Enhancing Truck Platooning Efficiency and Safety

Beatriz Baixinho Lourenço

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Rita Maria Mendes de Almeida Correia da Cunha
Prof. Daniel de Matos Silvestre

Examination Committee

Chairperson: Prof. João Luís Da Costa Campos Gonçalves Sobrinho
Supervisor: Prof. Rita Maria Mendes de Almeida Correia da Cunha
Member of the Committee: Prof. Francisco Fernandes Castro Rego

November 2023

Declaration

I declare that this document is an original work of my own authorship and that it fulfils all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgements

Foremost, I am greatly indebted to my thesis supervisor, Professor Daniel Silvestre, for providing me with definite direction, guidance, and constant encouragement. To my co-supervisor Maria Charitidou, whose insights and feedback have added depth and perspective to this research. To Pedro Roque for his valuable suggestion and direction to accomplish this study. I feel incredibly fortunate to have had the guidance and mentorship of such a highly respected academic.

My heartfelt appreciation goes out to my family. Especially to my parents, who surely cannot begin to fathom the profound impact of their presence on this journey. To them, I owe not only their boundless support but, more importantly, their endless patience.

And lastly, to Isabel Portugal, Mariana Toco, Rita Palma, Isabella Luppi, Jonne van Hasstregt, Miguel Garcia Naude, Filip Geib and Gonçalo Silva, my former colleagues who have not only greatly encouraged and inspired me throughout my university journey but continue to do so to this day. I could never thank them enough.

This work was partially supported by the Portuguese Fundação para a Ciência e a Tecnologia (FCT) through project FirePuma (https://doi.org/10.54499/PCIF/MPG/0156/2019), through Institute for Systems and Robotics (ISR), under Laboratory for Robotics and Engineering Systems (LARSyS) project UIDB/50009/2020, and through COPELABS, University Lusófona project UIDB/04111/2020.

Resumo

A condução autónoma surge como uma oportunidade próspera para avanços notáveis no domínio dos sistemas de transporte. A presente dissertação explora a área de frotas de camiões autónomos, focando-se no desenvolvimento de uma abordagem de Controlo Preditivo Não Linear (NMPC), inserida num enquadramento de Cruise Control Adaptativo Cooperativo (CACC). O trabalho desenvolvido aborda alguns desafios críticos no desvio de obstáculos e nas manobras de mudança de faixas em ambiente de estrada.

A contribuição central deste trabalho reside no desenvolvimento e implementação de um algoritmo NMPC inovador adaptado ao controlo de frotas. A estrutura deste integra uma restrição de penalização para garantir o desvio de obstáculos e manter a coesão da frota, otimizando simultaneamente os sinais de controlo em tempo real. Várias experiências, incluindo cenários de desvios de obstáculos estáticos e dinâmicos, validam a eficácia da abordagem proposta. Em todas as experiências realizadas, os veículos seguem-se uns aos outros de perto, resultando em trajetórias suaves para todos os estados do sistema e para os sinais de entrada. Além disso, no caso de uma travagem abrupta pelo veículo líder, a frota de camiões mantém-se coesa. O NMPC proposto demonstra ser computacionalmente eficiente quando comparado com o estado da arte.

Palavras-chave: Frotas de Camiões, Controlo Predictivo Não Linear (NMPC), Cruise Control Adaptativo Cooperativo (CACC), Desvio de Obstáculos

Abstract

The advent of autonomous driving technologies has paved the way for notable advancements in the realm of transportation systems. This thesis explores the dynamic field of truck platooning, focusing on the development of a Nonlinear Model Predictive Control (NMPC) approach within a Cooperative Adaptive Cruise Control (CACC) framework. The research tackles the critical challenges in obstacle avoidance and lane-changing manoeuvres.

The core contribution of this work lies in the development and implementation of a novel NMPC algorithm tailored to platoon control. This framework integrates a penalty soft constraint to guarantee obstacle avoidance and maintain platoon coherence while optimising control inputs in real-time. Several experiments, including static and dynamic obstacle avoidance scenarios, validate the efficacy of the proposed approach. In all experiments, the vehicles closely follow one another, resulting in smooth trajectories for all system states and control input signals. Even in the event of abrupt braking by the ego vehicle, the platoon remains cohesive. Moreover, the proposed NMPC proves to be computationally efficient when compared to the state-of-the-art.

Keywords: Truck Platooning, Nonlinear Model Predictive Control (NMPC), Cooperative Adaptive Cruise Control (CACC), Obstacle Avoidance

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

This document is incomplete. The external file associated with the glossary 'symbols' (which should be called `main.sls`) hasn't been created.

This has probably happened because there are no entries defined in this glossary. Did you forget to use `type=symbols` when you defined your entries? If you tried to load entries into this glossary with `\loadglsentries` did you remember to use `[symbols]` as the optional argument? If you did, check that the definitions in the file you loaded all had the type set to `\glsdefaulttype`.

This message will be removed once the problem has been fixed.

ACC Adaptive Cruise Control
APF Artificial Potential Field
CACC Cooperative Adaptive Cruise Control
CBF Control Barrier Function
CLF Control Lyapunov Function
FoV Field of View
GPU Graphics Processing Unit
ICR Instantaneous Center of Rotation
IP Interior-Point Method
KF Kalman Filter
LCA Lane Change Assistant
LiDAR Light Detection and Ranging
LKA Lane Keeping Assistant
MPC Model Predictive Control
MVEE Minimum-Volume Enclosing Ellipsoid
NMPC Nonlinear Model Predictive Control
NLP Nonlinear Programming
ODE Ordinary Differential Equations
OCP Optimal Control Problem
PID Proportional-Integral-Derivative Controller
ROS Robotic Operating System
RK4 Runge-Kutta 4th Order

RViz ROS Visualization Tool

URDF Unified Robot Description Format

V2V Vehicle-to-Vehicle Communication

Chapter 1

Introduction

1.1 Motivation

The realm of autonomous driving has already begun to prove its potential to reshape the way we interact with vehicles [**bimbraw_autonomous_2015**]. However, amidst the ongoing discourse surrounding automation, the broader impact on transportation systems often escapes our attention.

In recent years, the transportation industry has witnessed significant advancements aimed at improving the efficiency and safety of road-based freight transport systems. Among these advancements, truck platooning has emerged as a promising solution. A platoon formation refers to a coordinated and semi-autonomous convoy of vehicles travelling in close proximity [**porfiri_environmental_2006**]. By maintaining a short distance between the vehicles, platooning aims to leverage the benefits of reduced aerodynamic drag, resulting in improved fuel efficiency and consequently lower emissions [**tsugawa_review_2016**]. Additionally, platooning is reported to enhance safety and ease traffic congestion [**lee_impact_2021**].

In 2017, the European Truck Platooning Challenge marked a pivotal initiation of efforts in this direction [**european_truck_platooning_challenge_european_nodate**]. Furthermore, research stemming from the event revealed a potential reduction of up to 15% in fuel consumption through truck platooning, concurrently dispelling more conservative apprehensions regarding autonomous vehicle safety. Following such, the EU truck platoon road-map [**noauthor_eu_2017**] envisions the deployment of multi-brand platooning technology by 2025.

Moreover, these advancements in truck platooning hold significant relevance in the context of the freight and distribution sector, which is increasingly grappling with the critical issue of a shortage of qualified professional drivers [**ji-hyland_what_2020**]. As of 2021, approximately 10% of the total truck driver positions in Europe, equivalent to 425,000 vacancies, remained unfilled. The situation is expected to worsen, with a projected increase to 14% of unfilled positions by the end of 2022 [**noauthor_one_2023**]. In the broader context of logistics and transportation, where transportation costs often represent the most substantial portion of overall expenses, this shortage is causing ripple effects across multiple industries.

Despite its potential benefits, the implementation of truck platooning presents several challenges, including platoon coordination and communication reliability. In the pursuit of refining existing strategies, studies have delved into the intricacies of platoon shape, rearrangement, and formation dynamics, as exemplified by the work of **maiti_impact_2020** [**maiti_impact_2020**]. Furthermore, a parallel line of research has studied communication protocols, striving to engineer seamless and reliable mechanisms capable of addressing the challenges posed by the variable range between vehicles, as demonstrated in the investigation by **won_l-platooning_2022** [**won_l-platooning_2022**]. These studies collectively highlight some of the intricate factors that must be managed to successfully deploy truck platooning.

Additionally, one of the significant challenges in such a strategy is ensuring the safe and efficient navigation of platoons, particularly during dynamic scenarios such as obstacle avoidance and lane-changing manoeuvres. This underscores the critical need for the development of effective control strategies that can pave the way for platooning deployment in real-world contexts. Notably, Model Predictive Control (MPC) emerges as a fundamental control strategy because of its ability to handle constrained multi-variable systems [**hernandez_real-time_2016**]. Using a model of the dynamics, MPC generates solutions that minimise a cost function using both the future control actions and states.

Building upon this landscape of autonomous driving and platooning challenges, this thesis addresses the problem of truck platooning navigation involving obstacle avoidance and lane-changing manoeuvres. Through the lens of a NMPC approach within a CACC framework on a preceder-leader follower topology, this work seeks to achieve safe and efficient platoon navigation.

1.2 Contributions

This thesis makes a significant impact on the field of autonomous vehicle technology and transportation systems, particularly in the context of truck platooning. The key contributions of this work are outlined as follows:

- Introduction of an NMPC approach for guiding trucks along predefined paths, involving lane-changing manoeuvres with the aid of LiDAR data while adhering to formation and safety requirements;
- The proposed method exhibits computational efficiency, making it suitable for real-time applications in scenarios such as obstacle avoidance and overtaking;
- A comparative performance evaluation against a state-of-the-art approach for dynamic obstacle avoidance, implemented in ROS, which demonstrates its practical applicability in real-world situations;
- Providing insights into the real-world deployment of truck platooning by combining theoretical exploration, advanced control strategies, and practical implementation, thereby granting valuable information on the feasibility of deploying platooning solutions within real-world transportation systems.

1.3 Thesis Outline

This dissertation is structured in seven chapters, including the present one. Chapter 1 provides an overview of the motivation and objectives of the thesis. Chapter 2 delves into prior relevant research, setting the foundation for the work: ranging from vehicle modelling to the introduction of control strategies and multi-agent systems. Chapter 3 provides a comprehensive analysis of the current state-of-the-art of existing control strategies and their implications. In Chapter 4 it is presented the methodology, including a description of the proposed solution and details about its implementation. Chapter 5 rigorously assesses the proposed solution by subjecting it to various scenarios. The results obtained are critically analysed in Chapter 6, emphasising insights gained and potential implications. Lastly, Chapter 7 summarises the findings of this dissertation, acknowledges the limitations of the methods employed, and outlines promising directions for future research.

Chapter 2

Background

This chapter presents the mathematical and conceptual preliminaries required for the rest of the thesis, thus providing a comprehensive and robust foundation for readers to grasp the detailed research work that follows.

2.1 Vehicle Modelling

Understanding the motion of a vehicle is critical for developing an effective control system. While human drivers have an intuitive understanding of how turning the steering wheel affects the vehicle's rotation, conveying this information to a controller requires a mathematical approach.

There are two primary approaches to represent a vehicle: *dynamic* and *kinematic* modelling [**dandrea-novel_modelling_1991**]. A dynamical model evolves from force balances, while the latter uses velocity constraints. In the case of wheeled vehicles, these velocity constraints make up a *nonholonomic model*, which naturally arises when assuming the wheels of the vehicle roll without slipping.

A nonholonomic system can generally be represented as a nonlinear system [**mcloskey_nonholonomic_1993**]. This representation can be expressed as follows:

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

Here, the state vector $\boldsymbol{x}$ is typically chosen to represent the vehicle's configuration while $\boldsymbol{u}$ represents the various possible inputs such as steering, braking, and acceleration.

The effectiveness of model-based control is greatly influenced by the accuracy of the models employed. While more accurate models can provide more precise predictions, they can also be computationally expensive. One reason advanced vehicle models with higher fidelity are not commonly used for planning is the high-dimensional state space they imply. In practical applications, efficient motion planning modules require models with lower state dimensions, leading to the common use of kinematic models.

The point mass model, as depicted in Figure 2.1, offers a simplified representation of a vehicle's motion.

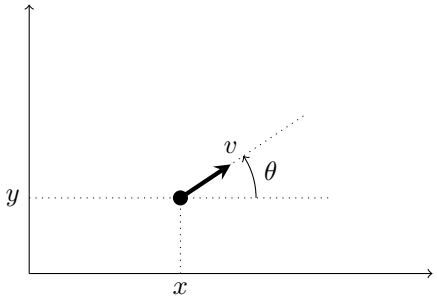


Figure 2.1: Illustration of the point mass model representation. The model simplifies the vehicle's motion by considering it as a single point in space.

This model assumes the vehicle moves without any rotational or translational motion other than along its path. From a computational perspective, we would want a model as simple as possible. And in that matter, the point mass model proves advantageous.

The state model captures the fundamental relationship between the vehicle's speed, heading angle, and its motion along the x and y directions. It can be represented mathematically as:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \end{bmatrix} \tag{2.2}$$

The current model lacks the ability to account for rotation or orientation. The point mass model focuses solely on the vehicle's translational motion along the x and y directions, disregarding any changes in its heading or yaw angle. To incorporate a sense of orientation into the model, we need to consider a more sophisticated representation.

2.1.1 Kinematic unicycle model

One widely used model that introduces the notion of orientation is the kinematic unicycle model. Unlike the point mass model, the kinematic unicycle model considers the vehicle as a rigid body with a well-defined orientation. This model captures the essential dynamics of a vehicle's motion, including both translational and rotational aspects, providing a more comprehensive representation.

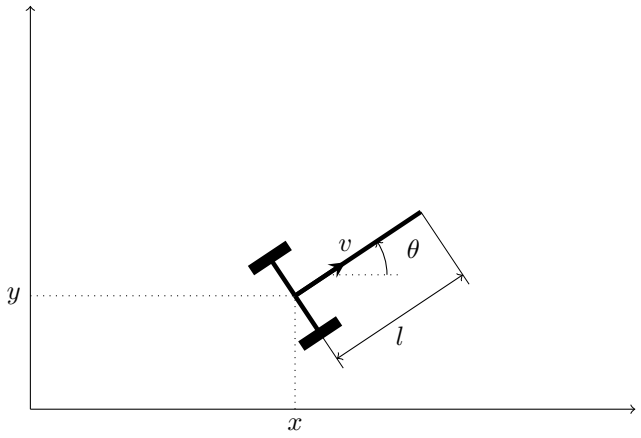


Figure 2.2: A schematic representation of a kinematic unicycle model. Inspired and adapted from **dandrea-novel_modelling_1991** (**dandrea-novel_modelling_1991**) [**dandrea-novel_modelling_1991**].

A unicycle is a vehicle with a single orientable wheel [**wu_lidar_2022**]. Its configuration is described by the position of the wheel contact point and the wheel orientation. In the context of a state space model, the state vector represents the variables that define the system's state, while the input vector represents the control inputs that affect the system. For a unicycle model, the state vector defined as $\boldsymbol{x} = \begin{bmatrix} x & y & \theta \end{bmatrix}^\top$ where x and y represent the position of the wheel contact point, and θ represents the wheel orientation. The input vector can be defined as $\boldsymbol{u} = \begin{bmatrix} v & \dot{\theta} \end{bmatrix}^\top$ where v represents the linear velocity of the unicycle, and $\dot{\theta}$ represents the angular velocity of the wheel.

To model the dynamics of the unicycle, we can write the complete state space model in matrix form:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \dot{\theta} \tag{2.3}$$

However, if we want our controller to be aware of the steering/tire angle, we need to introduce an additional element to represent the tire's movement relative to the current stick model. This introduces the concept of a kinematic bicycle model.

2.1.2 Kinematic bicycle model

The kinematic bicycle model is an advanced representation of a vehicle's motion, incorporating steering behaviour [**pereira_linear_2018**]. This model assumes that the vehicle operates on a flat surface and utilises front-wheel steering with perfect Ackermann steering geometry. In Ackermann steering, the turning radius of the inner front wheel is smaller than that of the outer front wheel, resulting in a specific relationship between the steering angles.

The Ackermann mechanism satisfies the following relation:

$$\kappa = 1/R = \tan(\alpha)/l \tag{2.4}$$

where κ is the car-like vehicle's curvature, α represents the steering angle of the front wheels, l denotes the wheelbase of the vehicle, and R signifies the turning radius of the vehicle's centre of rotation.

The concept of the Instantaneous Center of Rotation (ICR) is essential to understanding the kinematic behaviour of the vehicle. The instantaneous centre of rotation is a virtual point around which the vehicle appears to rotate instantaneously at a given moment. It lies at the intersection of the imaginary lines connecting the rotation axes of the front wheels, as depicted in Figure 2.3.

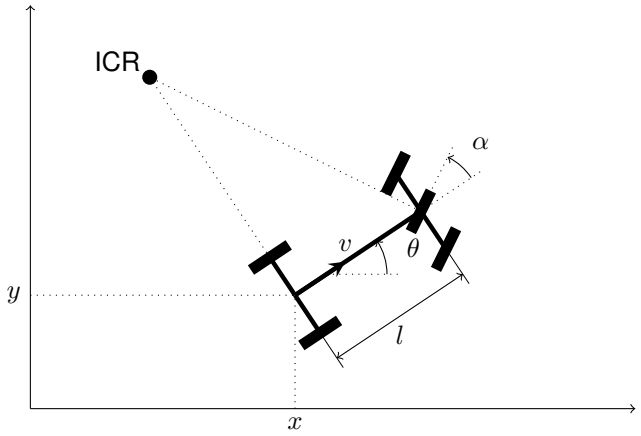


Figure 2.3: A schematic representation of a kinematic bicycle model. Inspired and adapted from **paden_survey_2016** (**paden_survey_2016**) [**paden_survey_2016**].

To model the kinematic behaviour, we define the state space as follows:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \dfrac{\tan\alpha}{l} \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \alpha \tag{2.5}$$

where $\boldsymbol{x}$ represents the state vector consisting of the vehicle's position (x, y), orientation θ, and steering angle α, and $\boldsymbol{u}$ represents the control input vector with longitudinal velocity v and α.

In the kinematic bicycle model, the vehicle's motion is governed by the equations above, which account for the influence of both the longitudinal and steering dynamics. This model provides a more realistic representation of the vehicle's behaviour compared to the simplified point mass model, making it suitable for various trajectory planning and control applications.

2.1.3 Kinematic model for the general n-trailer

In the context of modelling trucks and heavy-duty vehicles, a kinematic model known as the general n-trailer model is commonly employed [**moradi_non-linear_2022**] [**ljungqvist_motion_2020-1**]. This model provides a comprehensive representation of the motion and dynamics of these vehicles, taking into account the complex interactions between the tractor and multiple trailers.

The system consists of $n + 1$ vehicle segments, which include a leading car-like tractor connected to n passive trailers. The connections between the segments can be either on-axle or signed off-axle hitch

connections. Each vehicle segment has a segment length $l_i > 0$ and a signed hitching offset m_i. The sign of m_i is positive when the connection is located behind the preceding vehicle segment's axle. If the system has a mixture of hitching types, it is referred to as a general n-trailer. On the other hand, if the hitching is either entirely on-axle or off-axle, it is called a standard n-trailer or a non-standard n-trailer, respectively [**michalek_trailer-maneuverability_2020**].

The kinematic bicycle model is only capable of emulating the leading tractor, while the trailers according to the standard unicycle kinematics when connected passively.

Let us now consider a single-track n-trailer kinematic structure comprising a tractor and an arbitrary number of n single-axle trailers, all interconnected by passive rotary joints as depicted in Figure 2.4.

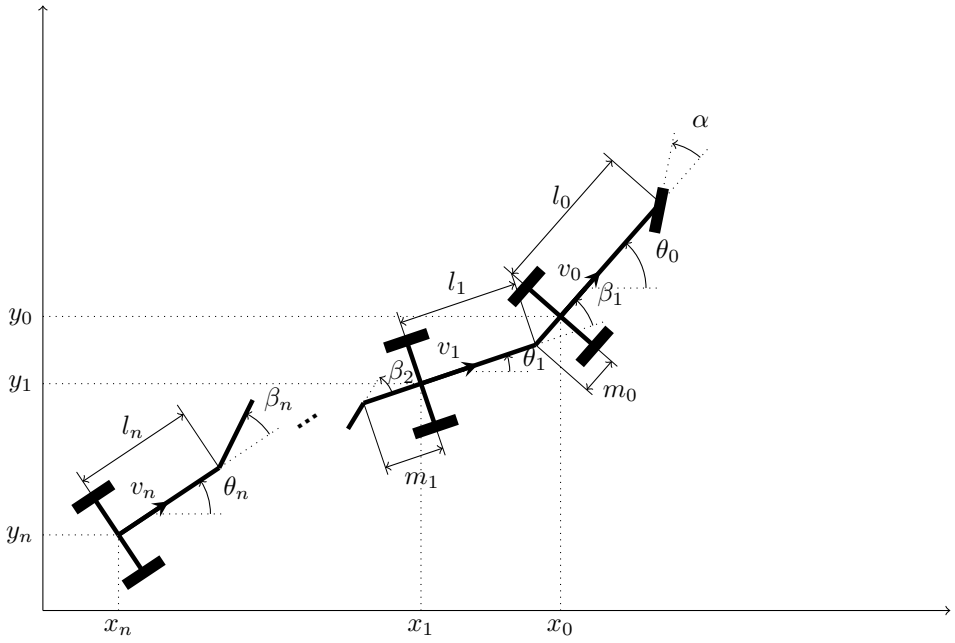


Figure 2.4: A schematic description of the general n-trailer with a car-like tractor. The system consists of a leading car-like tractor that is connected to n passive trailers with a mixture of on-axle and off-axle hitch connections. Inspired and adapted from **lukassek_model_2021** (**lukassek_model_2021**) [**lukassek_model_2021**].

The kinematic model of the general n-trailer system results in the nonholonomic model, i.e. the wheels of the vehicle are assumed to be rolling without slipping, given by

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{x}_0 \\ \dot{y}_0 \\ \dot{v}_0 \\ \dot{\delta}_0 \\ \dot{\theta}_0 \\ \dot{\theta}_1 \\ \vdots \\ \dot{\theta}_n \end{bmatrix} = \begin{bmatrix} v_0 \cos\theta_0 \\ v_0 \sin\theta_0 \\ u_0 \\ u_1 \\ v_0 \dfrac{\tan(\alpha_0)}{l_0} \\ g_1(\beta_1, v_0, \delta_0) \\ \vdots \\ g_n(\boldsymbol{\beta}, v_0, \delta_0) \end{bmatrix} \tag{2.6}$$

where $\boldsymbol{x} = [x_0, y_0, v_0, \delta_0, \theta_0, \ldots, \theta_n] \in \mathbb{R}^3 \times \left(\mathbb{S}^1\right)^{n+2}$, with $\mathbb{S} = (-\pi, \pi]$, represents the state. And $\boldsymbol{u} = [u_0, u_1] \in \mathbb{R}^2$ the control input with acceleration $\dot{v}_0 = u_0$ and steering rate $\dot{\delta}_0 = u_1$.

The global position $[x_0, y_0]^\top$ represents the Cartesian coordinates of the vehicles rear axle midpoint in the fixed world frame. The longitudinal vehicle velocity is denoted as v_0 and the steering angle as δ_0. The states of the trailers are provided by the heading angles $\theta_i, i = 1, \ldots, n$ and each segment position can be calculated by

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} - \sum_{j=1}^{i} l_j \begin{bmatrix} \cos\theta_j \\ \sin\theta_j \end{bmatrix} - \sum_{j=0}^{i-1} m_j \begin{bmatrix} \cos\theta_j \\ \sin\theta_j \end{bmatrix} \tag{2.7}$$

with respect to the vehicle-tractor position $[x_0, y_0]^\top$. The difference in heading angles between each segment, denoted by $\theta_i, i = 1, \ldots, n$, represents the hitching angle, given by

$$\beta_i \triangleq \theta_{i-1} - \theta_i \in \mathcal{B}_i = [-\bar{\beta}_i, \bar{\beta}_i], \quad \bar{\beta}_i \in (0, \pi), \quad i = 1, \ldots, n \tag{2.8}$$

with $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_n]^\top$.

The recursive formula for the transformation of the angular velocity, formulate the velocity transformation as follows and introducing $\boldsymbol{w} = \left[\dot{\theta}_0, v_0\right]^\top$ and $\boldsymbol{c} = [1, 0]^\top$ together with the transformation matrices

$$\boldsymbol{J}_i(\beta_i) = \begin{bmatrix} -\dfrac{m_{i-1}\cos\beta_i}{l_i} & \dfrac{\sin\beta_i}{l_i} \\ m_{i-1}\sin\beta_i & \cos\beta_i \end{bmatrix} \quad i = 1, \ldots, n, \tag{2.9}$$

the functions $g_i(\boldsymbol{\beta}, v_0, \delta_0) : \mathbb{R} \to \mathbb{R}$ are computed recursively

$$\begin{aligned} g_1(\beta_1, v_0, \delta_0) &= \boldsymbol{c}^\top \boldsymbol{J}_1(\beta_1)\, \boldsymbol{w} \\ &\vdots \\ g_n(\boldsymbol{\beta}, v_0, \delta_0) &= \boldsymbol{c}^\top \boldsymbol{J}_n(\beta_n)\, \boldsymbol{J}_{n-1}(\beta_{n-1}) \ldots \boldsymbol{J}_1(\beta_1)\, \boldsymbol{w} \end{aligned} \tag{2.10}$$

The direction of motion is essential for the stability of the system in (2.6), notably, the joint-angle kinematics are structurally unstable in backward motion ($v_0 < 0$), where it risks folding and entering what is called a *jack-knife* state (**goos_hybrid_2002**). Conversely, these modes remain stable during forward motion ($v_0 > 0$), which is the focus of this study. In the case of single-axle hitching only ($m_i = 0$ for $i = 0, \ldots, n$), the general n-trailer model aligns with the standard n-trailer configuration.

2.2 Model Predictive Control

MPC is a control strategy suitable for optimising the performance of constrained systems [**babu_model_2019**]. It is typically formulated as a finite-horizon optimisation problem, where the objective is to find the optimal control inputs over a finite time horizon, subject to the system dynamics

and constraints. Some of its state-of-the-art approaches are registered in [**tiriolo_design_2023**] [**cheng_model-predictive-control-based_2021**].

More traditional control laws like the PID do not explicitly consider state or input constraints and prediction of future state values. Thus, for cyber-physical systems involving potential human injuries, traditional methods do not provide enough performance guarantees. MPC on the other hand, explicitly computes the predicted behaviour over some horizon. An analogy that underlines the distinction is comparing MPC to driving while looking through the windshield, where you anticipate what lies ahead, while PID control resembles driving while looking through the rear-view mirror. It is essential to note that this formulation, while simplistic, highlights the predictive nature of MPC, as it takes into account future system behaviour.

In the next section, NMPC is introduced alongside with Control Lyapunov Function (CLF)s and CBFs, which have been commonly combined to enforce safety constraints.

2.2.1 Nonlinear Model Predictive Control

Nonlinearities appear in real systems and controllers frequently fail to take them into account [**wang_nonlinear_1995**]. NMPC is generally formulated as an online optimisation problem for a system with nonlinear dynamics while satisfying a set of both linear and nonlinear constraints [**allgower_nonlinear_2004**]. In accordance with the feedback strategy known as the receding horizon, the process can be summarised with the following steps:

1. Measure or estimate the current state of the system;
2. Solve the optimisation problem over a time horizon and find the optimal control policy;
3. Apply the first control from the optimal control policy to the system;
4. Repeat the process enabling real-time control and adaptation to changing system conditions.

To formulate the optimisation problem mathematically, let us consider a continuous-time system as in (2.1) with n state variables and m control inputs such that

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \tag{2.11}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$ is the state of the system, $\boldsymbol{u}(t) \in \mathbb{R}^m$ is the control input of the system and $\boldsymbol{f} : \mathbb{R}^n \times \mathbb{R}^m \longrightarrow \mathbb{R}^n$ is a globally Lipschitz continuous function, i.e. $\|\boldsymbol{f}(\mathbf{x}) - \boldsymbol{f}(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and L is the smallest possible constant satisfying the inequality. The nonlinear system satisfies the state and input constraints

$$\begin{aligned} \boldsymbol{x}(t) &\in \mathcal{X}, \quad \forall t \geq 0 \\ \boldsymbol{u}(t) &\in \mathcal{U}, \quad \forall t \geq 0 \end{aligned} \tag{2.12}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ is connected, and $\mathcal{U} \subseteq \mathbb{R}^m$ is compact. The objective of the optimisation problem is to find an optimal control policy $\boldsymbol{u}(\cdot)^*$ while minimising a cost function, that describes the desired performance

of the system, over a time horizon T

$$J\left(\boldsymbol{x}\left(t_0\right), \boldsymbol{u}(\cdot)\right)=\int_{t_0}^{t_0+T} \mathbf{L}(\boldsymbol{x}(t), \boldsymbol{u}(t)) dt \,. \tag{2.13}$$

The objective function should be chosen to translate the desired control objective. When considering reference tracking to a set of points $\boldsymbol{r}$, a common formulation of the objective function is as follows

$$J\left(\boldsymbol{x}\left(t_0\right), \boldsymbol{u}(\cdot)\right)=\int_{t_0}^{t_0+T}\left[\boldsymbol{r}\left(t_0+t\right)-\boldsymbol{x}\left(t_0+t\right)\right]^{\top} \mathbf{Q}\left[\boldsymbol{r}\left(t_0+t\right)-\boldsymbol{x}\left(t_0+t\right)\right] dt+\int_{t_0}^{t_0+T} \dot{\boldsymbol{u}}^{\top}(t) \mathbf{R} \dot{\boldsymbol{u}}(t) dt \,, \tag{2.14}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are controller tuning parameters, i.e. weights in the cost function. The online optimization problem in NMPC is thus structured as follows

$$\begin{aligned} \min_{\boldsymbol{u}(\cdot)} \quad & J\left(\boldsymbol{x}\left(t_0\right), \boldsymbol{u}(\cdot)\right) \\ \text{s.t.} \quad & \dot{\boldsymbol{x}}(t)=\boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad \forall t \in\left[t_0, t_0+T\right], \\ & \boldsymbol{x}\left(t_0\right)=\boldsymbol{x}_0, \\ & \boldsymbol{u}(t) \in \mathcal{U} \qquad\qquad \forall t \in\left[t_0, t_0+T\right], \\ & \boldsymbol{x}(t) \in \mathcal{X} \qquad\qquad \forall t \in\left[t_0, t_0+T\right]. \end{aligned} \tag{2.15}$$

Given that a controller is formally implemented through a digital computer that samples the variables of the system and transmits the control action back at discrete time steps, it is advantageous to consider the system specified in discrete time as

$$\boldsymbol{x}_{k+1}=\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{2.16}$$

where $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ denote the state and the control input vector at instant k, respectively. The discrete-time model is, however, only an approximation of the continuous-time model.

The objective function in this case is often composed by the sum of a staged cost $\boldsymbol{q}$ and a final cost $\boldsymbol{p}$ such that

$$J(\boldsymbol{x}, \boldsymbol{u})=\boldsymbol{p}(\boldsymbol{x}_N)+\sum_{k=0}^{N-1} \boldsymbol{q}\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right), \tag{2.17}$$

where N denote the discrete-time horizon. Thus, the discrete NMPC optimisation problem can be posed as

$$\begin{aligned} \min_{\boldsymbol{u}(\cdot)} \quad & \boldsymbol{p}(\boldsymbol{x}_N)+\sum_{k=0}^{N-1} \boldsymbol{q}\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) \\ \text{s.t.} \quad & \boldsymbol{x}_{k+1}=\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k), \quad k=0, \ldots, N-1, \\ & \boldsymbol{x}\left(0\right)=\boldsymbol{x}_0, \\ & \boldsymbol{x}_N \in \mathcal{X}_f, \\ & \boldsymbol{u}_k \in \mathcal{U}, \qquad k=0, \ldots, N-1, \\ & \boldsymbol{x}_k \in \mathcal{X}, \qquad k=0, \ldots, N-1 \end{aligned} \tag{2.18}$$

here $\boldsymbol{x}_0$ is the starting measured state and $\mathcal{X}_f \subseteq \mathbb{R}^n$ is a *terminal region* that we require the system states

to reach at the end of the horizon.

2.2.2 Control Lyapunov Function

In order to construct the concept of Control Lyapunov Functions, let us first consider the autonomous system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$, with an equilibrium point at $\mathbf{0}$, meaning that $\boldsymbol{f}(\mathbf{0}) = \mathbf{0}$, as established in [**jiang_input–state_2001**]. The following stability concept pertains to the classification of this equilibrium point.

Definition 1 (Stability concepts)**.** *The equilibrium point $\boldsymbol{x} = \mathbf{0}$ is said to be:*

- *stable if $\forall \varepsilon > 0, \exists \delta = \delta(\varepsilon) > 0$, s.t.*

$$\|\boldsymbol{x}(\mathbf{0})\| < \delta \Longrightarrow \|\boldsymbol{x}(t)\| < \varepsilon, \forall t > 0 \tag{2.19}$$

- *unstable, otherwise*
- *asymptotically stable if it is stable and δ can be chosen such that $\|\boldsymbol{x}(\mathbf{0})\| < \delta \Longrightarrow \lim_{t\to\infty} \|\boldsymbol{x}(t)\| = 0$*

Now, the idea behind Lyapunov stability is to introduce positive energy function for the system and prove that it decreases along possible system trajectories [**owens_multi-periodic_2004**].

Theorem 1 (Lyapunov Stability Theorem)**.** *Let $V : D \to \mathbb{R}$ be a continuously differentiable function such that $V(\mathbf{0}) = 0$ and $V(\boldsymbol{x}) > 0$ in $D - \{0\}, \dot{V}(\boldsymbol{x}) \leq 0$ in D. The derivative of $V(\boldsymbol{x})$ along the trajectories of $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$, denoted by $\dot{V}(\boldsymbol{x})$, is given by*

$$\begin{aligned}\dot{V}(\boldsymbol{x}) &= \sum_{i=1}^{n} \frac{\partial V}{\partial x_i}\dot{x}_i \\ &= \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} f_i(x_i) \\ &= \left(\frac{\partial V}{\partial \boldsymbol{x}}\right)^{\top} \boldsymbol{f}(\boldsymbol{x}).\end{aligned}$$

Then, $\boldsymbol{x} = \mathbf{0}$ is stable. Moreover, if $\dot{V}(\boldsymbol{x}) < 0$ in $D - \{0\}$, then $\boldsymbol{x} = \mathbf{0}$ is asymptotically stable.

V is thus referred to as a Lyapunov function. A CLF is an extension of the idea of Lyapunov functions to test whether a system is asymptotically stabilisable, i.e., if for any state $\boldsymbol{x}$ there exists a control $\boldsymbol{u}$ such that the system converges asymptotically to zero. This idea can be translated mathematically in the following definition.

Definition 2. *A Control Lyapunov Function (CLF) is a function $V : D \to \mathbb{R}$ that is continuously differentiable, positive-definite (that is, $V(\boldsymbol{x})$ is positive for all $\boldsymbol{x} \in D$ except at $\boldsymbol{x} = 0$ where it is zero), and such that for all $\boldsymbol{x} \in \mathbb{R}^n (x \neq 0)$, there exists $\boldsymbol{u} \in \mathbb{R}^m$ such that*

$$\dot{V}(\boldsymbol{x}, \boldsymbol{u}) := \langle \nabla V(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \rangle < 0$$

where $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ denotes the inner product of $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$.

Analogously, in a discrete-time domain approach, a candidate CLF such as V must satisfy

$$\Delta V\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) \leq -\alpha_k V\left(\boldsymbol{x}_k\right), 0 < \alpha_k \leq 1, \tag{2.20}$$

where $\Delta V\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) := V\left(\boldsymbol{x}_{k+1}\right) - V\left(\boldsymbol{x}_k\right)$. Thus, the upper bound of a CLF decreases exponentially at time k with the rate $1 - \alpha_k$.

2.2.3 Control Barrier Functions

The CBF method uses a Lyapunov-style argument to render a desired constraint set control invariant, meaning an input can always be applied to keep the system within the safe set for all time. Therefore, a CBF can be used to enforce safety constraints on the states of a dynamical system [**zeng_enhancing_2021**].

Consider a discrete-time control system such that

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right), \tag{2.21}$$

with $\boldsymbol{x} \in \mathcal{X}$ representing the system state with the control input $\boldsymbol{u}$ confined by an admissible input set $\mathcal{U}$. For safety-critical control, considering a set $\mathcal{C}$ defined as the superlevel set of a continuously differentiable function $h : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$,

$$\begin{aligned} \mathcal{C} &= \left\{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) \geq 0\right\}, \\ \partial\mathcal{C} &= \left\{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) = 0\right\}, \\ \mathrm{Int}(\mathcal{C}) &= \left\{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) > 0\right\}, \end{aligned} \tag{2.22}$$

$\mathcal{C}$ can be referred to as the safe set, and can be regarded as the ensemble of states satisfying distance constraints

$$h(\boldsymbol{x}) \geq 0. \tag{2.23}$$

In a stricter manner, the function h becomes a control barrier function in the discrete-time domain if it satisfies the following relation,

$$\Delta h\left(x_k, u_k\right) \geq -\gamma_k h\left(\boldsymbol{x}_k\right), 0 < \gamma_k \leq 1, \tag{2.24}$$

where $\Delta h\left(\boldsymbol{x}_k, \boldsymbol{x}_k\right) := h\left(\boldsymbol{x}_{k+1}\right) - h\left(\boldsymbol{x}_k\right)$. Satisfying such constraint implies $h\left(\boldsymbol{x}_{k+1}\right) \geq \left(1 - \gamma_k\right) h\left(\boldsymbol{x}_k\right)$, i.e, the lower bound of control barrier function $h(\boldsymbol{x})$ decreases exponentially at time k with the rate $1 - \gamma_k$. The CBFs are thus designed to guarantee the forward invariance of the safe set $\mathcal{C}$.

2.3 From Centralised to Distributed Predictive Control

In a platooning scenario the communication and computational cost of implementing a centralised MPC grows with the number of vehicles. Thus, it is attractive to produce a distributed scheme of MPC that both enables autonomy of the individual vehicles and improves tractability [**mishra_centralized_2020**].

This segment introduces the fundamental concepts of algebraic graph theory, which are pivotal for comprehending the upcoming problem statement. Additionally, it sets the stage for discussing control strategies within a multi-agent framework and provides an overview of the leader-follower multi-agent topology.

2.3.1 Graph Theory

Graph theory provides a structured framework for representing and analysing the interactions between the individual vehicles within the platoon.

Consider a connected undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ comprising a set of n vertices $\mathcal{V} := \{1, 2, \cdots, n\}$ and a set of edges $\mathcal{E} = \{(i, j) \in \mathcal{V} \times \mathcal{V} \mid j \in \mathcal{N}_i\}$ [**bullo_lectures_2022**]. Each edge $(i, j) \in \mathcal{E}$ signifies a communicative link between vertices i and j. Here, $m = |\mathcal{E}|$ is the number of edges and $\mathcal{N}_i$ denotes the agents in the neighbourhood of agent i that can communicate with i.

The adjacency matrix $\mathbb{A}$ of $\mathcal{G}$ is the $n \times n$ symmetric matrix whose elements a_{ij} are given by $a_{ij} = 1$, if $(i, j) \in \mathcal{E}$, and $a_{ij} = 0$, otherwise. The degree of vertex i is defined as $d_i = \sum_{j \in \mathcal{N}_i} a_{ij}$. Then the degree matrix is $\Delta = \operatorname{diag}(d_1, d_2, \ldots, d_n)$. The graph Laplacian of $\mathcal{G}$ is $L = \Delta - \mathbb{A}$, this holds particular importance for continuous time systems.

A *path* is a sequence of edges connecting two distinct vertices. A graph is connected if there exists a path between any pair of vertices, and fully connected if every vertex in the graph is directly connected to every other vertex by a unique edge.

By assigning an orientation to each edge of $\mathcal{G}$ we can define the incidence matrix $D = D(\mathcal{G}) = [d_{ij}] \in \mathbb{R}^{n \cdot m}$. The rows of D are indexed by the vertices and the columns are indexed by the edges with $d_{ij} = 1$ if the vertex i is the head of the edge $(i, j), d_{ij} = -1$ if the vertex i is the tail of the edge (i, j) and $d_{ij} = 0$ otherwise. Based on the incidence matrix, the graph Laplacian of $\mathcal{G}$ can be described as $L = DD^\top$ [**mesbahi_graph_2010**]. In addition, $L_e = D^\top D$ is the so-called edge Laplacian and c_{ij} denotes its elements.

2.3.2 Multi-Agent Systems

Multi-agent systems are interconnected control systems, named *agents* [**liu_controllability_2012**]. Vehicle platoons are a prime example of a multi-agent system, where each vehicle acts as an independent agent. This allows each vehicle to make decisions based on local information rather than being dictated by a unique central controller, thus optimising the platoon's performance.

By definition, the dynamics of each agent depend on the agent's state and the states of its neighbouring agents [**liu_controllability_2012**]. Thus, a general multi-agent system can be modelled

$$\dot{\boldsymbol{x}}_i = \boldsymbol{f}\left(\boldsymbol{x}_i, \cup_{j \in \mathcal{N}_i} \boldsymbol{x}_j, \boldsymbol{u}_i\right) \tag{2.25}$$

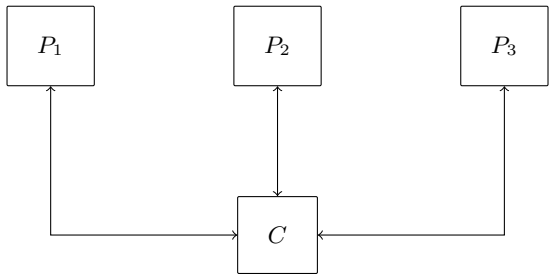
where $\boldsymbol{x}_i$ and $\boldsymbol{u}_i$ is the state and control input of agent i, respectively, and $\mathcal{N}_i$ denotes the neighbour set of agent i.

In control of multi-agent systems is important to distinguish between the different control approaches: *centralised*, *distributed*, and *decentralised* [**wang_distributed_2010**]. And, depending on which control architecture is employed, the control input may depend differently on the agents' states, such as

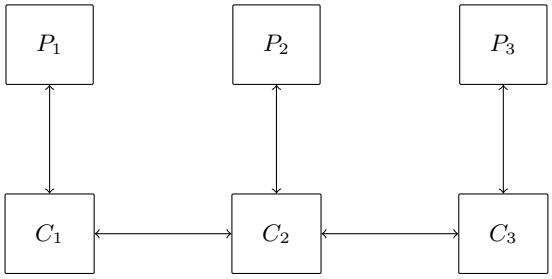
$$\boldsymbol{u}_i = \begin{cases} \boldsymbol{u}_i \left(\cup_{j \in \mathcal{V}} \boldsymbol{x}_j \right) & \text{centralised} \\ \boldsymbol{u}_i \left(\boldsymbol{x}_i, \cup_{j \in \mathcal{N}_i} \boldsymbol{x}_j \right) & \text{distributed} \\ \boldsymbol{u}_i \left(\boldsymbol{x}_i \right) & \text{decentralised} \end{cases} \tag{2.26}$$

where $\mathcal{V}$ denotes the set of all agents.

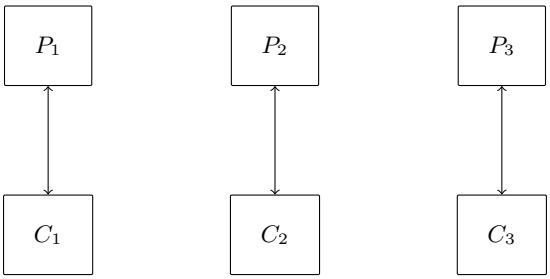
This is to say that: centralised control involves a single central authority that controls the actions of all agents in the system; distributed control takes in multiple agents working together to make decisions, but with a clear hierarchy of authority and finally in decentralised control each agent makes its own decisions based on local information, without the need for a central authority or clear hierarchy. This is better illustrated in 2.5.



(a) Centralized control architecture.



(b) Distributed control architecture.



(c) Decentralised control architecture.

Figure 2.5: Representation of decentralised 2.5a, distributed 2.5b and decentralised 2.5c control architectures. P_1, P_2 and P_3 represent plants controlled by the controller C or by the controllers C_1, C_2 and C_3 respectively.

In truck platooning, distributed control proves advantageous over both centralised and decentralised approaches due to its real-time adjustments based on local information. This strategy balances computational complexity from centralised control and potential synchronisation issues from decentralised control, leading to improved efficiency, safety, and adaptability in platoons.

2.3.3 Leader-follower Multi-agent Systems

Leader-follower dynamics in multi-agent systems has garnered substantial attention due to its capacity to mimic real-world scenarios where certain agents guide the behaviour of others [**he_leaderfollower_2019**]. Primarily, this structure is driven by the notion that the motion of a leader, often adhering to a reference trajectory, can significantly influence the collective behaviour of the entire group. Notably, followers can also ascend to leadership roles for subsequent agents, resulting in a cascading leadership pattern within the network.

Each vehicle belongs to one platoon which consists of one leader and several followers. To not lose generality let us consider a multi-platoon scenario. Let us consider a multi-agent system with vertices $\mathcal{V} = \{1, \ldots, n\}$. We suppose that the first n_f agents are selected as followers while the last n_l agents are selected as leaders with respective vertices set $\mathcal{V}_F = \{1, \ldots, n_f\}, \mathcal{V}_L = \{n_f + 1, \ldots, n_f + n_l\}$ and $n = n_f + n_l$.

We will have a time-varying graph topology which switches among different structures according to the coordination phases, e.g., merging, splitting, etc. The overall graph of the multi-platoon network can be specified according to the following Laplacian matrix [**sharifi_platoons_2023**]

$$L_{\text{platoon}} = \begin{bmatrix} L_{ff} & L_{fl} \\ L_{lf} & L_{ll} \end{bmatrix} \tag{2.27}$$

where L_{ff} corresponds to the Laplacian matrix of followers interconnections, L_{fl} and L_{lf} model the communications from the leaders to followers and vice-versa, respectively, and L_{ll} demonstrates the communications among the leaders of platoons. We consider directed communication from the leaders to their followers. Hence $L_{fl} \neq L_{lf}^{\top}$ and $L_{lf} = \mathbf{0}$. In addition, the communication among the leaders of platoons is assumed to be undirected. In this manner, while each platoon is subject to its local tasks, there is no coordination between platoons, i.e., $L_{ll} = \mathbf{0}$. When platoon coordination actions such as merging or splitting are considered, the Laplacian matrix will change according to the new graph topology.

Let $\boldsymbol{x}_i \in \mathbb{R}^{n_{\text{states}}}$ and $\boldsymbol{u}_i \in \mathbb{R}^{m_{\text{inputs}}}$ denote the state and input of vehicle $i \in \mathcal{V}$, respectively. Moreover, $\mathcal{N}_i$ denotes the set of neighbours of vehicle i and $|\mathcal{N}_i|$ determines the cardinality of the set $\mathcal{N}_i$.

In addiction $\boldsymbol{f}_i : \mathbb{R}^{n_{\text{states}} + n_{\text{states}} \cdot |\mathcal{N}_i|} \to \mathbb{R}$ is assumed to be locally Lipschitz continuous functions.

We define the stacked vector of all elements in the set $\mathcal{X}$ with cardinality $|\mathcal{X}|$, as

$$[\boldsymbol{x}_i]_{i \in \mathcal{X}} := \left[\boldsymbol{x}_{i_1}^{\top}, \cdots, \boldsymbol{x}_{i_{|\mathcal{X}|}}^{\top} \right]^{\top}, i_1, \cdots, i_{|\mathcal{X}|} \in \mathcal{X} \tag{2.28}$$

and write the stacked dynamics for the network of platoons containing the vehicle dynamics $i \in \mathcal{V}$, as

$$\dot{\boldsymbol{z}}_i = \boldsymbol{f}_{pl}(\boldsymbol{z}, \boldsymbol{w}) \tag{2.29}$$

where $\boldsymbol{z} := [\boldsymbol{x}_i]_{i \in \mathcal{V}} \subseteq \mathbb{R}^{n_{\text{states}} \cdot n}$ and $\boldsymbol{w} := [\boldsymbol{u}_i]_{i \in \{n_f + 1, \cdots, n\}} \in \mathbb{R}^{m_{\text{inputs}} \cdot n_l}$ and $\boldsymbol{f}_{pl}(\cdot) = [\boldsymbol{f}_i(\cdot)]_{i \in \mathcal{V}} \in \mathbb{R}^{n_{\text{states}} \cdot n}$. the local dynamic functions $\boldsymbol{f}_{i,i}(x_i)$ correspond to the terms of $\boldsymbol{f}_i(x)$ which are only dependent on $\boldsymbol{x}_i$, and $\boldsymbol{f}_{i,j}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ contains the terms of $\boldsymbol{f}_i(\boldsymbol{x})$ which are dependent on agent $j \in \mathcal{V}, j \neq i$ as well. For

the case of one platoon (one leader), with follower and leader sets $\mathcal{V}_f := \{1, \cdots, n-1\}$ and $\mathcal{V}_l := \{n\}$, respectively. For networks containing more than one platoon (multiple leaders), with $\mathcal{V}_f := \{1, \cdots, n_f\}$ and $\mathcal{V}_l := \{n_f + 1, \cdots, n\}, n = n_f + n_l$.

Chapter 3

State-of-the-Art

In the rapidly evolving landscape of autonomous driving and platooning, the state-of-the-art chapter navigates through the advancements, methodologies, and breakthroughs that have shaped the current understanding and capabilities within the field. These not only contribute to the current state of research but also serve as essential benchmarks against which our own results and innovations can be effectively evaluated and compared.

3.1 Safety Guarantees

This section explores the critical aspect of safety in advanced control systems. Specifically, we focus on CBF as a contemporary approach to address obstacle avoidance challenges. We delve into the evolving field of safety-critical control, where theoretical advancements are applied to enhance the reliability of autonomous systems. In this context, we draw insights from the research conducted by **thirugnanam_safety-critical_2022** in [**thirugnanam_safety-critical_2022**].

As previously defined in subsection 2.2.3, a CBF serves as a function employed to enforce safety constraints on the states of a dynamical system. Once again, we refer to $\mathcal{C}$ as the safe set, representing the region outside the obstacle. The function h is termed a discrete-time CBF if $\forall \boldsymbol{x} \in \mathcal{C}, \exists \boldsymbol{u} \in \mathcal{U}$, as expressed in (2.24).

In the subsequent formulation, it becomes evident that if $\gamma(x)$ is close to 1, the system converges to $\partial\mathcal{C}$ slowly but can easily become infeasible. Conversely, if $\gamma(x)$ is close to 0, the constraint is feasible in a larger domain but can approach $\partial\mathcal{C}$ quickly, potentially leading to unsafe conditions.

The novel approach introduces a relaxing variable, denoted as $\boldsymbol{\gamma}$. Their proposed formulation rewrites the CBF constraint in (2.24) as follows

$$h(\boldsymbol{x}_{k+1}) \geq \boldsymbol{\omega}_k \boldsymbol{\gamma}_k h(\boldsymbol{x}_k), \quad 0 \leq \gamma(\boldsymbol{x}) < 1 \,, \tag{3.1}$$

where the relaxing variable ω resolves the trade-off between feasibility and safety and is optimised with other variables inside an optimisation formulation.

The NMPC problem can be posed as,

$$\begin{aligned}
\min_{\boldsymbol{u},\boldsymbol{\omega}} \quad & \sum_{k=0}^{N-1} \left[\boldsymbol{q}\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) + \boldsymbol{\psi}\left(\boldsymbol{\omega}_k\right)\right] \\
\text{s.t.} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k), \\
& \boldsymbol{u}_k \in \mathcal{U}, \qquad\qquad\qquad k = 0, \ldots, N-1, \\
& \boldsymbol{x}_k \in \mathcal{X}, \qquad\qquad\qquad k = 0, \ldots, N-1, \\
& h(\boldsymbol{x}_{k+1}) \geq \boldsymbol{\omega}_k \boldsymbol{\gamma}_k h(\boldsymbol{x}_k) \quad \boldsymbol{\omega}_k \geq 0 \quad k = 0, \ldots, N-1\,.
\end{aligned} \tag{3.2}$$

The proposed construction of CBFs involves formulating a non-convex optimisation problem to ensure safety guarantees and constraint satisfaction. However, non-convexity introduces the challenge of multiple local minima, making the efficient discovery of the global solution arduous.

3.2 Cooperative Communication

Within the scope of cooperative network communication, a critical consideration arises concerning choosing an appropriate topology. While inter-vehicle communication has long been explored and utilised in various vehicular applications, the concept of a leader-follower communication topology has emerged as a preferred and promising approach in the context of platooning [**boulu-reshef_impact_2020**]. The rationale behind this preference lies in the inherently less computationally demanding and simpler hierarchical structure it offers, while still achieving consensus within the network. When relying solely on inter-vehicle communication this introduces significant complexities associated with establishing and maintaining direct communication links among all vehicles within a platoon. This requirement leads to a continuous exchange of information between multiple nodes, resulting in heightened network traffic and the potential for latency issues. Additionally, within the latter topology, the spanning communication tree rooted in the leader proves advantageous [**wang_leaderfollower_2018**].

As of now, low-level vehicle automation systems are already in use on the road, including technologies like ACC, Lane Change Assistant (LCA) and Lane Keeping Assistant (LKA). CACC emerges as a prominent extension of the fundamental principles of ACC. In this segment, we will delve into the state-of-the-art of CACC, beginning with a short explanation of ACC, its commencement algorithm.

3.2.1 The ACC Model

The ACC system automatically controls the speed of the ego vehicle to ensure a suitable distance with an obstacle or vehicle ahead, based on the relative distance and speed detected by the onboard sensors. The system comprises two key controllers: the upper and bottom controllers. The upper controller generates the desired acceleration control signal based on the current driving profile, while the lower controller translates this desired acceleration signal into appropriate brake or throttle control actions according to the ego vehicle's current acceleration [**zhao_full-range_2014**].

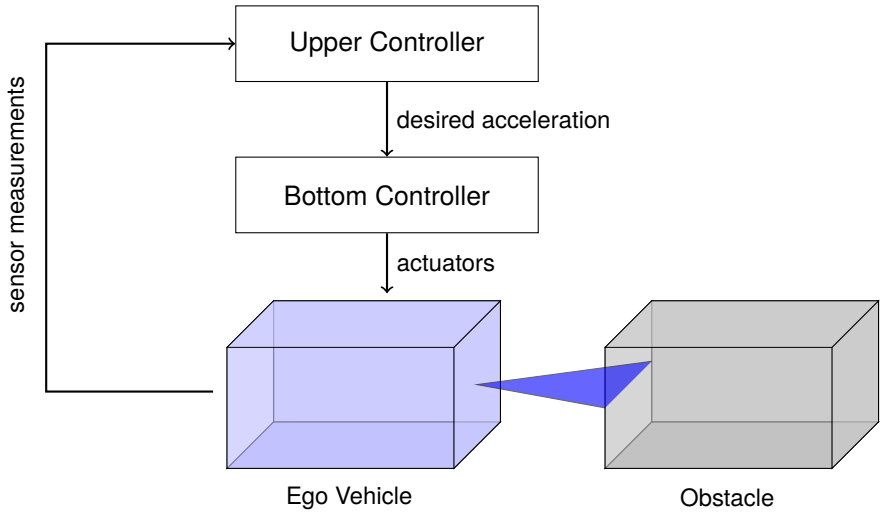


Figure 3.1: Illustration of the ACC framework. The upper controller generates the desired acceleration signal, while the bottom controller maps the acceleration to the brake or the throttle control signals. The LiDAR sensor detects the distance between the two vehicles and predicts the obstacle speed. Inspired and adapted from **zhao_full-range_2014** (**zhao_full-range_2014**).

Denote as d_k^{real} the distance at step k between the ego and the target vehicles or obstacle. Such a distance is used to compute the instant speed of the target vehicle v_k^{obstacle} (refer to Figure 3.1); the desired distance d_k^{desired} between these vehicles is always set by the driver while the ego vehicle speed v_k^{ego} can be read from the speed encoder. The control goal is to keep the ego vehicle within a safe distance and maintain the safe relative speed $\Delta v_k = v_k^{\text{ego}} - v_k^{\text{obstacle}}$. Similarly, the relative distance Δd_k at step k is $\Delta d_k = d_k^{\text{real}} - d_k^{\text{desired}}$. The upper controller goal is to simultaneously drive variables $(\Delta v_k, \Delta d_k)$ to zero by enforcing the most appropriate acceleration control action. The bottom controller manages both the throttle and the brake.

This technology, indeed, improves traffic safety and comfort significantly. Nevertheless, a sizeable inter-vehicle distance is required for safety in the case of an emergency scenario, which influences traffic flow efficiency negatively. This can be remedied by obtaining more detailed information about the vehicle ahead such as acceleration and control inputs and feeding them to the ego vehicle. This extension of ACC, called CACC, uses Vehicle-to-Vehicle Communication (V2V) communication to convey the acceleration or control input information of the vehicle ahead. CACC could result in closer spacing of the vehicles in a platoon while preserving stable operation.

3.2.2 The CACC Model

Cooperative Adaptive Cruise Control (CACC) represents a pioneering advancement in vehicular control systems, building upon the foundations laid by ACC. With its capability to facilitate remarkably short time gaps between vehicles within a platoon, CACC holds the potential to revolutionise road capacity and alleviate traffic congestion significantly. This remarkable achievement is attributed to the integration of sensors and communication technologies, which enable not only adaptive cruise control but also cooperative elements essential for safe platoon manoeuvres.

3.2.2.1 Controller Structure

The core of the CACC system is rooted in its control structure, illustrated in Figure 3.2. The platoon comprises a leader vehicle and n_f follower vehicles, and communication between adjacent vehicles is facilitated through V2V techniques. When an obstacle appears in front of the ego vehicle within the same lane, maintaining an appropriate distance and a higher speed than the platoon, the CACC system orchestrates the speed reduction of all platoon members and assigns new cruising speeds to accommodate the ego vehicle. This control strategy, outlined in more detail in [**ma_cooperative_2020**], ensures seamless platoon integration while epitomising traffic flow.

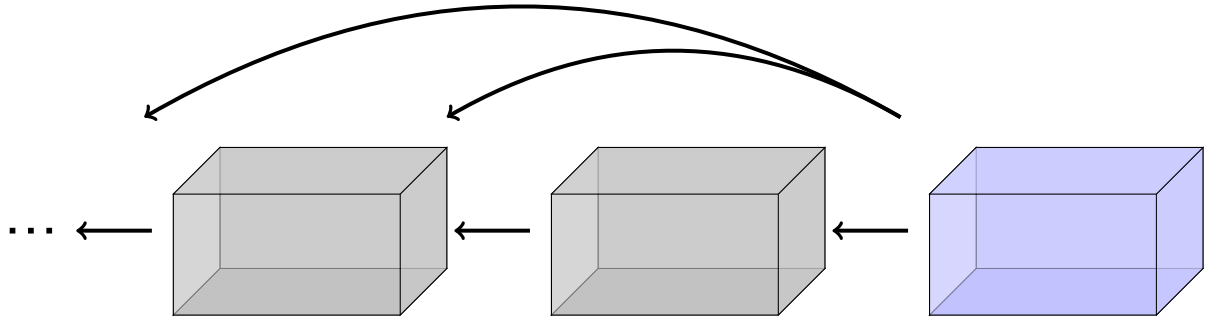


Figure 3.2: Illustration of the CACC Leader-Follower Topology. This schematic depicts the structure of a CACC-enabled vehicle platoon, including the leader vehicle (represented in *blue*) and multiple follower vehicles (represented in *light grey*), demonstrating the communication and coordination essential for safe and efficient platoon operation.

3.2.2.2 Gap Management

The primary goal of the CACC controller is to maintain the driver-desired time gap with the preceding vehicle under various traffic conditions, prioritising both smoothness and precision. The driver interface, inherited from ACC, plays a pivotal role in managing the CACC controller. It offers options to activate and deactivate the CACC controller and adjust cruise control speed when no obstacle is detected in front of the ego vehicle.

For the CACC system, the time gap settings are more diverse than those in the ACC factory system. The shortest gap can be determined based on safety considerations and driver acceptance tests. It's important to note that the CACC controller operates within certain acceleration and deceleration limits set by the low-level controller.

3.2.2.3 Controller Stages

The CACC controller operates in two distinct stages. The first stage is activated when the CACC system is engaged, and there is no vehicle in front of the ego vehicle or when the ego vehicle is significantly distant from the preceding one. In such scenarios, the vehicle typically adheres to its preset speed, necessitating a smooth transition to the second stage. This initial controller, known as the *gap-closing controller* focuses on executing approach manoeuvres smoothly, ensuring a seamless transition to the gap regulation controller. Additionally, it plays a pivotal role in handling cut-out manoeuvres when a vehicle within the platoon decides to exit, requiring the following vehicle to close the gap.

Once the ego vehicle has successfully joined its predecessor, the second-stage controller, referred to as the *CACC gap regulation controller* takes over. This controller is responsible for implementing car-following policies based on the driver-selected time gap. Different time gaps are available, mirroring the production ACC structure. Furthermore, it manages cut-in makeovers when non-equipped vehicles merge into the platoon. Both controllers should be meticulously designed to emulate human driver behaviour in various driving situations, ensuring a harmonious and safe driving experience within CACC-enabled platoons [**milanes_cooperative_2014**].

Chapter 4

Method

In this chapter, a comprehensive explanation of the proposed algorithm will be carried out along with a detailed description of its implementation. Foremost, the problem statement is introduced.

4.1 Problem Statement

In the context of autonomous driving, and as noted earlier, platooning refers to a group of vehicles that move in close proximity, coordinated by an ego vehicle. The single ego vehicle serves as the leader of the formation while the followers align closely behind it.

A highway-like environment is to be considered, defined by a set of L constitutional lanes denoted as

$$\mathcal{E} = \{l_0, l_1, \dots, l_{L-1}\} \ , \tag{4.1}$$

assuming a left-hand driving configuration. The primary constraints in this environment are the driving direction and lane-specific speed limits, which are applicable to all vehicles within the platoon.

All vehicles considered should be modelled as trucks using the general n-trailer model, as described in (2.6). The dynamics of the ego agent are then given by

$$\dot{\boldsymbol{x}}_{\text{ego}} = \boldsymbol{f}_{\text{ego}}(\boldsymbol{x}_{\text{ego}}, \boldsymbol{u}_{\text{ego}}) \ , \tag{4.2}$$

while the dynamics of the follower by

$$\dot{\boldsymbol{x}}_{\text{follower}} = \boldsymbol{f}_{\text{follower}}(\boldsymbol{x}_{\text{follower}}, \boldsymbol{u}_{\text{follower}}) \ . \tag{4.3}$$

These equations, both nonlinear, should be further discretised.

It is important to emphasise that, in this setup, only the ego vehicle is equipped with a sensor, specifically a LiDAR. To achieve a simulation that closely mirrors reality, we must not assume that the entire environment is constantly known. Instead, we should account for the limited range of perception for the sensor. This necessitates the introduction of the following concept:

Definition 3 (Field of View (FoV))**.** *FoV$_k \subset \mathbb{R}^2$ is defined as the set of points at time step k which are within direct line of sight from the sensor resolution, restricted to the ego vehicle's current lane l_i. The index i auxiliaries the lane designation present in the road segment considered, ranging from the leftmost to the rightmost lane.*

Furthermore, providing a precise definition of the FoV is crucial for a comprehensive understanding of the sensor's range, resolution, and limitations.

This research is centred on the development of a NMPC strategy integrated within a CACC framework capable of effectively controlling all the agents in the network, and the platoon while ensuring a safe and feasible trajectory at all time steps. An emphasis also lies on real-time computation and decision-making capabilities, with a specific aim to render the strategy computationally manageable when compared to state-of-the-art solutions.

4.2 Proposed Algorithm

This section on the proposed algorithm is structured to provide a comprehensive understanding of the platooning system's inner workings. First, we delve into the control mechanisms of the ego vehicle in subsection 4.2.1. This segment elucidates how lane-changing and obstacle avoidance manoeuvres are handled. Next, the follower vehicles controller is discussed in subsection 4.2.2. Here, the intricacies of how these agents adapt their behaviour, maintaining the desired distance and speed relative to the ego vehicle and the platoon are unravelled. And finally, in subsection 4.2.3, the entire platooning system—CACC— is detailed. This component ties together the actions of the ego and follower vehicles, orchestrating the platoon dynamics through advanced control algorithms. It ensures that the platoon functions as a cohesive unit, optimising performance, and enhancing safety. The proposed formulation serves as a local planner to generate dynamically-feasible and collision-free trajectories, through the predictive capabilities of NMPC.

4.2.1 Ego Vehicle

This segment introduces the control scheme for the reference tracking problem, an essential component for the lane-changing manoeuvre while setting the stage for the obstacle avoidance control algorithm as part of the overtake scheme, all centred around the ego vehicle.

The continuous model dynamics presented in section 2.1 is assumed to be applicable to the ego vehicle dynamics. However, to enable the transition from an Optimal Control Problem (OCP) to a Nonlinear Programming (NLP) formulation, a discrete model is required, thus, the continuous-time dynamics are discretised through the explicit Runge-Kutta 4th Order (RK4) method, according to Appendix A, yielding:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) \, . \tag{4.4}$$

Note that here the ego subscripts were omitted as to simplify notation. Hence, $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n$, denoting the ego state, and $\boldsymbol{u} \in \mathcal{U} \subset \mathbb{R}^m$, $\mathcal{U}$ is a compact set and $\boldsymbol{f}$ encapsulates the system's behaviour. This method approximates the continuous dynamics of the system and enables the derivation of a discrete

representation. As to account for the inherent limitations of the system described in section 2.1 and the constraints on the input signals, the aforementioned sets $\mathcal{X}$ and $\mathcal{U}$ are naturally bounded. In this context, we will focus on specifying the bounded state and input variables without explicitly mentioning the sets $\mathcal{X}$ and $\mathcal{U}$. Therefore the control strategy results in a NMPC reference tracking problem, as formulated below:

$$
\begin{aligned}
\min_{\boldsymbol{u}} \quad & \sum_{k=0}^{N-1} \Big((\boldsymbol{x}_k - \boldsymbol{r}_k)^\top \mathbf{Q} (\boldsymbol{x}_k - \boldsymbol{r}_k) + (\boldsymbol{u}_k - \boldsymbol{u}_k^{\text{ref}})^\top \mathbf{R} (\boldsymbol{u}_k - \boldsymbol{u}_k^{\text{ref}}) \Big) \\
\text{s.t.} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k), \\
& \beta_i \in [-\beta_{i,\max}, \beta_{i,\max}], \qquad i = 1, \dots, n, \\
& \delta_0 \in [-\delta_{\max}, \delta_{\max}], \\
& v_0 \in [v_{\min}, v_{\max}], \\
& u_0 \in [a_{\min}, a_{\max}], \\
& u_1 \in [-\dot{\delta}_{\max}, \dot{\delta}_{\max}].
\end{aligned}
\tag{4.5}
$$

In this formulation, $\boldsymbol{u}$ represents the control inputs, $\boldsymbol{x}$ denotes the state variables, and $\boldsymbol{r}$ corresponds to the reference trajectory. This reference trajectory is designed to be feasible within the system's limits and constraints, ensuring that it adheres to the physical and operational restrictions of the system. The objective function consists of a cost term penalising deviations from the reference trajectory given the weight matrix $\mathbf{Q}$ and a control effort term $\mathbf{R}$ penalising deviations from the desired control input $\boldsymbol{u}^{\text{ref}}$. These weight matrices should be diagonal, positive definite and unbalanced, i.e. unequal magnitudes of the diagonal elements: the weight matrices should be positive definite to ensure convexity and the unbalanced nature allows for individual control effort tuning and emphasising certain state variables or control inputs over others in terms of their impact on the control objective [**malisoff_tracking_2020**]. By assigning larger weights to certain elements, the control system can prioritise specific performance criteria or desired behaviour.

The constraints impose limits on the system's variables, such as the hitching angle (β_i), initial steering angle (δ_0), initial velocity (v_0), longitudinal acceleration (u_0), and the steering rate (u_1). The index i denotes i-th trailer in the general n-trailer representation presented in subsection 2.1.3. For the sake of simplicity, terminal costs are omitted from the objective function, but they can be included if necessary. Terminal costs are used to prioritise achieving specific behaviour or performance at the end of the horizon, while terminal constraints ensure requirements are met precisely at the final time step. Indeed, the decision to include them depends on the control problem and desired system behaviour.

4.2.1.1 Environment Interpretation

Recall that the highway environment is described by the expression in (4.1), thus its associated constraints, here denoted as $\mathcal{C}$, can be formally defined as follows

$$
\mathcal{C} = \{ v_0 \leq v_{\text{limit}} \mid l_i \in \mathcal{E}, \text{constraint on lane } l_i \} \ . \tag{4.6}
$$

In a real-life scenario, various lanes on the same segments of road exhibit distinct velocity constraints, and this is what $\mathcal{C}$ emulates.

Note that v_0 represents the longitudinal velocity of each vehicle. That is when assuming a highway-like environment, the constraints are the driving direction, $v \geq 0$, and the speed limit in the lane, v_{limit}.

Thus, the basic NMPC controller for the ego vehicle can be here reformulated as

$$\begin{aligned} \min_{\boldsymbol{u}(\cdot)} \quad & \sum_{k=0}^{N-1} \boldsymbol{q}\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) \\ \text{s.t.} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k), \\ & \boldsymbol{u}_k \in \mathcal{U}, \qquad k = 0, \ldots, N-1, \\ & \boldsymbol{x}_k \in \mathcal{X} \cap \mathcal{C}, \quad k = 0, \ldots, N-1, \end{aligned} \tag{4.7}$$

where $\boldsymbol{q}$ depicts the reference tracking penalty. i.e the stage cost such that

$$\boldsymbol{q}\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) = \left(\boldsymbol{x}_k - \boldsymbol{r}_k\right)^\top \mathbf{Q} (\boldsymbol{x}_k - \boldsymbol{r}_k) + \left(\boldsymbol{u}_k - \boldsymbol{u}_k^{\text{ref}}\right)^\top \mathbf{R} (\boldsymbol{u}_k - \boldsymbol{u}_k^{\text{ref}}) \, . \tag{4.8}$$

To further refine the definition of the FoV we must take into account the onboard used, a LiDAR. Therefore, the FoV can be redefined as a subset of points within a certain range and angular span [**yin_spherical_2016**] [**yuan_pixel-level_2021**]. Let the sensor's position be denoted by $(x_{\text{LiDAR}}, y_{\text{LiDAR}})$ in the 2D plane, and let θ_{min} and θ_{max} represent the minimum and maximum angles of the FoV relative to the LiDAR's orientation. Thus, the FoV at a time instant k can be mathematically defined as the set of points (x, y) such that:

$$\text{FoV}_k = \{(x, y) \in \mathbb{R}^2 \mid \sqrt{(x - x_{\text{LiDAR}})^2 + (y - y_{\text{LiDAR}})^2} \leq R_{\text{max}}, \theta_{\text{min}} \leq \theta \leq \theta_{\text{max}}\}. \tag{4.9}$$

That is the point (x, y) is within a certain distance or range R_{max} of the LiDAR sensor and the angle θ formed by the vector from the LiDAR sensor to point (x, y) falls within the FoV boundaries.

Additionally, it is assumed that the sensor is perfectly coupled and aligned within the ego vehicle such that $(x_{\text{LiDAR}}, y_{\text{LiDAR}})$ coincides exactly with (x_0, y_0) - recall the system's model in subsection 2.1.3. Note that the FoV does not necessarily coincide with the predefined horizon of the NMPC. Meaning, that information outside the NMPC's horizon may not be used immediately for control decisions, even if the sensor can perceive it, which can affect system performance and safety.

4.2.1.2 Obstacle Detection and Geometric Modelling

Obstacles often exhibit intricate shapes, making it necessary to represent their geometry concisely. To address this, a bounded ellipsoid is employed as the geometric model for all obstacles. The convex nature of the ellipsoid provides an ideal framework for describing obstacles with varying dimensions and orientations, enabling efficient and reliable trajectory planning [**villasenor_ellipsoidal_2021**]. The geometry of the ellipsoid is determined by reference points obtained from sensors. In other words, the obstacle can be represented by an ellipse that encompasses all the reference points acquired through the sensors. This representation is achieved using Khachiyan's algorithm for Minimum-Volume Enclosing Ellipsoid (MVEE), as discussed in Appendix B. We remark that there exists a faster algorithm for computing the ellipsoid but without guarantees of minimum volume as in [**silvestre_model_2023**].

The proposed approach addresses the prediction of overtaking or lane-changing manoeuvres within the available left lane, following the driving conventions and regulations specific to the road network of the region under consideration. Thus, the leftmost boundary of obstacles encapsulated within the ellipsoid acts as a reference point for the manoeuvre represented as $\boldsymbol{b} \subset \mathbb{R}^2$. This is particularly valuable in the context of predicting a leftward overtaking strategy during obstacle avoidance.

The determination of the leftmost boundary point, denoted as $\boldsymbol{b}$, is made concerning a reference frame aligned and centred on the orientation of the ego vehicle, as illustrated in Figure 4.1.

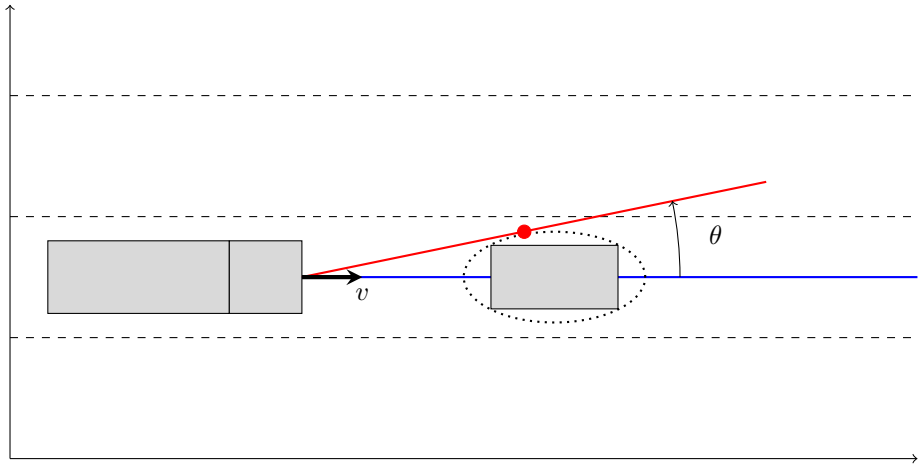


Figure 4.1: A schematic description depicting the modelling of an obstacle and the resultant waypoint generation in a simulated highway environment. On the left, a rectangle represents the ego vehicle, while the rectangle on the right symbolises the obstacle ahead, while an ellipsoid bounding it is displayed above it. The blue line illustrates the initial reference trajectory of the ego vehicle. The red line traces the path used to determine the obstacle's leftmost boundary, where θ represents the line's inclination angle. The red dot marks the algorithm's retrieval of the leftmost boundary.

Therefore, let $\mathcal{D}$ be an ellipse obtained from Khachiyan's algorithm with centre $\boldsymbol{c}$ and shape matrix $\mathbf{A}$. To find the leftmost boundary point of $\mathcal{D}$, we determine the line $\mathcal{L}$ denoted by its slope angle θ, so that,

$$y = \tan(\theta)x. \tag{4.10}$$

Let $\boldsymbol{p}$ be a possible set of intersection points of line $\mathcal{L}$ and the ellipse $\mathcal{D}$, in the format $\boldsymbol{p} = [x, y]^\top$, accordingly the equation

$$(\boldsymbol{p} - \boldsymbol{c})^\top \mathbf{A}^{-1} (\boldsymbol{p} - \boldsymbol{c}) = 1 \tag{4.11}$$

should hold. If there exists a single intersection point $\boldsymbol{p}$ lying on $\mathcal{L}$ and coincides with the leftmost boundary of $\mathcal{D}$.

This formulation describes the iterative process of tracing lines at various angles and checking for the intersection points between those lines and the ellipse. The leftmost boundary point is updated when a line intersects the ellipse at only one point, indicating that the line coincides with the leftmost boundary. This process is summarised in the Algorithm 1.

Algorithm 1 Retrieve Leftmost Boundary Point of an Ellipse

Initialise $\boldsymbol{b}_0 \leftarrow [-\infty, 0]$
Initialise angle range $[\theta_{\text{min}}, \theta_{\text{max}}] \leftarrow [0, \frac{\pi}{2}]$
while $\theta_{\text{max}} - \theta_{\text{min}} > \delta$ **do**
    Compute midpoint angle $\theta_m \leftarrow \dfrac{\theta_{\text{min}} + \theta_{\text{max}}}{2}$
    Construct line $\mathcal{L}$ with equation $y = \tan(\theta_m)x$
    Compute intersection points between $\mathcal{L}$ and ellipse $\mathcal{D}$
    if there is exactly one boundary intersection point $\boldsymbol{p}$ **then**
        Update $\boldsymbol{b}_i \leftarrow \boldsymbol{p}$
        break
    else
        if $\boldsymbol{p}$ has 2 solutions **then** $\theta_{\text{min}} = \theta_m$
        end if
        if $\boldsymbol{p}$ has 0 solutions **then** $\theta_{\text{max}} = \theta_m$
        end if
        Update angle range $[\theta_{\text{min}}, \theta_{\text{max}}]$
    end if
end while
Output final leftmost boundary point $\boldsymbol{b} \leftarrow \boldsymbol{b}_{\text{final}}$

δ is defined here as a small empirical threshold; theoretically, it should be zero.

Here, follows an additional reasoning on the proposed algorithm.

Proposition 1. *If there exists a single intersection point $\boldsymbol{p}$ lying on $\mathcal{L}$ then $\boldsymbol{p}$ coincides with the leftmost boundary of $\mathcal{D}$.*

Proof. Let us consider the possible cases for the first iteration of the Algorithm 1 where θ_m is set to $\frac{\pi}{2}$:

1. If the ellipse $\mathcal{D}$ is intersected by the line $\mathcal{L}$ such that it produces two solutions, this implies that the leftmost boundary lies above the line. Thus in the next iteration, θ_m will increase, bringing the line closer to the point of interest. This indicates that the algorithm is converging towards the leftmost boundary.

2. If the ellipse $\mathcal{D}$ is not intersected by the line $\mathcal{L}$, it means that the ellipse is situated below the line. In the next iteration, θ_m will decrease, moving the line closer to a vertical orientation. This suggests that the algorithm is progressing towards finding the leftmost boundary.

3. In this case, when the ellipse $\mathcal{D}$ intersects the line at exactly one point, it is a strong indicator that this point is the leftmost boundary. This is supported by the fact that the LiDAR range selected is below 90 degrees, which ensures that the intersection point is indeed the leftmost boundary.

In each of these cases, the algorithm is either directly aligned with the leftmost boundary point or actively moving towards it. Therefore, the presence of a single intersection point in the first iteration indicates that it coincides with the leftmