enabling its seamless integration into real-time simulations. This efficiency ensures that the model's responses are prompt and accurate, allowing for realistic and responsive behavior in simulated environments.

The main challenges in achieving these goals are:

- **Algorithmic Complexity:** Designing an algorithm capable of effectively handling a wide spectrum of driving scenarios, ranging from complex geometric configurations to dynamic and unpredictable environments, presents a formidable challenge. The algorithm must possess the versatility and robustness to navigate diverse road layouts and adapt to rapidly changing conditions while maintaining optimal performance.

- **Trade-off Between Accuracy and Efficiency:** Balancing the need for accuracy with computational efficiency is a delicate balancing act. While accuracy is essential for precise path tracking and responsive behavior, excessive computational demands can hinder real-time performance. Striking the right balance between these conflicting requirements is essential for the successful implementation of the model.

- **Integration with Simulation Frameworks:** Ensuring compatibility with existing SIL simulation frameworks adds another layer of complexity to the development process. The algorithm must be seamlessly integrated into these frameworks, leveraging existing infrastructure and resources while minimizing disruptions and compatibility issues.

1.6 Scope & Limitations

This section outlines the scope of the research and the associated limitations to provide a clear understanding of the study's boundaries and constraints.

1.6.1 Scope

The scope of this project involves the following key components and activities:

- **Path-Following Algorithm Investigation:** Research and review existing path-following algorithms used in autonomous driving. Identify a range of algorithms that show promise for the desired application.

- **Driver Model Development:** Design and implement a driver model that incorporates the most suitable path-following algorithm. Ensure the model can accurately track paths from ghost vehicles, even in the presence of complex scenarios. Ensure the model can quickly and safely adapt to changes in road environments and traffic conditions.

- **Simulation Integration:** Integrate the selected path-following algorithm into the SIL simulation framework. Develop a controlled environment to assess the driver model's performance in response to dynamic test scenarios.

- **Comparative Analysis:** Develop criteria for evaluating the effectiveness of different path-following algorithms. Perform simulations to compare the selected algorithms under various test scenarios involving dynamic and unpredictable environments.

- **Optimization and Refinement:** Identify areas of improvement based on the comparative analysis. Optimize the selected path-following algorithm for enhanced adaptability while maintaining reliability.

1.6.2 Limitations

The project is subject to the following limitations, which may impact the outcomes and applicability of the developed driver model:

- **Simulation Accuracy:** The performance of the driver model is evaluated within simulated environments, which may not fully capture all real-world variables and scenarios. Results obtained from simulations might not translate perfectly to real-world performance due to limitations in simulation fidelity.

- **Computational Resources:** The efficiency of the driver model is dependent on the computational resources available for real-time simulations. Limited computational power may restrict the complexity and accuracy of the algorithms used.

- **Scope of Testing Scenarios:** The test scenarios used in the comparative analysis and simulations are predefined and may not cover all possible real-world driving conditions. Unexpected scenarios not included in the test cases might reveal weaknesses in the model's adaptability and performance.

- **Time Constraints:** The project timeline may limit the depth and breadth of the research and development activities. Comprehensive testing and validation of the driver model might be constrained by available time, leading to potential gaps in the evaluation process.

- **Model development limitation:** The project timeline also limits the Number of model that is developed.

1.7 Background

This project is conducted within Volvo Cars in Gothenburg, Sweden, under the Solution of Safe Vehicle Automation (SVA) initiative, specifically under the Advanced

Research & Technology (ART) division focusing on Software-in-the-Loop (SIL) simulations. The project is led by the Software In Loop (SIL) team.

The primary objective of this project is to develop new driver models that outperform the existing model used by the team. These driver models are essential for simulating and testing vehicle behaviors and interactions in various scenarios.

The project utilizes vehicle dynamics and control theory, C++, Python, and simulation environments such as ESmini OpenScenario Player for various scenario simulations. It involves creating and integrating new driver models alongside the existing ones within the SVADDS repository, enabling comprehensive comparisons and evaluations.

Within the SVA framework, the project focuses on:

- Developing new driver models aimed at enhancing vehicle motion control and automation capabilities.
- Integrating these new models into the existing software architecture to improve simulation accuracy and performance.
- Conducting thorough testing and validation to benchmark the performance of the new driver models against the existing ones.

By developing superior driver models, the SIL team aims to advance Volvo Cars' capabilities in safe and efficient vehicle automation technologies.

2

Theory

This Chapter delves into the theory behind the Vehicle model, concepts, Environments, selection of driver models, and also evaluation criteria used in developing the driver models.

2.1 Vehicle Model

Understanding vehicle dynamics is essential for developing effective path-following algorithms. Vehicle models can be broadly classified as:

2.1.1 Kinematic Vehicle Model

Figure 2.1 is a kinematic vehicle model that simplifies the representation of vehicle movement by focusing on geometric relationships and ignoring forces. It assumes low speeds and no lateral slip, making it computationally efficient for basic path-following applications.

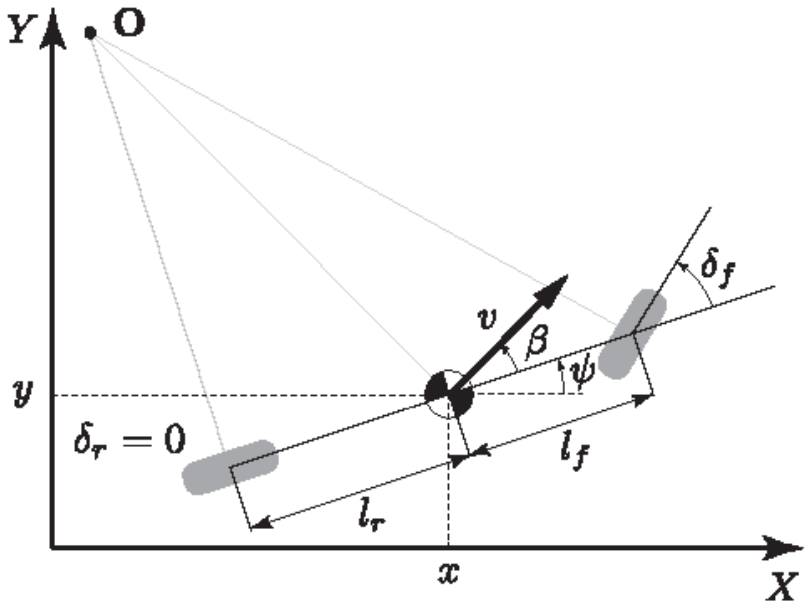


Figure 2.1: Kinematic Bicycle Model
[14]

The equations of motion for a kinematic bicycle model are:

$$\dot{x} = v\cos(\theta), \tag{2.1}$$
$$\dot{y} = v\sin(\theta), \tag{2.2}$$
$$\dot{\theta} = \frac{v}{L}\tan(\delta), \tag{2.3}$$

where:

- x, y are the vehicle's position coordinates,
- v is the longitudinal velocity,
- θ is the heading angle,
- δ is the steering angle,
- L is the wheelbase.

This model is widely used for trajectory tracking when precision in dynamics is not critical [14].

2.1.2 Dynamic Vehicle Model

2.1.2.1 Single-Track Bicycle Model for Lateral Dynamics

The single-track bicycle model (also known as the bicycle model) in Figure 2.2 is a simplified representation of a vehicle's dynamics. It reduces the four-wheel vehicle to a two-wheel equivalent, where the front and rear wheels are represented as single wheels. This simplification makes the model computationally efficient while preserving essential vehicle dynamics, particularly for lateral control at low to moderate speeds [4].

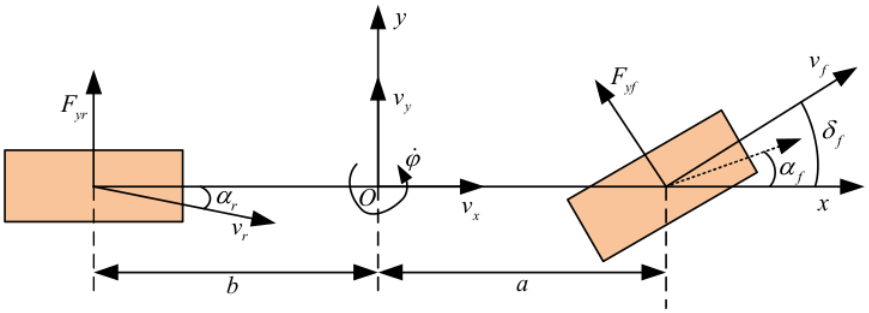


Figure 2.2: Vehicle Dynamics Model
[10]

2.1.2.1.1 Assumptions of the Single-Track Bicycle Model The following assumptions are made to simplify the vehicle dynamics [13, 14, 10]:

- **Symmetry of the Vehicle:** The vehicle is symmetric along its longitudinal axis.
- **Small Steering Angles and Wheel Slip Angles:** The front wheel steering angle (δ) and the front and rear wheel slip angles (α_f) and (α_r) are assumed to be small, so $\sin(\delta) \approx \delta$ and $\cos(\delta) \approx 1$.

- **Negligible Roll and Pitch Dynamics:** Only lateral (sideways) and yaw motions are considered.
- **Linear Tire Model:** Tire lateral forces (F_y) are assumed to be proportional to slip angles (α) within the linear region.

These assumptions allow the lateral dynamics of the vehicle to be modeled efficiently using linearized equations [4].

2.1.2.1.2 Lateral Dynamics Formulation The lateral dynamics of the vehicle are governed by two key motions [4, 14]:

- **Lateral Displacement (y):** Sideward motion of the vehicle.
- **Yaw Motion (ψ):** Rotation about the vertical axis.

The forces acting on the front and rear tires contribute to the vehicle's lateral dynamics and yaw motion. Using Newton's second law, the governing equations for the lateral and yaw dynamics are derived as:

$$m\left(\ddot{y} + v_x\dot{\psi}\right) = F_{yf} + F_{yr}, \tag{2.4}$$

$$I_z\ddot{\psi} = L_f F_{yf} - L_r F_{yr}, \tag{2.5}$$

$$\dot{y} = v_y \tag{2.6}$$

where:

- m: Mass of the vehicle,
- v_x: Longitudinal velocity,
- v_y: Lateral velocity,
- ψ: Yaw angle,
- I_z: Moment of inertia about the z-axis,
- L_f and L_r: Distances from the center of gravity to the front and rear axles,
- F_{yf} and F_{yr}: Lateral forces at the front and rear tires [4].

The lateral tire forces can be approximated using the linear tire model:

$$F_{yf} = -C_f\alpha_f, \quad F_{yr} = -C_r\alpha_r, \tag{2.7}$$

where:

- C_f and C_r: Cornering stiffness of the front and rear tires,
- α_f and α_r: Tire slip angles [19].

The slip angles are defined as:

$$\alpha_f = \delta - \frac{\dot{y} + L_f\dot{\psi}}{v_x}, \tag{2.8}$$

$$\alpha_r = -\frac{\dot{y} - L_r\dot{\psi}}{v_x}. \tag{2.9}$$

Substituting the slip angles and simplifying the equations yields the linearized lateral dynamics [14, 4].

2.1.2.2 Error Dynamics Model

Error functions are critical to understanding and evaluating the performance of controllers in trajectory tracking. Figure 2.3 illustrates the error dynamics, which represent the vehicle's deviations from the reference path. The errors functions utilized in this thesis are:

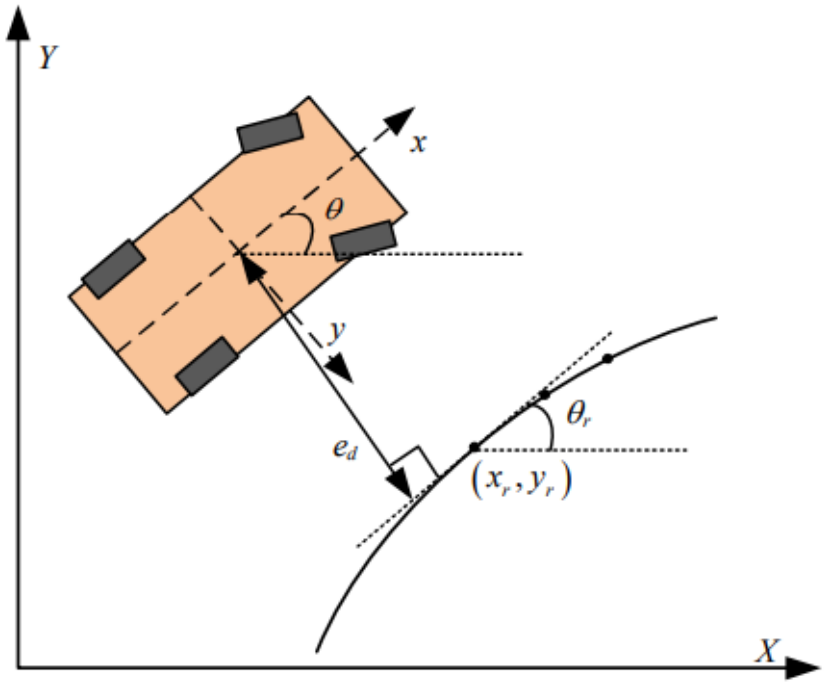


Figure 2.3: Path Tracking Model
[10]

- **Cross-Track Error (e_d):**The perpendicular distance between the vehicle's current position and the reference trajectory,
- **Heading Error (e_ψ):** The angular difference between the vehicle's heading and the tangent to the reference trajectory.

Each error plays a unique role in controller design:

- **Cross-Track Error** influences lateral adjustments.
- **Heading Error** ensures the vehicle's orientation aligns with the trajectory.

By addressing these errors, controllers optimize trajectory tracking and enhance vehicle stability. The consideration of these errors into dynamics, and how these errors are mitigated over time, can be seen further in section 3.1.2. The state vector for the lateral error dynamics can be defined as:

$$\mathbf{x} = \begin{bmatrix} e_d \\ \dot{e}_d \\ e_\psi \\ \dot{e}_\psi \end{bmatrix},$$

where:

- e_{ct} and $\dot{e}_{ct}$: Cross-track error and its rate,
- e_{ψ} and $\dot{e}_{\psi}$: Heading error and its rate [13].

The control input is the front wheel steering angle (δ), and the state-space formulation is used to describe the system dynamics [15, 16].

2.2 Vehicle Model Parameters

In this section, we discuss the various parameters that are crucial in the modeling and simulation of a vehicle's dynamic behavior. These parameters are essential for understanding and predicting the vehicle's response under different driving conditions. Below are descriptions of the key parameters used in the vehicle model.

2.2.1 Primary & Additional Vehicle Model Parameters

Table 2.1: Key Parameters for Vehicle Dynamics

Parameter	Description
`cf_`	Tire cornering stiffness for the front tires
`cr_`	Tire cornering stiffness for the rear tires
`m1_`	Vehicle mass
`lf_`	Distance from the center of gravity (CG) to the front axle
`lr_`	Distance from the center of gravity (CG) to the rear axle
`iz_`	Yaw inertia

Additional parameters are used to simulate realistic vehicle control inputs and constraints. These include maximum brake force and torque, maximum propulsion torque, pedal and steering positions, and the time to achieve these maximum positions. Parameters also include thresholds for determining vehicle standstill.
Understanding these parameters is crucial for accurate vehicle dynamics modeling and simulation. They play a significant role in determining the vehicle's performance and behavior under various driving conditions and scenarios.

2.3 Simulation Environment

2.3.1 Introduction

Simulation environments play a crucial role in the development and testing of autonomous driving systems. They provide a controlled, reproducible, and safe setting to evaluate various algorithms and scenarios that would be impractical or dangerous to test in real life. In this thesis, we utilize the esmini OpenSCENARIO player as our

primary simulation tool. Esmini, short for 'essential mini', is a lightweight, open-source simulator designed to facilitate the development and testing of automated driving functions, with a particular focus on scenario-based testing.

2.3.2 Overview of esmini

esmini is designed to interpret and execute scenarios defined in the OpenSCENARIO format, an XML-based standard for describing dynamic content in driving simulation. This includes traffic situations, road layouts, and environmental conditions. By adhering to the OpenSCENARIO standard, esmini ensures compatibility with a wide range of simulation tools and facilitates the sharing of scenarios across different platforms.

2.3.3 Use Cases

Esmini has a wide range of use cases in the field of autonomous driving research and development:

- **Algorithm Development**: Researchers can develop and test new path-following algorithms, adaptive cruise control, lane-keeping systems, and other driving functions in a controlled environment.
- **Scenario Testing**: Engineers can create and execute predefined scenarios to assess the performance of autonomous driving systems under various conditions, including complex traffic situations and adverse weather.
- **Safety Assessment**: By simulating potentially dangerous scenarios, such as sudden pedestrian crossings or vehicle malfunctions, esmini allows for rigorous safety assessments without risking real-world accidents.
- **Training and Validation**: Esmini can be used to train machine learning models for tasks like object detection and decision-making, as well as validate these models against a diverse set of scenarios.

2.3.4 Scenario Features

Esmini supports a comprehensive set of features for defining and executing driving scenarios as in Figure 2.4:

- **Traffic Participants**: Users can define multiple types of traffic participants, including different vehicle models, cyclists, and pedestrians, each with customizable behaviors and properties. As seen in Figure 2.4, the right top image gives a representation of bringing in additional Vehicle on opposite Lane. Also Bottom Right image of Figure 2.4 represrents a red vehicle,which can be seen along with the blue vehicle. The red Vehicle is the actual position the ego vehicle model should be at the moment, but there is a slight deviation.
- **Environment Conditions**: Scenarios can include various environmental conditions, such as different weather states (rain, fog, snow), lighting conditions (day, night, dusk), and road surface conditions (wet, icy).
- **Dynamic Elements**: Esmini allows for the inclusion of dynamic elements like traffic lights, road signs, and construction zones, which can influence the behavior of the simulated vehicles.

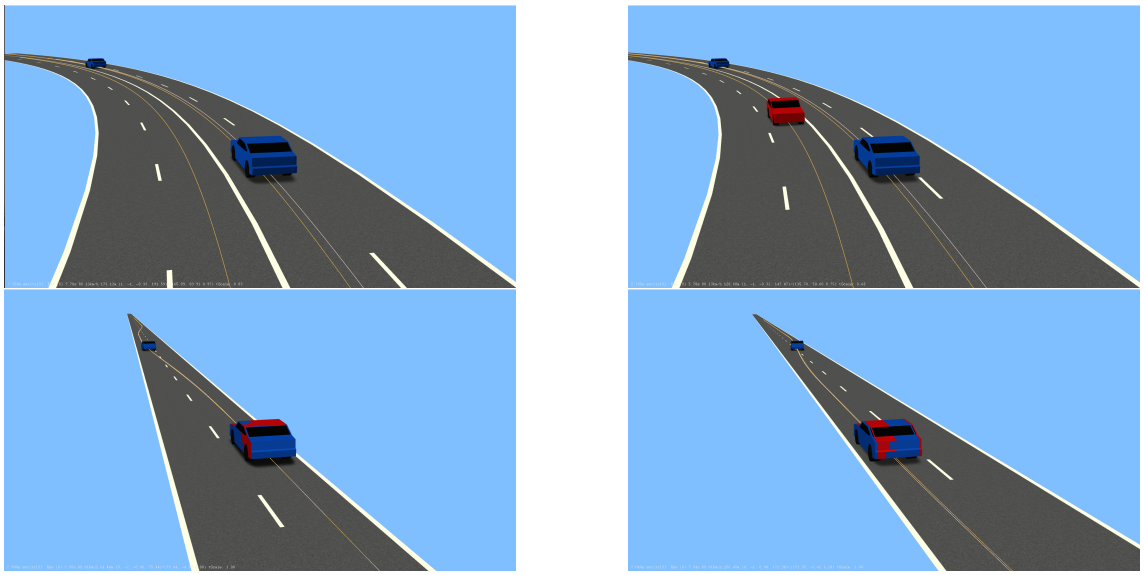


Figure 2.4: Representation of Scenario Features

- **Interactions and Events**: Scenarios can specify interactions and events, such as vehicles merging into lanes, sudden stops, and emergency maneuvers, to test the robustness of the autonomous driving system.
- **Complex Road Networks**: The simulation environment supports complex road networks, including highways, urban streets, intersections, and roundabouts, providing a realistic setting for testing various driving scenarios. All the images in the Figure 2.4, represents a highway and country road.

2.4 Selection of Driver Models

The selection of driver models in this thesis was motivated by the need to evaluate and improve path-following performance in autonomous vehicles. The focus was on developing advanced models and comparing their performance with the **Stanley Controller**, which served as the baseline model. The Stanley Controller, already integrated into the existing framework, provided a reliable reference point for benchmarking the effectiveness of newly developed models under various test scenarios.

The limitations of the Stanley Controller in handling high-speed dynamics, sharp turns, and rapid transitions necessitated the exploration of more advanced control models[15]. Specifically, the Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC) were selected due to their robust optimization capabilities and adaptability to dynamic conditions[13][14].

The driver models explored in this research include:

- **Linear Quadratic Regulator (LQR):** LQR was selected for its optimization-driven approach to lateral control. By minimizing a cost function that incorporates cross-track error, heading deviation, and control effort, LQR offers

improved performance over the Stanley Controller in dynamic scenarios. Its adaptability and robust control make it a strong candidate for replacing or augmenting the baseline model.

- **Model Predictive Control (MPC):** MPC was included for its predictive capabilities, which allow the model to anticipate future states and optimize control actions accordingly. Unlike the Stanley Controller, MPC can handle complex constraints and dynamic environmental changes, making it well-suited for real-world path-following tasks.

The comparison between these models focuses on their ability to address the limitations of the baseline while maintaining or exceeding its performance in critical metrics such as cross-track error, heading deviation, and computational efficiency which will be further discussed in subsections 2.5.1, 2.5.4 . By benchmarking against the Stanley Controller, this research demonstrates the superiority of the newly developed models in handling diverse and complex test scenarios. This comparative analysis provides valuable insights into the strengths and trade-offs of each model, guiding their refinement and potential deployment in autonomous vehicle systems.

2.5 Evaluation Criteria - Post Analysis

To conduct a comprehensive comparative analysis of different path-following models and evaluate their performance under various conditions, we will use the following criteria. These metrics are carefully chosen to provide an in-depth understanding of the controllers' performance while addressing key challenges in fair comparisons.

2.5.1 Trajectory Accuracy

This criterion measures how closely the vehicle follows the desired path. Accurate trajectory tracking ensures that the vehicle minimizes deviations, reducing the risk of unsafe maneuvers or inefficient navigation.

Metrics:

- **Euclidean Distance:** Measures point-by-point deviation from the desired path. This provides insight into localized performance, especially during rapid maneuvers.
- **Root Mean Square Error (RMSE):** A holistic measure of trajectory accuracy over the entire path. RMSE accounts for cumulative deviations and reflects overall tracking performance.

The choice of these metrics highlights the importance of precision in trajectory adherence, a key requirement for safe and reliable autonomous driving.

2.5.2 Performance at different speeds with various scenarios

This criterion evaluates the model's adaptability and robustness across a range of operating conditions. Differences in controller tuning can significantly affect performance, emphasizing the need for fair comparisons under consistent scenarios.

2.5.2.1 Test Scenarios

- **Constant speed curved road scenarios:** Tests the controllers' ability to handle sustained lateral demands.
- **Double lane change scenarios:**Challenges controllers with abrupt transitions, simulating emergency maneuvers.

Metrics:

- **Stability of Control Inputs Across Speed Ranges:** Assesses the consistency and smoothness of steering and throttle inputs as speed varies.
- **Consistency of Trajectory Accuracy at Different Speeds:** Evaluates how well the controller maintains precision across low, moderate, and high-speed conditions.

These scenarios and metrics are selected to ensure that controllers are tested under realistic and challenging conditions. However, controller tuning parameters can introduce variability in performance, making direct comparisons complex. Addressing this challenge requires careful analysis of tuning impacts versus inherent strategy effectiveness.

2.5.3 Convergence Speed

This criterion measures how quickly the model can correct deviations and return to the desired path, a critical factor in dynamic environments.

Metrics:

- **Time to Convergence:** The duration required for the vehicle to return to within an acceptable error threshold after a disturbance. Shorter convergence times indicate higher responsiveness and control effectiveness.

Convergence speed reflects the controller's ability to recover from unexpected disturbances, a vital aspect of autonomous navigation.

2.5.4 Computational Time

This criterion measures the computational efficiency of the model, reflecting its feasibility for real-time applications.

Metrics:

- **Mean Elapsed Time:** The average duration required for the model to complete the scenario across all three speed settings.

While advanced controllers like MPC offer superior accuracy, their computational demands may limit real-time applicability. Comparing computational times provides insights into the trade-offs between performance and efficiency.

3
Methods

This Chapter focuses on separation of dynamics, and formulation of driver models.

3.1 Formulation of Driver Models or Controllers

This section Focuses on the formulation of driver models, in terms of longitudnal and lateral controllers separately. Since, all three driver models utilizes the same longitudnal controller, it is important to discuss its formulation and assumptions made for decoupling for the controller separation to be valid.

3.1.1 Longitudinal Controller and Separation of Dynamics

The longitudinal controller remains constant across all lateral controllers in this study, simplifying comparisons and focusing on lateral performance. The primary function of the longitudinal controller is to regulate the vehicle's velocity by managing throttle and brake inputs to achieve a desired speed. This decoupling of longitudinal and lateral dynamics is based on specific assumptions, enabling modular controller design and independent performance evaluation.

3.1.1.1 Proportional-Integral-Derivative (PID)

The PID controller is widely used for longitudinal control, adjusting the vehicle's acceleration and braking to maintain the desired speed. The PID controller minimizes the error between the desired speed (v_d) and the actual speed (v_a) by adjusting the control input (u) based on three terms:

- **Proportional (P)**: The proportional term produces an output that is proportional to the current error $e(t) = v_d - v_a$.

  In continuous form:

  $$u_p = K_p e(t) \tag{3.1}$$

  In discrete form:

  $$u_p[k] = K_p e[k] \tag{3.2}$$

- **Integral (I)**: The integral term is based on the accumulation of past errors, providing a corrective action to eliminate the residual steady-state error.

  In continuous form:
  $$u_i = K_i \int_0^t e(\tau)\, d\tau \tag{3.3}$$
  In discrete form:
  $$u_i[k] = K_i \sum_{j=0}^{k} e[j]\Delta t \tag{3.4}$$
  The integral term is conditionally updated to prevent integral windup:

  If ($u_{prev} \geq u_{max}$ and $e[k] \geq 0$) or ($u_{prev} \leq u_{min}$ and $e[k] \leq 0$):
  No update to integral term
  Else:
  Integral term is updated

- **Derivative (D)**: The derivative term predicts future error based on its rate of change, adding a damping effect to the control action.

  In continuous form:
  $$u_d = K_d \frac{de(t)}{dt} \tag{3.5}$$
  In discrete form:
  $$u_d[k] = K_d \frac{e[k] - e[k-1]}{\Delta t} \tag{3.6}$$
  A filtered derivative is used to reduce noise sensitivity:
  $$u_d^{filtered}[k] = (1-\alpha) u_d^{filtered}[k-1] + \alpha \frac{u_d[k] + u_d[k-1]}{2} \tag{3.7}$$

unconstrained mpcThe final control output in discrete form is the sum of these terms:
$$u[k] = u_p[k] + u_i[k] + u_d^{filtered}[k] \tag{3.8}$$

Assumptions for Decoupling: The decoupling of longitudinal and lateral control is valid under the following assumptions:

1. **Minimal Coupling Effects:** Longitudinal dynamics (throttle and braking) have negligible influence on lateral behavior within the tested operational range.
2. **Linear Speed Ranges:** The scenarios are constrained to speed ranges where coupling effects are minimal, ensuring independent control actions.
3. **Negligible Lateral Forces Impact:** Lateral maneuvers, such as sharp turns, do not significantly affect the vehicle's longitudinal velocity.
4. **Simplified Modeling:** External factors like tire slip and road surface variations are not explicitly modeled in this study, focusing solely on the trajectory tracking performance.

By maintaining a consistent longitudinal controller, the study isolates and evaluates the unique contributions of lateral controllers, such as LQR, MPC, and Stanley, to overall path-following performance.

3.1.2 Lateral Controllers

3.1.2.1 Stanley Controller - VCC-SIL Existing Driver Model

The Stanley controller is used for lateral control to minimize the cross-track error (e_{ct}) and heading error (θ_e). The control law is designed to adjust the steering angle (δ) to keep the vehicle on its desired path [12]. It combines two error terms:

1. Heading error (θ_e):

$$\theta_e = \psi_{ghost} - \psi_{ego} \tag{3.9}$$

2. Cross-track error (θ_d):
The cross-track error is calculated by projecting the position error onto the front axle vector:

$$f_e = (x_{ego} - x_{ghost}) \cos\left(\psi_{ego} + \frac{\pi}{2}\right) + (y_{ego} - y_{ghost}) \sin\left(\psi_{ego} + \frac{\pi}{2}\right) \tag{3.10}$$

$$\theta_d = \arctan\left(\frac{k \cdot f_e}{k_v + v_x}\right) \tag{3.11}$$

where:
- ψ_{ghost} is the Ghost vehicle heading
- ψ_{ego} is the ego vehicle heading
- k is the gain parameter
- f_e is the front axle error
- k_v is a speed-dependent gain
- v_x is the longitudinal velocity of the vehicle

The final steering angle command is the sum of these two terms:

$$\delta = \theta_e + \theta_d \tag{3.12}$$

This steering angle is then limited to a maximum range, typically:

$$\delta_{final} = \text{clamp}(\delta, -12.7^\circ, 12.7^\circ) \tag{3.13}$$

3.1.3 Model Predictive Controller (MPC)

The first driver model developed in this master thesis integrates a PID controller for longitudinal control and a Model Predictive Controller (MPC) for lateral control. The longitudinal dynamics, which involve acceleration and braking, are managed using a Proportional-Integral-Derivative (PID) controller and are not covered in this model. The MPC is used for lateral control by optimizing steering commands over a finite time horizon. It predicts the vehicle's future trajectory using a dynamic model and minimizes a cost function, which typ- ically includes cross-track

error, heading error, and control effort. MPC is partic- ularly suited for complex driving scenarios due to its ability to handle constraints and anticipate future events.

3.1.3.1 State-Space Model for Lateral Dynamics

The continuous-time state-space vehicle model for the lateral dynamics can be represented as:

$$\dot{X} = A_c \cdot X + B_c \cdot U \tag{3.14}$$

$$Y = C_c \cdot X + D_c \cdot U \tag{3.15}$$

where:

- A_c, B_c, C_c, and D_c are the continuous-time system matrices.
- $\dot{X}$ is the derivative of the state vector X.
- U is the control input vector.
- Y is the output vector.
- A_c is the system matrix.
- B_c is the input matrix.
- C_c is the output matrix.
- D_c is the feedforward or disturbance matrix (not considered in thesis), D = 0.

3.1.3.2 System Matrices

Matrix A_c (System Matrix):

The matrix A_c captures the dynamics of the vehicle. For our linearized lateral vehicle model, it is initialized as follows:

$$A_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{c_f + c_r}{m \cdot v} & -\frac{c_f + c_r}{m} & \frac{l_f \cdot c_f - l_r \cdot c_r}{m \cdot v} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{l_f \cdot c_f - l_r \cdot c_r}{I_z \cdot v} & \frac{l_f \cdot c_f - l_r \cdot c_r}{I_z} & \frac{l_f^2 \cdot c_f + l_r^2 \cdot c_r}{I_z \cdot v} \end{bmatrix}$$

Matrix B (Input Matrix):

The matrix B_c defines how the control input U affects the state dynamics. For our model, it is initialized as:

$$B_c = \begin{bmatrix} 0 \\ -\frac{c_f}{m} \\ 0 \\ -\frac{l_f \cdot c_f}{I_z} \end{bmatrix}$$

Matrix C (Output Matrix):

The matrix C_c translates the state vector into the output vector. For this model, the matrix C_c is initialized as:

$$C_c = \begin{bmatrix} 0 \\ \left(\frac{l_f \cdot c_f - l_r \cdot c_r}{m \cdot v}\right) - v \\ 0 \\ \frac{l_f^2 \cdot c_f + l_r^2 \cdot c_r}{I_z \cdot v} \end{bmatrix}$$

State Vector X:

The state vector X comprises variables that describe the lateral dynamics of the vehicle. These variables are initialized as:

$$X = \begin{bmatrix} e_y \\ \dot{e}_y \\ \theta_e \\ \dot{\theta}_e \end{bmatrix}$$

Control Input U:

The control input U for this model is the steering angle, which directly influences the yaw dynamics of the vehicle.

By utilizing this state-space representation, we can apply several control strategies to optimally manage the vehicle's lateral dynamics, MPC. This model captures the essential dynamics needed for effective steering control, while the longitudinal dynamics are managed separately using a PID controller.

3.1.3.3 Unified State-Space Formulation

In this subsection, we consolidate the matrices A, B, and C into a unified continuous-time state-space formulation, which fully characterizes the vehicle's lateral dynamics [11].

$$\dot{X} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{c_f + c_r}{m \cdot v} & -\frac{c_f + c_r}{m} & \frac{l_f \cdot c_f - l_r \cdot c_r}{m \cdot v} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{l_f \cdot c_f - l_r \cdot c_r}{I_z \cdot v} & \frac{l_f \cdot c_f - l_r \cdot c_r}{I_z} & \frac{l_f^2 \cdot c_f + l_r^2 \cdot c_r}{I_z \cdot v} \end{bmatrix} \cdot \begin{bmatrix} e_y \\ \dot{e}_y \\ \theta_e \\ \dot{\theta}_e \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{c_f}{m} \\ 0 \\ -\frac{l_f \cdot c_f}{I_z} \end{bmatrix} \cdot U \tag{3.16}$$

$$Y = \begin{bmatrix} 0 \\ \left(\frac{l_f \cdot c_f - l_r \cdot c_r}{m \cdot v}\right) - v \\ 0 \\ \frac{l_f^2 \cdot c_f + l_r^2 \cdot c_r}{I_z \cdot v} \end{bmatrix} \cdot \begin{bmatrix} e_y \\ \dot{e}_y \\ \theta_e \\ \dot{\theta}_e \end{bmatrix} \tag{3.17}$$

This comprehensive state-space formulation is pivotal for implementing advanced control strategies. By manipulating the state vector X through the control input

U, the system can be driven towards the desired output Y, optimizing the vehicle's lateral dynamics under various driving conditions.

3.1.3.4 System Model Discretization

When implementing a Model Predictive Control (MPC) algorithm in a digital controller, the controller needs to work with a discrete-time version of the system. This is because digital controllers operate by processing data at discrete intervals, not continuously. So, the continuous-time system (2.1.2) is discretized using a sampling time T_{sam}. The discrete-time state-space model is represented as:

$$x_{k+1} = A_d x_k + B_d u_k \tag{3.18}$$

$$y_k = C_d x_k \tag{3.19}$$

$$\mathbf{A}_d = \left(\mathbf{I} - \frac{T_{\text{sam}}}{2}\mathbf{A}\right)^{-1}\left(\mathbf{I} + \frac{T_{\text{sam}}}{2}\mathbf{A}\right) \tag{3.20}$$

$$\mathbf{B}_d = T_{\text{sam}} \cdot \mathbf{B} \tag{3.21}$$

$$\mathbf{C}_d = \mathbf{C} \tag{3.22}$$

Where:
$\mathbf{A}_d$, $\mathbf{B}_d$, and $\mathbf{C}_d$ are the discrete-time system matrices and $\mathbf{I}$ is the identity matrix of appropriate dimensions.

3.1.3.5 MPC Cost Function

The MPC aims to minimize a cost function that penalizes deviations from the desired trajectory(Q) and control input (R).

At each time step, the following optimization problem is solved, subject to the system dynamics.

$$J = \sum_{i=1}^{H_p}(y_{k+i|k} - r_{k+i})^T Q(y_{k+i|k} - r_{k+i}) + \sum_{i=0}^{H_c-1} \Delta u_{k+i|k}^T R \Delta u_{k+i|k} \tag{3.23}$$

Where:

- $y_{k+i|k}$ is the predicted output,
- r_{k+i} is the reference trajectory(Ghost_Vehicle),
- $\Delta u_{k+i|k}$ is the change in control input,
- H_p is the prediction horizon,
- H_c is the control horizon, and
- Q and R are weighting matrices.

3.1.3.6 Weight Matrices

The weight matrices **Q** and **R** are initialized to penalize the state deviation errors and control input, respectively:

$$Q = \text{diag}(Q_0, Q_1, Q_2, ..., Q_{H_p}) \tag{3.24}$$

where $Q_{0,1,2,....,H_p}$ are individual scalar weights for each prediction step.

$$\mathbf{R}_1 = \begin{bmatrix} \mathbf{I} & -\mathbf{I} & & \\ & \mathbf{I} & -\mathbf{I} & \\ & & \ddots & \ddots \\ & & & \mathbf{I} \end{bmatrix} \tag{3.25}$$

$$\mathbf{R}_2 = \begin{bmatrix} Q_0 & 0 & \dots & 0 \\ 0 & Q_{\text{other}} & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & Q_{\text{other}} \end{bmatrix} \tag{3.26}$$

$$\mathbf{R} = \mathbf{R}_1^T \mathbf{R}_2 \mathbf{R}_1 \tag{3.27}$$

3.1.3.7 Control Law

The optimal control sequence is computed as:

$$U = K(R - Ox_k) \tag{3.28}$$

where U is the vector of future control inputs, K is the gain matrix, R is the vector of future reference values, O is the observability matrix, and x_k is the current state.

3.1.3.8 Lifted System Matrices

The lifted system matrices O and M are constructed as:

$$O = \begin{bmatrix} C_d A_d \\ C_d A_d^2 \\ \vdots \\ C_d A_d^{H_p} \end{bmatrix} \tag{3.29}$$

$$M = \begin{bmatrix} C_d B_d & 0 & \cdots & 0 \\ C_d A_d B_d & C_d B_d & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_d A_d^{H_p-1} B_d & C_d A_d^{H_p-2} B_d & \cdots & C_d A_d^{H_p-H_c} B_d \end{bmatrix} \tag{3.30}$$

3.1.3.9 Gain Matrix

The gain matrix K is computed as:

$$K = (M^T Q M + R)^{-1} M^T Q \tag{3.31}$$

3.1.3.10 Prediction and Control Horizons

The prediction horizon H_p and control horizon H_c are chosen based on the system dynamics and control requirements, with $H_c \leq H_p$.

This formulation provides a comprehensive overview of the MPC controller design, capturing the key elements from the provided code without directly including the implementation details.

3.1.4 Linear Quadratic Regulator (LQR)

The second driver model in this master thesis employs a PID controller for longitudinal control and a Linear Quadratic Regulator (LQR) for lateral control. The LQR is used for lateral control and is designed to minimize a quadratic cost function that balances state errors and control inputs. It works by solving the Algebraic Riccati Equation (ARE) to find the optimal gain matrix K that minimizes the cost function. The LQR aims to minimize the cross-track error and heading error while considering the control input's magnitude. The LQR is known for its robustness and efficiency in achieving desired performance in linear systems, making it suitable for real-time vehicle control applications [10].

3.1.4.1 State-Space Model for Lateral Dynamics

The vehicle model is a fundamental part of the design and analysis of control systems for vehicle dynamics. In this section, we focus on the state-space representation of the lateral dynamics of a vehicle, which is essential for LQR controller in steering control[10].

The longitudinal dynamics, which involve acceleration and braking, are managed using a Proportional-Integral-Derivative (PID) controller, which has been previously discussed in 3.1.1.1.

The continuous-time state-space vehicle model for the lateral dynamics can be represented as:

$$\dot{X} = A_c \cdot X + B_c \cdot U + C_c \cdot \dot{\theta}_r \tag{3.32}$$

where:

- A_c, B_c, and C_c are the continuous-time system matrices.
- $\dot{X}$ is the derivative of the state vector X.
- U is the control input vector.
- A_c is the system matrix.
- B_c is the input matrix.
- C_c is the Feed-Forward matrix.

3.1.4.1.1 System Matrices: Matrix A_c (System Matrix):

The matrix A_c captures the dynamics of the vehicle. For our linearized lateral vehicle model, it is initialized as follows:

$$A_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{2(c_f+c_r)}{m_1 \cdot v_x} & -\frac{2(c_f+c_r)}{m_1} & \frac{2(l_f \cdot c_f - l_r \cdot c_r)}{m_1 \cdot v_x} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{2(l_f \cdot c_f - l_r \cdot c_r)}{I_z \cdot v_x} & -\frac{2(l_f \cdot c_f - l_r \cdot c_r)}{I_z} & \frac{2(l_f^2 \cdot c_f + l_r^2 \cdot c_r)}{I_z \cdot v_x} \end{bmatrix}$$

Matrix B_c (Input Matrix):

The matrix B_c defines how the control input U affects the state dynamics. For our model, it is initialized as:

$$B_c = \begin{bmatrix} 0 \\ -\frac{2c_f}{m_1} \\ 0 \\ -\frac{2l_f \cdot c_f}{I_z} \end{bmatrix}$$

Matrix C_c (Output Matrix):

The matrix C_c is Feedforward Matrix. For this model, the matrix C_c is initialized as:

$$C_c = \begin{bmatrix} 0 \\ \frac{2(l_f \cdot c_f - l_r \cdot c_r)}{m_1 \cdot v_x} - v_x \\ 0 \\ \frac{2(l_f^2 \cdot c_f + l_r^2 \cdot c_r)}{I_z \cdot v_x} \end{bmatrix}$$

The states and control inputs are defined in previous section 2.1.2.2.
By utilizing this state-space representation, we can apply several control strategies to optimally manage the vehicle's lateral dynamics, . This model captures the essential dynamics needed for effective steering control.

3.1.4.2 Unified State-Space Formulation

In this subsection, we consolidate the matrices A_c, B_c, and C_c into a unified continuous-time state-space formulation, which fully characterizes the vehicle's lateral dynamics[11].

$$\dot{X} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{2(c_f+c_r)}{m_1 \cdot v_x} & -\frac{2(c_f+c_r)}{m_1} & \frac{2(l_f \cdot c_f - l_r \cdot c_r)}{m_1 \cdot v_x} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{2(l_f \cdot c_f - l_r \cdot c_r)}{I_z \cdot v_x} & -\frac{2(l_f \cdot c_f - l_r \cdot c_r)}{I_z} & \frac{2(l_f^2 \cdot c_f + l_r^2 \cdot c_r)}{I_z \cdot v_x} \end{bmatrix} \cdot \begin{bmatrix} e_y \\ \dot{e}_y \\ \theta_e \\ \dot{\theta}_e \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{2c_f}{m_1} \\ 0 \\ -\frac{2l_f \cdot c_f}{I_z} \end{bmatrix} \cdot \delta + \begin{bmatrix} 0 \\ \frac{2(l_f \cdot c_f - l_r \cdot c_r)}{m_1 \cdot v_x} - v_x \\ 0 \\ \frac{2(l_f^2 \cdot c_f + l_r^2 \cdot c_r)}{I_z \cdot v_x} \end{bmatrix} \cdot \dot{\theta}_r \tag{3.33}$$

This comprehensive state-space formulation is pivotal for implementing advanced control strategies. By manipulating the state vector X through the control input U, the system can be driven towards the desired output Y, optimizing the vehicle's lateral dynamics under various driving conditions.

3.1.4.3 System Model Discretization

When implementing a Linear Quadratic Regulator(LQR) algorithm in a digital controller, the controller needs to work with a discrete-time version of the system. This is because digital controllers operate by processing data at discrete intervals, not continuously. So, the continuous-time system (3.33) is discretized using a sampling time T_{sam}. The discrete-time state-space model is represented similar to section 3.1.3.4 discussed before.
Where:
$\mathbf{A}_d$, $\mathbf{B}_d$, and $\mathbf{C}_d$ are the discrete-time system matrices and $\mathbf{I}$ is the identity matrix of appropriate dimensions.

3.1.4.4 Path-Tracking Controller with LQR and Feedforward Control

The overall structure of the path-tracking controller is shown in Figure 3.1. Based on the path-tracking error model, an LQR controller is designed as the central component of the path-tracking system. The LQR controller ensures the vehicle can follow the reference path by minimizing tracking errors and balancing control effort. To further improve performance, a feedforward control method is integrated to eliminate the steady-state error caused by system dynamics, thereby optimizing the overall path-tracking performance [20, 10].

The feedforward controller generates a compensation term, δ_{ff}, to account for the steady-state error in the steering angle. This term works alongside the LQR feedback control term, $-\mathbf{KX}$, to provide the total steering angle δ_f, ensuring accurate path tracking.

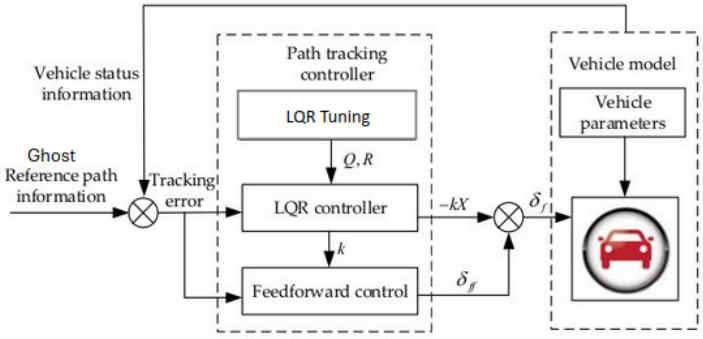


Figure 3.1: Overall structure of the path-tracking controller, incorporating LQR and feedforward control.

3.1.4.4.1 Controller Structure

The system works as follows:

- **LQR Controller**: The LQR controller minimizes the tracking errors by calculating the optimal feedback term, $-\mathbf{KX}$, using the weight matrices $\mathbf{Q}$ and $\mathbf{R}$.
- **Feedforward Controller**: This component generates the compensation term δ_{ff} to eliminate steady-state errors caused by the vehicle dynamics.
- **Vehicle Model**: The feedback and feedforward control inputs are combined to adjust the steering angle δ_f, ensuring the vehicle follows the desired path while considering the vehicle parameters.

The combination of feedback and feedforward control ensures robust and precise path-tracking performance in autonomous vehicles.

3.1.4.5 LQR Cost Function

The Linear Quadratic Regulator (LQR) optimizes the steering commands by minimizing a quadratic cost function that balances the trade-off between performance (cross-track error and heading error) and control input. [20].

LQR is widely used in autonomous driving path tracking control due to its ability to handle deviations from the steady state caused by obstacles or unexpected events. This ensures the vehicle tracking control system approaches the desired path without excessive computational workload. The LQR control problem for the vehicle lateral path tracking controller can be formulated as:

$$\min J = \sum_{k=0}^{\infty} \left(\mathbf{X}_k^T \mathbf{Q} \mathbf{X}_k + \mathbf{U}_k^T \mathbf{R} \mathbf{U}_k \right), \tag{3.34}$$

where:

- $\mathbf{X}$ denotes the system state variable,
- $\mathbf{U}$ represents the control variable,

- **Q** and **R** are weight matrices that penalize the state errors and control inputs, respectively.

The weight matrix **Q** indicates the relative importance of each state error in the performance index and is represented as:

$$\mathbf{Q} = \text{diag}[q_1, q_2, q_3, q_4], \tag{3.35}$$

where q_1, q_2, q_3, q_4 correspond to the weights for lateral error, lateral error rate of change, heading error, and heading error rate of change, respectively. The weight matrix **R**, representing the control effort for the steering angle, is defined as:

$$\mathbf{R} = [q_5]. \tag{3.36}$$

Higher values in **Q** prioritize minimizing the associated state variable errors, while a larger value in **R** emphasizes reducing control efforts, balancing performance and efficiency.

The LQR framework allows to tune the weight coefficients in **Q** and **R** to balance path tracking accuracy and control effort. This makes LQR a robust and computationally efficient choice for path tracking control in autonomous vehicles.

3.1.4.6 LQR Control Law

The control quantity of the LQR controller is derived by solving the extreme value of the quadratic performance index. The optimal control law is expressed as:

$$\mathbf{U}_k = -\left(\mathbf{R} + \mathbf{B}^T\mathbf{P}\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{P}\mathbf{A}\mathbf{X}_k, \tag{3.37}$$

where **P** is the solution to the discrete Riccati equation [20]:

$$\mathbf{P}_k = \mathbf{Q} + \mathbf{A}^T\mathbf{P}_{k+1}\mathbf{A} - \mathbf{P}_{k+1} - \mathbf{A}^T\mathbf{P}_{k+1}\mathbf{B}\left(\mathbf{R} + \mathbf{B}^T\mathbf{P}_{k+1}\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{P}_{k+1}\mathbf{A}. \tag{3.38}$$

For practical implementation, the control law can be simplified as:

$$\mathbf{U}_k = -\mathbf{K}\mathbf{X}_k, \tag{3.39}$$

where $\mathbf{K} = [k_1, k_2, k_3, k_4]$ is the feedback gain matrix derived from the LQR controller, representing the control coefficients for the front wheel angle [21].

3.1.4.7 Feedforward Control

To address the steady-state error that occurs in the presence of disturbances or system dynamics, a feedforward control term, δ_{ff}, is introduced [4]. The system dynamics, incorporating the feedforward term, are given by:

$$\dot{\mathbf{X}} = (\mathbf{A} - \mathbf{B}\mathbf{K})\,\mathbf{X} + \mathbf{C}\dot{\theta}_r. \tag{3.40}$$

The feedforward control term is designed to eliminate the influence of $\mathbf{C}\dot{\theta}_r$ on the system. The total control variable after including the feedforward term becomes:

$$\mathbf{U} = -\mathbf{K}\mathbf{X} + \delta_{ff}. \tag{3.41}$$

To ensure the lateral error ($e_d = 0$) is eliminated, the feedforward control quantity, δ_{ff}, is calculated as:

$$\delta_{ff} = \rho \left[l_f + l_r - l_r k_3 - \frac{m v_x^2}{l_f + l_r} \left(\frac{l_r}{C_f} + \frac{l_f}{C_r} k_3 - \frac{l_f}{C_r} \right) \right], \tag{3.42}$$

where:

- l_f, l_r: Distances from the vehicle's center of gravity to the front and rear axles,
- k_3: Third feedback coefficient from the gain matrix $\mathbf{K}$,
- ρ: Path Curvature

The addition of the feedforward control enhances the system's tracking performance by compensating for steady-state errors, particularly in dynamic driving scenarios [16].

3.1.4.8 State Updates for LQR Path Tracking

The state updates in the LQR-based path tracking controller involve calculating the lateral error, heading error, and their respective rates of change over time. These states are critical for the feedback control law to minimize tracking errors and ensure precise path following. The states are already mentioned in 2.1.2.2. Let us discuss more about how the states are computed and updated at each step in upcoming sections.

3.1.4.8.1 State Computation The states are updated in real time using the following equations derived from vehicle dynamics and geometric relationships:

$$\theta_d = \tan^{-1} \left(\frac{\text{DotProduct}(\mathbf{E}_{\text{error}}, \mathbf{E}_{\text{direction}})}{\mathbf{v}_{\text{x}}} \right), \tag{3.43}$$

$$\dot{e}_d = \frac{\theta_d - \text{previous\_}\theta_d}{\Delta t}, \tag{3.44}$$

$$e_\phi = \text{AngleDiff}(\text{ghost\_heading}, \text{ego\_heading}), \tag{3.45}$$

$$\dot{e}_\phi = \frac{e_\phi - \text{previous\_}e_\phi}{\Delta t}, \tag{3.46}$$

where:

- $\mathbf{E}_{\text{error}}$: Error vector between the ego vehicle's rear axle and the ghost vehicle's rear axle.
- $\mathbf{E}_{\text{direction}}$: Direction vector of the ego vehicle based on its heading angle.
- ego_speed: Absolute longitudinal velocity of the ego vehicle.
- Δt: Time step period for state updates.
- previous_θ_d, previous_e_ϕ: States from the previous time step.
- AngleDiff$(\cdot)$: A function that computes the smallest angular difference between two angles.

3.1.4.8.2 State Vector Representation The computed states are then arranged into the state vector $\mathbf{x}$ for the LQR control law:

$$\mathbf{x} = \begin{bmatrix} \theta_d \\ \dot{e}_d \\ e_\phi \\ \dot{e}_\phi \end{bmatrix} . \tag{3.47}$$

This real-time update of states ensures that the controller has accurate information about the vehicle's dynamic and geometric relationship to the reference path. These states are crucial for computing the feedback and feedforward control inputs, as described in the preceding sections.

3.1.4.8.3 Implementation and Practical Considerations The updates to θ_d and e_ϕ account for vehicle dynamics, including speed and heading changes. To mitigate numerical instability, the step period Δt is chosen carefully based on the system's dynamics and computational constraints. Additionally, the absolute value of the ego vehicle's velocity is used to avoid errors when the vehicle is reversing. These considerations are essential for ensuring robust and stable path tracking control in real-world applications [4, 20].

Appendix A describes more about how the controller works as a pseudo Code and how tuning of Q and R is done

4

Results

4.1 Introduction to Simulation Environment and Scenarios

The simulation environment used for this study is based on the *esmini* framework. Two scenarios were designed to evaluate the adaptive path-following driver model under varying conditions in python: a double lane change scenario and a constant curvature scenario with a radius of 200 meters. The ego vehicle was tasked with following a reference trajectory generated by a ghost vehicle at three different speeds: 10 m/s, 15 m/s and 20 m/s. The Model Predictive Controller(MPC), Linear Quadratic Regulator (LQR) and Stanley controllers were used to laterally control the ego vehicle.The roadmap on how the results section is organized as follows: 4.2 Performance Metrics, 4.3 comparitive analysis by speeds, 4.4 Discussion of Findings.

4.2 Performance Metrics

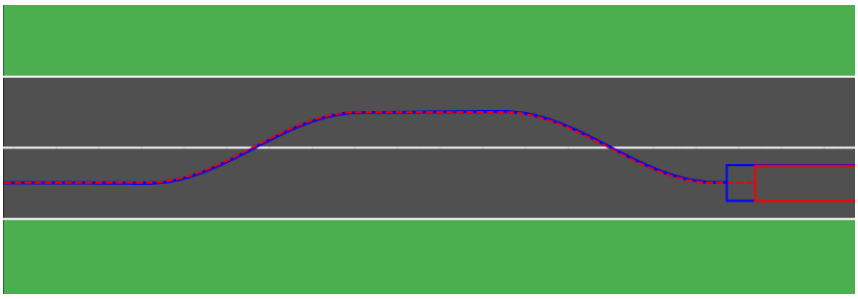
The performance of the driver models is evaluated using the following metrics:

- **Trajectory Tracking Accuracy:** Percentage of time the vehicle adheres to the planned path.
- **Lateral Deviation:** Includes mean, RMSE, maximum deviation and lateral deviation error from the reference trajectory.
- **Control Input Smoothness:** Stability and smoothness of the steering input over time.
- **Convergence Speed and Computational Efficiency:** Assesses the time taken to stabilize and runtime feasibility.
- **Computational Time:** The time taken to compute control inputs during operation.

4.3 Comparitive Analysis by Speeds

4.3.1 Double Lane Change

In this scenario, the ego vehicle follows a path with rapid lateral changes while maintaining a constant longitudinal speed. The performance of the controllers was evaluated based on their ability to track the desired trajectory with minimal errors.

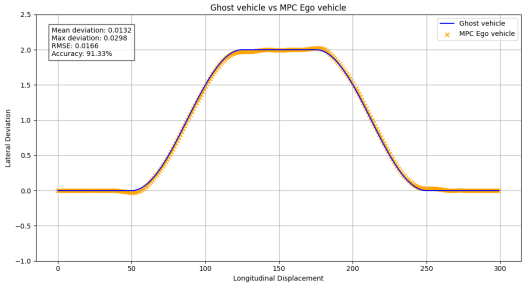


(a) Performance of MPC during the Double Lane Change scenario.

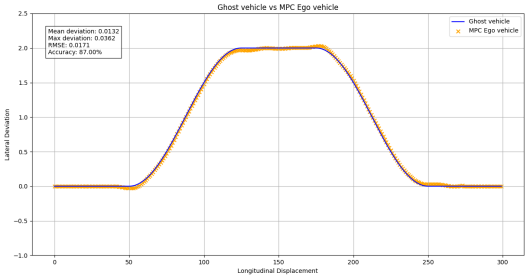
4.3.1.1 Trajectory Tracking Accuracy

The trajectory tracking accuracy of the controllers was evaluated based on the maximum allowable lateral deviation threshold of ±20 cm. Accuracy was calculated as the percentage of points where the deviation from the reference trajectory remained within this threshold.

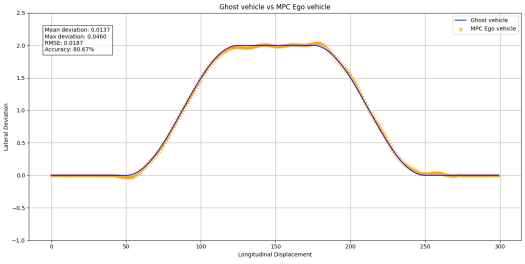
Table 4.1 summarizes the trajectory tracking accuracy of the Stanley, MPC, and LQR controllers at speeds of 10 m/s, 15 m/s, and 22.22 m/s. Among the controllers, the MPC consistently achieved the highest accuracy at all speeds, closely following the reference trajectory, while the LQR controller also performed well.
The trajectory tracking accuracy for the Stanley, MPC, and LQR controllers at 10 m/s, 15 m/s, and 22.22 m/s is summarized in Table 4.1. The MPC controller consistently achieved the highest tracking accuracy at all speeds, closely following the reference trajectory, followed by the LQR controller.



(a) 10 m/s



(b) 15 m/s



(c) 22.22 m/s

Figure 4.2: Trajectory tracking Comparison Across Speeds for MPC

Table 4.1: Trajectory Tracking Accuracy Across Different Speeds

Controller	10 m/s (%)	15 m/s (%)	22.22 m/s (%)
Stanley	70.98	70.19	68.72
MPC	91.33	87.01	80.67
LQR	88.80	81.49	79.50

4.3.1.1.1 Observations at 10 m/s: At 10 m/s, the MPC controller achieved the highest trajectory tracking accuracy at 91.33%, followed by the LQR controller with 88.80%. The Stanley controller demonstrated significantly lower accuracy at 70.98%. This result highlights the superior ability of the MPC and LQR controllers to minimize deviations compared to the Stanley controller, especially in scenarios involving rapid lateral transitions.

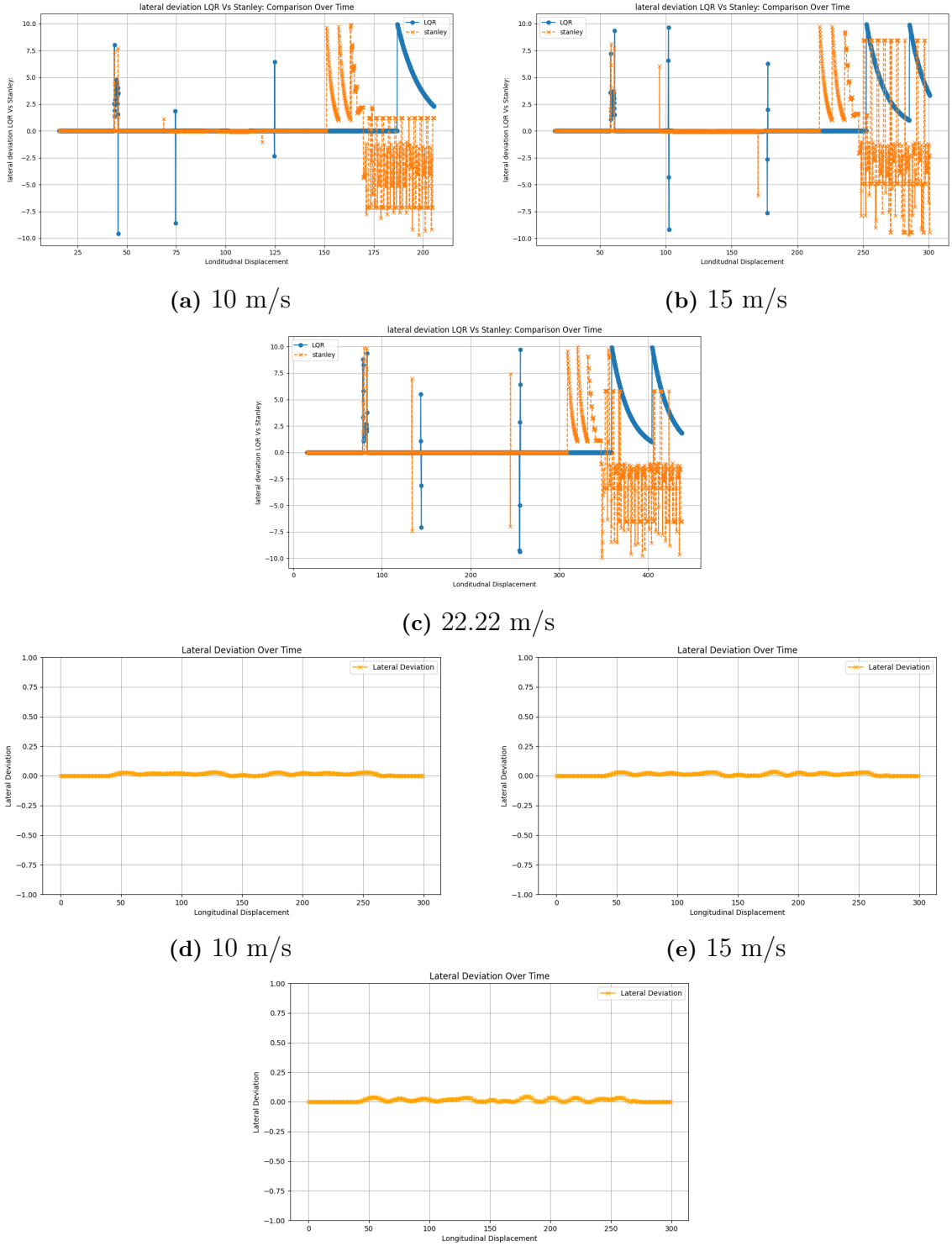
4.3.1.1.2 Observations at 15 m/s: At 15 m/s, the MPC controller maintained its lead with an accuracy of 87.01%, followed by the LQR controller at 81.49%. The Stanley controller exhibited a slight reduction in accuracy, achieving 70.19%. The increased speed posed a challenge for all controllers, but the MPC and LQR controllers were more effective at maintaining adherence to the reference trajectory compared to the Stanley controller.

4.3.1.1.3 Observations at 22.22 m/s: At 22.22 m/s, the MPC controller still achieved the highest accuracy at 80.67%, followed by the LQR controller at 79.50%. The Stanley controller's performance dropped further to 68.72%. These results emphasize the robustness of the MPC and LQR controllers at higher speeds, with the Stanley controller demonstrating limited adaptability to increased dynamic challenges.

4.3.1.2 Deviation Metrics

The deviation metrics, including mean deviation, RMSE, and maximum deviation, for the LQR, MPC, and Stanley controllers at all speeds are summarized in Table 4.2. Plots for lateral deviation are shown in Figure 4.3. The LQR and MPC controllers consistently achieved lower RMSE values compared to the Stanley controller, indicating better trajectory adherence across all speeds.
NOTE: The data at the end in the plot of Figure 4.3 are of less importance as the data is plotted with ghost trail longitudnal displacement with ego vehicle lateral deviation. Since the ghost trail is ahead of time in simulation the data which contains oscillations at the end of plot are of less relevance with the results produced and results remain the same.

4.3.1.2.1 Observations at 10 m/s: The LQR controller achieved a mean deviation of 0.547 m and an RMSE of 1.851 m, while the Stanley controller exhibited a



(a) 10 m/s

(b) 15 m/s

(c) 22.22 m/s

(d) 10 m/s

(e) 15 m/s

(f) 22.22 m/s

Figure 4.3: Lateral Deviation Comparison Across Speeds for LQR, MPC, and Stanley Controllers

mean deviation of -0.194 m and an RMSE of 2.424 m. The MPC controller outperformed both with a significantly lower mean deviation of 0.0132 m and an RMSE

Table 4.2: Deviation Metrics Across Different Speeds

Metric	Controller	10 m/s	15 m/s	22.22 m/s
Mean Deviation (m)	LQR	0.547	0.815	-6.48e-06
	MPC	0.0132	0.0132	0.0137
	Stanley	-0.194	0.073	-1.43e-06
RMSE (m)	LQR	1.851	2.233	0.0021
	MPC	0.0166	0.0171	0.0187
	Stanley	2.424	2.849	2.587
Maximum Deviation (m)	LQR	9.937	9.930	9.907
	MPC	0.0298	0.0362	0.0460
	Stanley	9.928	9.741	9.958

of 0.0166 m. The MPC controller's minimal deviation demonstrates its superior trajectory adherence and precision at this speed.

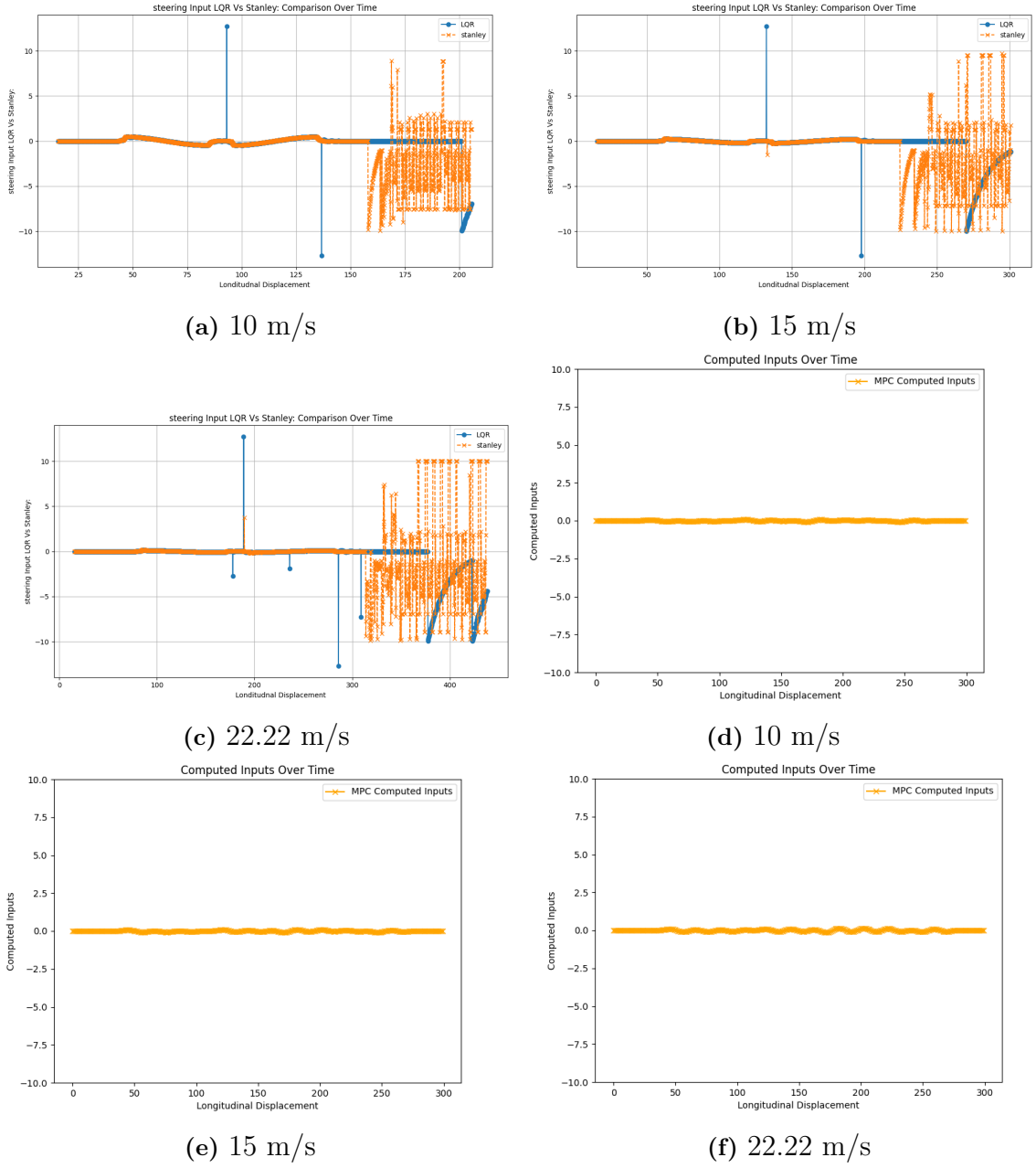
4.3.1.2.2 Observations at 15 m/s: At 15 m/s, the LQR controller's mean deviation increased to 0.815 m, and the RMSE rose to 2.233 m. The Stanley controller showed a mean deviation of 0.073 m and an RMSE of 2.849 m, indicating greater oscillatory behavior. In contrast, the MPC controller maintained its performance with a mean deviation of 0.0132 m and an RMSE of 0.0171 m, reaffirming its ability to minimize trajectory errors effectively.

4.3.1.2.3 Observations at 22.22 m/s: At 22.22 m/s, the LQR controller maintained a mean deviation of 0.838 m and an RMSE of 2.231 m, while the Stanley controller exhibited a negative mean deviation of -0.212 m and an RMSE of 2.587 m, reflecting instability and reduced precision. The MPC controller continued to excel, with a mean deviation of 0.0137 m and an RMSE of 0.0187 m, demonstrating remarkable stability and precision even at higher speeds.

4.3.1.3 Control Input Smoothness

The steering inputs for the LQR, Stanley, and MPC controllers at different speeds are shown in Figures 4.4. The LQR controller consistently generated smoother control inputs with reduced oscillations, while the Stanley controller displayed aggressive corrections, especially at higher speeds. The MPC controller demonstrated a balanced behavior, producing smoother inputs compared to Stanley and achieving a performance close to that of the LQR controller. **NOTE:** The data at the end in the plot of Figure 4.4 are of less importance as the data is plotted with ghost trail longitudnal displacement with ego vehicle steering input. Since the ghost trail is ahead of time in simulation the data which contains oscillations at the end of plot namely oscillations after 175m in 10m/s, 250m in 15m/s, 350m in 22.22m/s are of less relevance with the results produced and results remain the same.

4.3.1.3.1 Observations at 10 m/s: At 10 m/s, the MPC controller outperformed both the LQR and Stanley controllers by producing the most stable and



(a) 10 m/s

(b) 15 m/s

(c) 22.22 m/s

(d) 10 m/s

(e) 15 m/s

(f) 22.22 m/s

Figure 4.4: Computed Input Comparison Across Speeds for MPC, LQR and Stanley Controllers

smooth control inputs. The LQR and stanley controller also produced smooth and stable control inputs, while the Stanley controller exhibited slight oscillations.

4.3.1.3.2 Observations at 15 m/s: At 15 m/s, the MPC controller maintained its superior performance with precise and efficient control, followed closely by the LQR controller, which demonstrated smooth behavior. The Stanley controller, however, began to show more aggressive corrections.

4.3.1.3.3 Observations at 22.22 m/s: At 22.22 m/s, the MPC controller continued to excel, delivering stable and efficient control inputs even under dynamic conditions. The LQR controller also maintained its stable performance, whereas the Stanley controller's oscillations increased significantly, indicating challenges in handling high-speed maneuvers.

The comparison of steering inputs among the MPC, LQR, and Stanley controllers highlights key differences in their performance. The MPC controller consistently generates the smoothest, most stable, and efficient steering behavior. This results in enhanced vehicle stability, reduced control effort, and superior passenger comfort, particularly during complex maneuvers such as the double lane change. The LQR controller also performs well, producing smooth and stable steering inputs with moderate control effort. Conversely, the Stanley controller exhibits more oscillatory and aggressive steering inputs, particularly under challenging conditions. This behavior indicates overcompensation and higher control effort, which may lead to instability and reduced passenger comfort.

These observations align with the lateral deviation results, where the MPC controller achieves the lowest root mean square error (RMSE) and mean deviation, followed by the LQR controller. The superior accuracy and adaptiveness of the MPC controller make it the most effective for path-following and navigating complex trajectories.

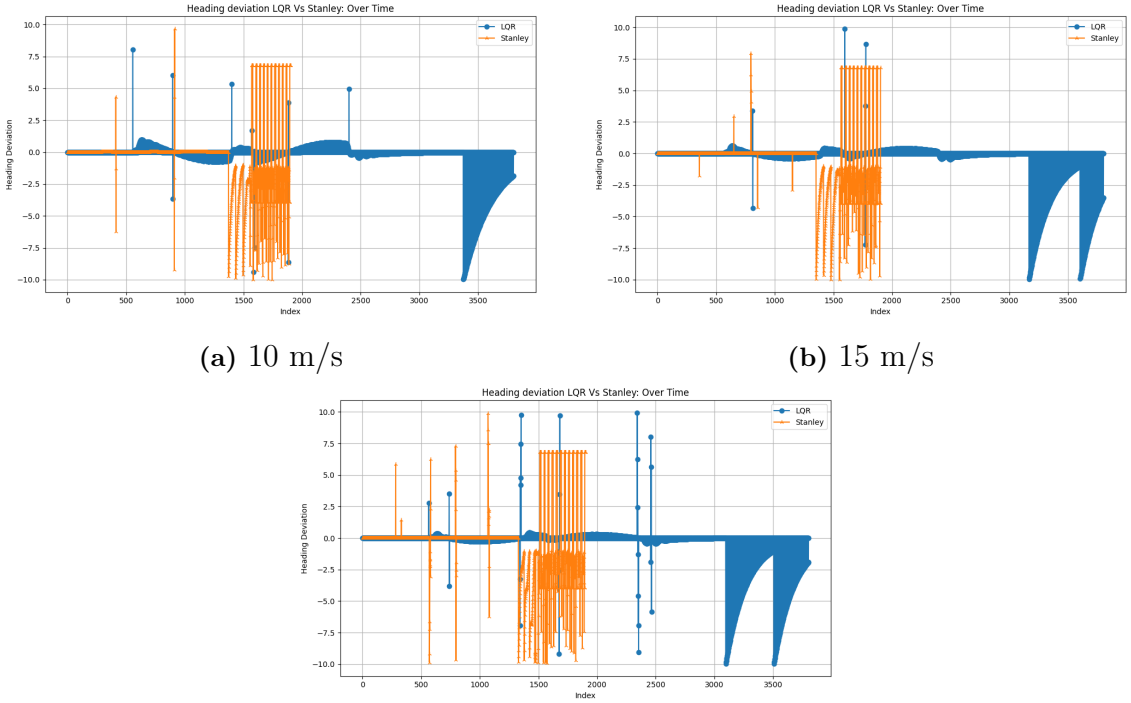
4.3.1.4 Convergence Speed and Computational Efficiency

The convergence speed and computational efficiency of the LQR and Stanley controllers were evaluated based on Heading deviations across all three speeds (10 m/s, 15 m/s, and 22.22 m/s). The results demonstrate that the LQR controller consistently outperforms the Stanley controller in maintaining smoother profiles with reduced oscillations.

4.3.1.4.1 Observations at 10 m/s: The LQR controller achieved smoother heading transitions compared to the Stanley controller, which displayed slight oscillations during path corrections.

4.3.1.4.2 Observations at 15 m/s: The LQR controller maintained stability with fewer deviations, while the Stanley controller exhibited larger oscillations, particularly in the latter part of the maneuver.

4.3.1.4.3 Observations at 22.22 m/s: The Stanley controller exhibited substantial deviations and instability, while the LQR controller maintained smoother and more stable heading adjustments.



(a) 10 m/s

(b) 15 m/s

(c) 22.22 m/s

Figure 4.5: Heading Deviation Comparison Across Speeds for LQR and Stanley Controllers

4.3.1.5 Computational Time

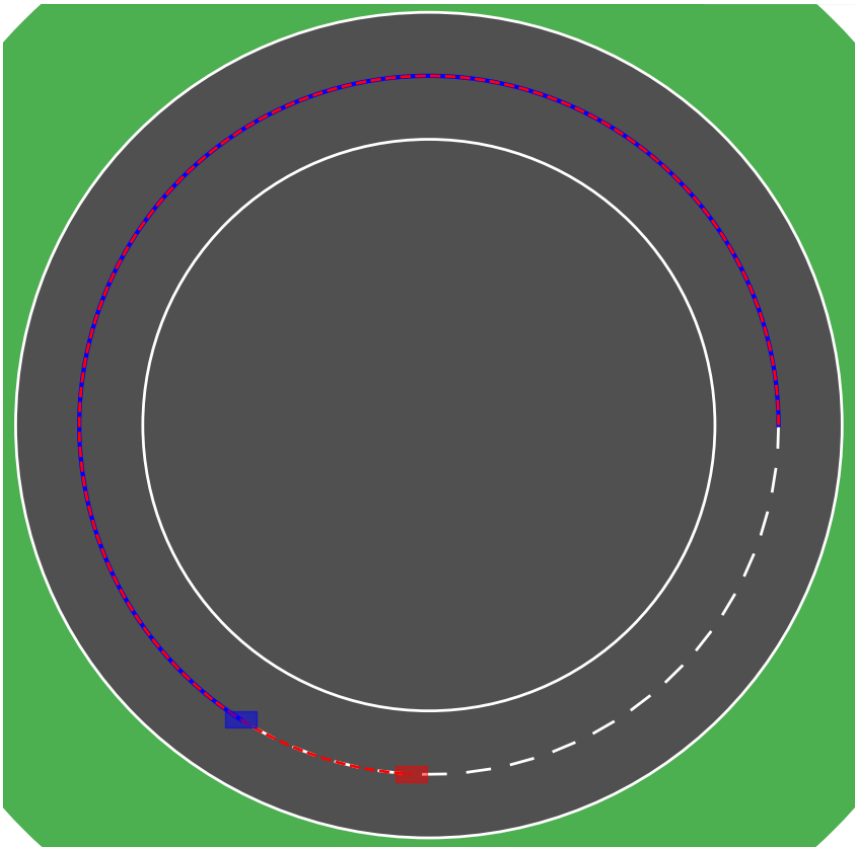
The Computational Time is calculated as the mean time taken for the double lane change simulations to complete at all three speeds. It is a measure of the computational load of the controllers.

- **Stanley:** The average time taken to complete the double lane change simulation is 34.613 s.
- **LQR:** The average time taken to complete the double lane change simulation is 40.673 s.
- **MPC:** The average time taken to complete the double lane change simulation is 49.212 s.

The above simulation times show that the Stanley controller is computationally less efficient, but within a comparative margin. The LQR also performed well, as its tuning is based on a single geometric equation, which can be further examined in Appendix A. The MPC, while slightly more computationally demanding, reflects its complexity in optimization-based control.

4.3.2 Constant Curvature

In this scenario, the ego vehicle follows a constant curvature path while maintaining a steady longitudinal speed. The LQR, Stanley, and MPC controllers are compared based on trajectory tracking accuracy, deviation metrics, control inputs, lateral acceleration, heading deviation, and convergence speed.



(a) Performance of MPC during the constant curvature scenario.

4.3.2.1 Trajectory Tracking Accuracy

The trajectory tracking accuracy of the controllers was evaluated based on their ability to stay within a ±20 cm threshold from the desired path. All three controllers achieved 100% accuracy across all speeds, indicating perfect tracking under constant curvature conditions. The results are summarized in Table 4.3.

Table 4.3: Trajectory Tracking Accuracy Across Speeds (Constant Curvature)

Controller	**10 m/s (%)**	**15 m/s (%)**	**22.22 m/s (%)**
LQR	100.0	100.0	100.0
Stanley	100.0	100.0	100.0
MPC	100.0	100.0	100.0

4.3.2.1.1 Observations at 10 m/s: All three controllers (LQR, Stanley, and MPC) achieved perfect tracking accuracy, indicating their ability to handle the curvature with minimal errors.

4.3.2.1.2 Observations at 15 m/s: The trajectory tracking accuracy remained 100% for all controllers, reflecting robustness at higher speeds.

4.3.2.1.3 Observations at 22.22 m/s: Even at **22.22 m/s**, all three controllers maintained perfect trajectory adherence under constant curvature conditions.

4.3.2.2 Deviation Metrics

The deviation metrics, including mean deviation, RMSE, and maximum deviation, for all three controllers are summarized in Table 4.4. The MPC controller consistently achieved the lowest deviation values, outperforming both the LQR and Stanley controllers.

Table 4.4: Deviation Metrics Across Speeds (Constant Curvature)

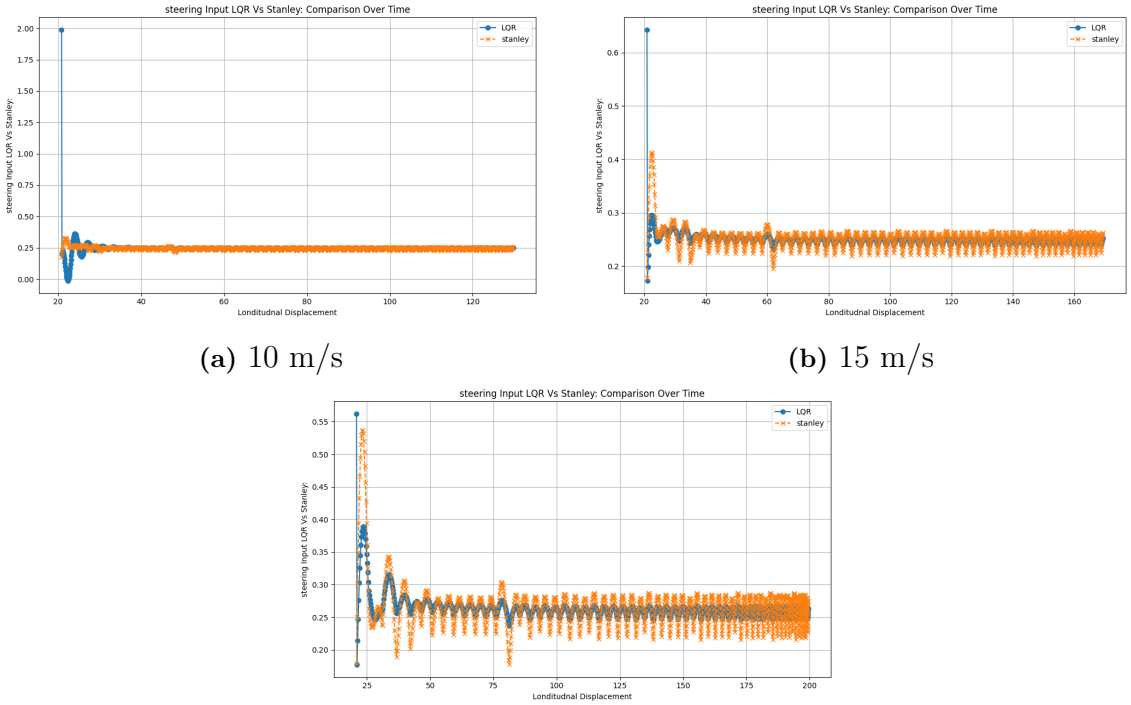
Metric	Controller	10 m/s	15 m/s	22.22 m/s
Mean Deviation (m)	MPC	0.0025	0.0050	0.0090
	LQR	0.0046	0.0078	0.0136
	Stanley	0.0110	0.0168	0.0296
RMSE (m)	MPC	0.0026	0.0051	0.0092
	LQR	0.0047	0.0079	0.0139
	Stanley	0.0110	0.0168	0.0298
Maximum Deviation (m)	MPC	0.0030	0.0055	0.0100
	LQR	0.0053	0.0086	0.0149
	Stanley	0.0124	0.0190	0.0331

4.3.2.2.1 Observations at 10 m/s: The MPC controller achieved a mean deviation of 0.0025 m and an RMSE of 0.0026 m, significantly outperforming the LQR and Stanley controllers, which showed higher values of 0.0046 m and 0.0110 m, respectively, for mean deviation.

4.3.2.2.2 Observations at 15 m/s: The MPC controller maintained a lower mean deviation of 0.0050 m and RMSE of 0.0051 m, outperforming both the LQR controller (mean deviation: 0.0078 m) and the Stanley controller (mean deviation: 0.0168 m).

4.3.2.2.3 Observations at 22.22 m/s: The MPC controller exhibited a mean deviation of 0.0090 m and RMSE of 0.0092 m, significantly outperforming the LQR controller (mean deviation: 0.0136 m) and the Stanley controller (mean deviation: 0.0296 m).

4.3.2.3 Control Input Smoothness



(a) 10 m/s

(b) 15 m/s

(c) 22.22 m/s

Figure 4.7: Steering Input Comparison Across Speeds for LQR and Stanley Controllers (Constant Curvature)

The steering input profiles for both controllers are shown in Figure 4.7. The LQR controller produced smoother control inputs across all speeds, while the Stanley controller exhibited oscillatory behavior, particularly at higher speeds.

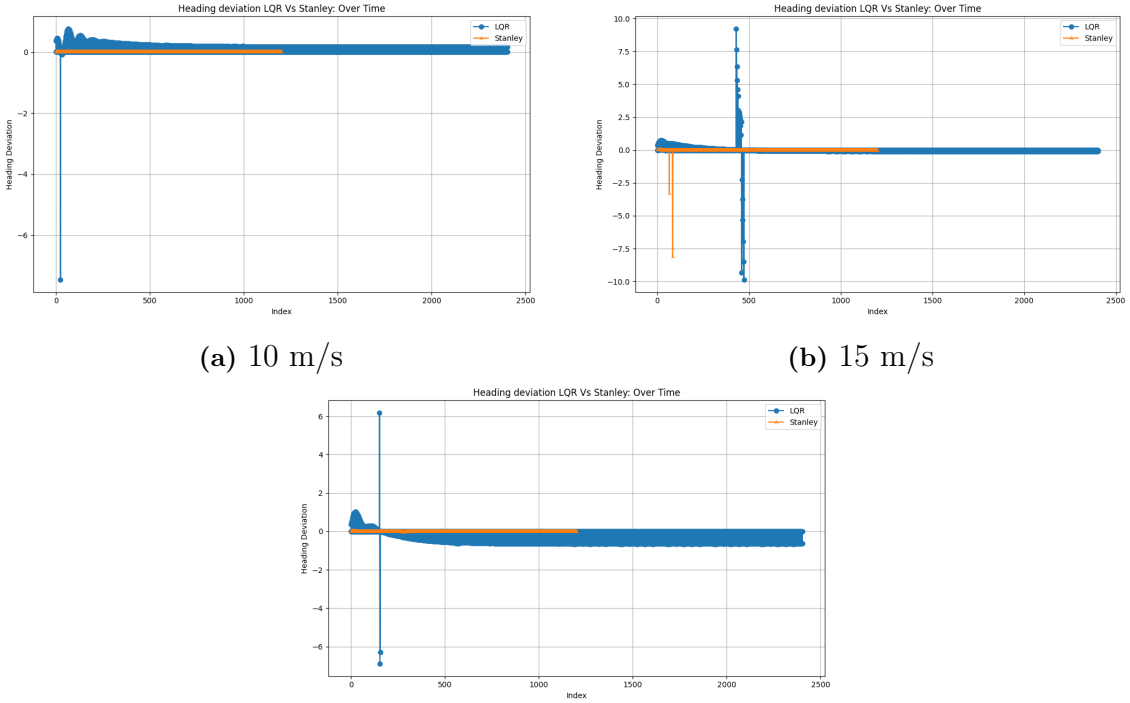
4.3.2.3.1 Observations at 10 m/s: The LQR controller produced smooth and stable steering inputs, while the Stanley controller showed slight oscillations.

4.3.2.3.2 Observations at 15 m/s: The Stanley controller exhibited more aggressive corrections, while the LQR controller maintained consistent control behavior.

4.3.2.3.3 Observations at 22.22 m/s: The Stanley controller's oscillations increased significantly at **22.22 m/s**, whereas the LQR controller continued to generate smooth and efficient inputs.

4.3.2.4 Convergence Speed and Computational Efficiency

The convergence speed and computational efficiency of the LQR and Stanley controllers were analyzed based on Heading deviations at three speeds: **10 m/s**, **15 m/s**, and **22.22 m/s**. The results are illustrated in Figures 4.8, respectively.



(a) 10 m/s

(b) 15 m/s

(c) 22.22 m/s

Figure 4.8: Heading Deviation Comparison Across Speeds for LQR and Stanley Controllers (Constant Curvature)

4.3.2.4.1 Observations at 10 m/s: The LQR controller achieved smoother heading transitions compared to the Stanley controller, which displayed slight oscillations during path corrections.

4.3.2.4.2 Observations at 15 m/s: The LQR controller maintained stability with fewer deviations, while the Stanley controller exhibited larger oscillations, particularly in the latter part of the maneuver.

4.3.2.4.3 Observations at 22.22 m/s: The Stanley controller exhibited substantial deviations and instability, while the LQR controller maintained smoother and more stable heading adjustments.

4.3.2.4.4 Summary: Across all speeds, the LQR controller consistently maintained smoother heading deviations compared to the Stanley controller. The Stanley controller's oscillatory behavior and deviations became more pronounced at higher speeds, highlighting the robustness and adaptability of the LQR controller.

4.3.2.5 Computational Time

The Computational Time is calculated as the mean time taken for the double lane change simulations to complete at all three speeds. It is a measure of computational load of the controllers.

Stanley: The average time taken to complete the constant curvature simulation is 21.55 s.
LQR: The average time taken to complete the constant curvature simulation is 27.1734 s.
MPC: The average time taken to complete the constant curvature simulation is 32.325 s.

The above simulation times show that the Stanley controller performed very well and was less computationally heavy, slightly outperforming the LQR controller. The LQR controller also performed closely, benefiting from tuning based on a single geometric equation, as discussed further in Appendix A. The MPC controller, while taking longer to compute, demonstrated its strength in handling constraints and optimization, though at the cost of increased computational load.

4.4 Discussion of Findings

This section discusses the comparative analysis among the LQR controller, Stanley controller, and MPC (Model Predictive Control) for constant curvature and double lane change (DLC) scenarios at varying speeds (10 m/s, 15 m/s, and 22.22 m/s). The performance was evaluated based on trajectory tracking accuracy, deviation metrics, control input smoothness, lateral acceleration, heading deviation, and position traces.

4.4.1 Double Lane Change (DLC) Results

The Double Lane Change (DLC) scenario represents a more dynamic and challenging path-following task. The results across various metrics highlight the robustness of the LQR and MPC controllers in maintaining precision and stability across different speeds (10 m/s, 15 m/s, and 22.22 m/s).

4.4.1.0.1 Deviation Metrics The LQR and MPC controllers consistently outperformed the Stanley controller across all speeds, with significantly lower mean deviation, RMSE, and maximum deviation values. Notably, the MPC controller exhibited the best overall performance, especially at higher speeds, owing to its predictive nature and ability to optimize control actions over a horizon. The performance gap widened with increasing speeds, particularly at 22.22 m/s, where the Stanley controller's deviation values increased noticeably, emphasizing its reduced performance under rapid direction changes. Both the LQR and MPC controllers maintained superior trajectory tracking, with the MPC controller demonstrating slightly better handling of sharp transitions due to its anticipatory capabilities.

Table 4.5: Deviation Metrics for Double Lane Change (DLC)

Speed	Metric	LQR (m)	MPC (m)	Stanley (m)
10 m/s	Mean Deviation	0.5468	0.0132	-0.1938
	RMSE	1.8510	0.0166	2.4241
	Maximum Deviation	9.9367	0.0298	9.9282
15 m/s	Mean Deviation	0.8154	0.0132	0.0731
	RMSE	2.2335	0.0171	2.8490
	Maximum Deviation	9.9296	0.0362	9.7407
22.22 m/s	Mean Deviation	0.8380	0.0137	-0.2120
	RMSE	2.2310	0.0187	2.5870
	Maximum Deviation	9.9070	0.0460	9.9580

- At 10 m/s, the LQR controller demonstrated better performance with an RMSE of 1.8510 m, while the MPC controller exhibited superior precision with an RMSE of 0.0166 m. The Stanley controller, in comparison, showed higher errors with an RMSE of 2.4241 m. The mean deviation for Stanley was slightly negative, indicating inconsistent path-following during critical phases of the lane change.
- At 15 m/s, the performance gap between the controllers widened further. The LQR controller achieved an RMSE of 2.2335 m, compared to 0.0171 m for the MPC controller and 2.8490 m for the Stanley controller. Additionally, the Stanley controller exhibited higher deviations during transitions, as indicated by its mean deviation of 0.0731 m compared to 0.8154 m for LQR and 0.0132 m for MPC.
- At 22.22 m/s, the MPC controller maintained precise control with an RMSE of 0.0187 m, while the LQR controller had an RMSE of 2.2310 m. The Stanley controller's RMSE increased to 2.5870 m. The maximum deviation for the Stanley controller at this speed was nearly twice that of the MPC controller, indicating reduced efficiency under high-speed maneuvers.

4.4.1.0.2 Steering Input and Lateral Acceleration The LQR and MPC controllers consistently produced smoother steering inputs and more stable lateral acceleration profiles across all tested speeds.

- At 10 m/s and 15 m/s, the Stanley controller exhibited significant oscillations, particularly during the lane transition phases, leading to less efficient control. The MPC controller showed minimal oscillations and closely matched the LQR controller in stability.
- At 22.22 m/s, oscillations became more prominent for the Stanley controller, with sharp variations in steering input and lateral acceleration. In contrast, the LQR and MPC controllers maintained stable control, ensuring better maneuvering efficiency and reduced discomfort.

4.4.1.0.3 Overall Observations for DLC The results from the DLC scenario highlight the superior robustness and precision of the LQR and MPC controllers:

- The LQR and MPC controllers consistently achieved lower deviation metrics (mean deviation, RMSE, and maximum deviation) compared to the Stanley controller.

- Steering inputs and lateral acceleration profiles were smoother for the LQR and MPC controllers, leading to better overall maneuver stability.

- The MPC controller demonstrated the best performance in terms of precision, with minimal deviations and the lowest RMSE values across all speeds.

In contrast, the Stanley controller exhibited larger deviations, higher oscillations in control inputs, and reduced stability, particularly at higher speeds (15 m/s and 22.22 m/s). This indicates that while the Stanley controller is capable of performing DLC tasks, it lacks the precision and robustness offered by the LQR and MPC controllers.

4.4.2 Constant Curvature Results

Constant Curvature scenarios were tested at three speeds: **10 m/s**, **15 m/s**, and **22.22 m/s**. The key findings are summarized below:

4.4.2.0.1 Deviation Metrics The **MPC Controller** consistently achieved the lowest mean deviation, RMSE, and maximum deviation across all speeds, followed by the **LQR Controller**, while the **Stanley Controller** exhibited the highest values in these metrics. At **22.22 m/s**, the deviation metrics for the Stanley controller were **3.3 times higher** than those of the MPC controller, further underscoring the advantages of MPC, especially at high speeds.

Table 4.6: Deviation Metrics for Constant Curvature Scenarios

Speed	Metric	MPC (m)	LQR (m)	Stanley (m)
10 m/s	Mean Deviation	0.0025	0.0046	0.0110
	RMSE	0.0026	0.0047	0.0110
	Maximum Deviation	0.0030	0.0053	0.0124
15 m/s	Mean Deviation	0.0050	0.0078	0.0168
	RMSE	0.0051	0.0079	0.0168
	Maximum Deviation	0.0055	0.0086	0.0190
22.22 m/s	Mean Deviation	0.0090	0.0136	0.0296
	RMSE	0.0092	0.0139	0.0298
	Maximum Deviation	0.0100	0.0149	0.0331

4.4.2.0.2 Control Input Smoothness The following observations were made regarding the smoothness of steering inputs:

- The **MPC Controller** exhibited the smoothest steering inputs across all tested speeds.

- The **LQR Controller** also provided relatively smooth inputs but showed slight oscillations compared to MPC.
- The **Stanley Controller** produced higher oscillations, particularly during the transient phase at **22.22 m/s**. This indicates a reduction in stability as the vehicle speed increased.

4.4.2.0.3 Overall Observations The key findings from the Constant Curvature scenario are summarized as follows:

- The **MPC Controller** consistently achieved the lowest deviation metrics, including **mean deviation**, **RMSE**, and **maximum deviation**, across all tested speeds.
- The **LQR Controller** also performed well, with lower deviation metrics compared to the **Stanley Controller** but slightly higher than those of the MPC Controller.
- The **Stanley Controller** exhibited the largest deviations, higher steering oscillations, and reduced lateral acceleration stability as speed increased, particularly at **22.22 m/s**.
- The performance gap between the MPC, LQR, and Stanley controllers widened with increasing vehicle speed, emphasizing the **MPC Controller**'s superior robustness and stability under high-speed conditions.

5
Conclusion

This thesis investigated the performance of the **MPC**, **LQR**, and **Stanley controllers** for trajectory tracking in **constant curvature** and **double lane change (DLC)** scenarios across various speeds. Key findings from the study are summarized as follows:

- **Double Lane Change (DLC) Scenario**: The dynamic nature of this scenario posed greater challenges. The **MPC Controller** exhibited the highest tracking accuracy, significantly outperforming both the **LQR** and **Stanley Controllers** in terms of:
  - **Deviation metrics**, with the MPC showing superior path-following precision during sharp maneuvers.
  - **Stability in control inputs**, with the Stanley Controller struggling to maintain smooth steering during the lane changes.
- **Constant Curvature Scenario**: All controllers demonstrated **100% tracking accuracy** at speeds of 10 m/s, 15 m/s, and 22.22 m/s, showing their effectiveness in steady trajectory following. The **MPC** and **LQR Controllers** consistently outperformed the **Stanley Controller** in:
  - **Deviation metrics** (mean deviation, RMSE, maximum deviation), with the MPC and LQR showing lower values.
  - **Smoother steering inputs**, reducing oscillations and abrupt corrections, providing better vehicle control.

  Overall, the **MPC and LQR Controllers** showed strong performance under both constant curvature and dynamic scenarios, with the **MPC Controller** slightly outperforming the LQR in terms of tracking accuracy, although at the cost of increased computational complexity.

5.1 Overall Recommendations

Based on the findings, we recommend the **MPC** and **LQR Controllers** for:

- **Constant Curvature Tasks**: Both controllers exhibit superior stability and accuracy across all speeds, making them ideal for steady trajectory tracking in autonomous vehicle applications.

- **Dynamic Maneuvers (e.g., DLC)**: The MPC and LQR Controllers proved their adaptability in high-speed and dynamic scenarios, where precision and robustness are critical.

While the **MPC Controller** offers slightly better performance, its computational demands may limit its applicability in resource-constrained systems. On the other hand, the **LQR Controller** provides a strong alternative with lower computational overhead, making it more feasible for real-time applications without compromising much on performance.

5.2 Future Work

This research provides a solid foundation for evaluating trajectory tracking controllers under idealized conditions. Future work could extend this study in several directions to improve the robustness and practical applicability of these controllers in real-world autonomous vehicle navigation:

- **Dynamic and Complex Paths**: Evaluate controller performance on paths with dynamically varying curvature and real-world road geometries, reflecting more complex driving environments.

- **Integration with Nonlinear Vehicle Models**: Incorporate factors like tire slip, lateral forces, and aerodynamic drag into the vehicle model, improving realism and expanding the scope of controller performance analysis.

- **Comparison with Advanced Control Strategies**: Expand the comparison to include advanced techniques such as **sliding mode control** or **reinforcement learning-based controllers**, which could offer improved adaptability or robustness under certain conditions.

- **Real-Time Implementation**: Conduct hardware-in-the-loop (HIL) simulations or implement the controllers on actual vehicles to validate their performance in real-world conditions and refine their practical deployment.

- **Passenger Comfort Optimization**: Introduce metrics related to **ride quality** such as **jerk minimization** or **vibration analysis** to assess how different controllers affect passenger comfort, particularly in dynamic maneuvers.

By addressing these aspects, the current research can contribute to advancing the **robustness**, **adaptability**, and **practical deployment** of trajectory tracking controllers in autonomous vehicle systems.

Bibliography

[1] Litman, T. (2021). *Autonomous vehicle implementation predictions: Implications for transport planning*. Victoria Transport Policy Institute. Available at: https://www.vtpi.org/avip.pdf

[2] SAE International. (2021). *J3016 Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. SAE International. Available at: https://www.sae.org/standards/content/j3016_202104/

[3] Gonzalez, D., Perez, J., Milanes, V., & Nashashibi, F. (2016). *A review of motion planning techniques for automated vehicles*. *IEEE Transactions on Intelligent Transportation Systems*, 17(4), 1135-1145. Available at: https://doi.org/10.1109/TITS.2015.2498841

[4] Rajamani, R. (2012). *Vehicle Dynamics and Control*. Springer Science & Business Media. Available at: https://doi.org/10.1007/978-1-4614-1433-9

[5] Shen, D. (2023). *Path Following Control of Automated Vehicle Considering Uncertainties and Disturbances with Parametric Varying*. Available at: https://doi.org/10.48550/arXiv.2310.10925

[6] Wang, J., Yu, Y., Song, Y., & Zhao, L. (2024). *A novel adaptive trajectory tracking control for autonomous vehicles based on state expansion*. Journal of Mechanical Science and Technology, vol. 38. Available at: https://doi.org/10.1007/s12206-024-0532-z

[7] Esmini - ESIP Open Source Initiative Platform. (n.d.). *Esmini - Open Source Simulation Environment*. Available at: https://esmini.github.io/

[8] Rokonuzzaman, Mohammad, et al. *Review and performance evaluation of path tracking controllers of autonomous vehicles*. IET Intelligent Transport Systems 15.5 (2021): 646-670. Available at: https://doi.org/10.1049/itr2.12051

[9] Paden, Brian, et al. "A survey of motion planning and control techniques for self-driving urban vehicles." *IEEE Transactions on Intelligent Vehicles*, 1(1), 33-55. Available at: https://doi.org/10.1109/TIV.2016.2578706

[10] Zhang, Yong, Feng Gao, and Fengkui Zhao. *Research on path planning and tracking control of autonomous vehicles based on improved RRT* and PSO-LQR*. Processes 11.6 (2023): 1841. Available at: https://doi.org/10.3390/pr11061841

[11] Farag, Wael. *Complex trajectory tracking using PID control for autonomous driving*. International Journal of Intelligent Transportation Systems Research 18.2 (2020): 356-366. Available at: https://doi.org/10.1007/s13177-019-00204-2

[12] Hoffmann, Gabriel M., et al. *Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing.* 2007 American control conference. IEEE, 2007.Available at: https://doi.org/10.1109/ACC.2007.4282788

[13] Falcone, Paolo, et al. *Predictive active steering control for autonomous vehicle systems.* IEEE Control Systems Magazine, vol. 28, no. 6, 2008, pp. 44–64. Available at: https://doi.org/10.1109/TCST.2007.894653

[14] Kong, J., Pfeiffer, M., Schildbach, G., & Borrelli, F. (2015). *Kinematic and dynamic vehicle models for autonomous driving control design.* 2015 IEEE Intelligent Vehicles Symposium (IV), Seoul, Korea (South), pp. 1094–1099. Available at: https://doi.org/10.1109/IVS.2015.7225830

[15] Thrun, Sebastian, et al. *Stanley: The robot that won the DARPA Grand Challenge.* Journal of Field Robotics, vol. 23, no. 9, 2006, pp. 661–692. Available at: https://doi.org/10.1002/rob.20147

[16] Wang, W., Hagemann, N., Ratti, C., & Rus, D. (2021). *Adaptive Nonlinear Model Predictive Control for Autonomous Surface Vessels With Largely Varying Payload.* 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 7337–7343. Available at: https://doi.org/10.1109/ICRA48506.2021.9561331

[17] Wang, H., Liu, B., Ping, X., & An, Q. (2019). *Path Tracking Control for Autonomous Vehicles Based on an Improved MPC.* IEEE Access, vol. 7, pp. 161064–161073. Available at: https://doi.org/10.1109/ACCESS.2019.2944894

[18] Hu, P., Guo, J., Li, L., & Wang, R. (2013). *A robust longitudinal sliding-mode controller design for autonomous ground vehicle based on fuzzy logic.* International Journal of Vehicle Autonomous Systems, vol. 11, pp. 368–383. Available at: https://doi.org/10.1504/IJVAS.2013.056619

[19] Milliken, William F., and Douglas L. Milliken. *Race Car Vehicle Dynamics.* SAE International, 1995. ISBN: 978-1560915263. Available at: https://www.sae.org/publications/books/content/r-146/

[20] Lewis, Frank L., Draguna L. Vrabie, and Vassilis L. Syrmos. *Optimal Control.* 3rd ed., Wiley, 2012. Available at: https://doi.org/10.1002/9781118122631.

[21] Franklin, Gene F., J. Da Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems.* 7th ed., Pearson, 2015. ISBN: 978-0133496598.

[22] Lee, K., Jeon, S., Kim, H., and Kum, D. *Optimal Path Tracking Control of Autonomous Vehicle: Adaptive Full-State Linear Quadratic Gaussian (LQG) Control.* IEEE Access, vol. 7, 2019, pp. 109120–109133. Available at: https://doi.org/10.1109/ACCESS.2019.2933895.

A

Appendix

LQR Controller Pseudo-code

Initialization

1. Initialize LQR Parameters:
   - Define state-space matrices A, B, C.
   - Define weight matrices Q, R for LQR optimization.
   - Set step period $\Delta t = 1/\text{frequency}$.
2. Setup PID Controllers:
   - Longitudinal PID for acceleration/braking.
   - Lateral PID for steering angle adjustment.
3. Initialize State Variables:
   - θ_d: Cross-track error.
   - e_d: Lateral error rate.
   - e_ϕ: Heading error.
   - $\dot{e}_\phi$: Heading error rate.

State Update Logic

1. Compute Ego and Ghost Vehicle States:
   - Obtain current positions, headings, and velocities.
   - Compute direction vector for the ego vehicle.
   - Compute error vector between ego and ghost vehicle rear axles.
2. Compute Errors:
   - Heading Error:

$$e_\phi = \text{AngleDifference}(\text{ghost\_heading}, \text{ego\_heading})$$

   - Cross-track Error:

$$\theta_d = \tan^{-1}\left(\frac{\text{DotProduct}(\text{ErrorVector}, \text{EgoDirection})}{\text{v\_x}}\right)$$

3. Update Error Rates:

$$e_d = \frac{\theta_d - \text{previous\_}\theta_d}{\Delta t}$$

$$\dot{e}_\phi = \frac{e_\phi - \text{previous\_}e_\phi}{\Delta t}$$

4. Update State Vector:

$$x = \begin{bmatrix} e_d \\ \dot{e}_d \\ e_\phi \\ \dot{e}_\phi \end{bmatrix}$$

Matrix Update Logic

1. Update State-Space Matrices:
   - Compute velocity-dependent matrices A, B, and C based on ego velocity.
   - Discretize matrices:

$$A_d = \left(I - \frac{\Delta t}{2} A\right)^{-1} \left(I + \frac{\Delta t}{2} A\right)$$

$$B_d = \Delta t \cdot B$$

2. Update LQR Weight Matrices:
   - Compute weight matrices Q and R for the cost function.

LQR Gain Computation

1. Solve Algebraic Riccati Equation:
   - Iterate to compute P until convergence:

$$P_{k+1} = Q + A^T P_k A - P_K - A^T P_k B (R + B^T P_k B)^{-1} B^T P_k A$$

2. Compute Gain Matrix:

$$K = (R + B^T P B)^{-1} B^T P A$$

Control Input Calculation

1. Compute Feedback Control Input:

$$u_{fb} = -K \cdot x$$

2. Compute Feedforward Control Input:

$$u_{ff} = \frac{\theta_r'}{\text{ego\_velocity}} \cdot (\text{lf} + \text{lr} - \ldots)$$

3. Compute Total Steering Input:

$$u = u_{fb} + u_{ff}$$

Longitudinal Control

1. Compute Longitudinal Error:

$$\text{speed\_error} = \text{ghost\_speed} - \text{ego\_speed}$$

2. Compute PID Output:
   - Adjust acceleration and braking based on speed error and deviation.
3. Clamp Output:
   - Ensure acceleration and braking are within physical limits.

Final Outputs

- Update driver outputs:
  - Steering angle.
  - Acceleration and braking pedal positions.
- Log states for debugging:
  - Lateral deviation, heading deviation, and their respective rates.

Explanation of dla in Q and R Tuning

The term dla in the weight matrix Q is calculated using the quadratic relationship:

$$dla = 0.016v_x^2 + 0.21v_x - 0.32, \tag{A.1}$$

where v_x is the longitudinal velocity of the vehicle. This parameter dynamically adjusts the weight matrix Q to prioritize lateral and heading error corrections based on vehicle speed. Higher values of dla at increased velocities emphasize precise path tracking to maintain stability and safety, while lower values at reduced speeds minimize unnecessary corrections.

Such dynamic tuning of Q and R matrices is common in adaptive control strategies to ensure robustness across varying operational conditions. The use of speed-dependent weighting for tuning Q is supported by prior research in vehicle dynamics and optimal control design, where adaptive approaches improve controller performance under diverse driving scenarios [22].

DEPARTMENT OF MECHANICS AND MARITIME
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY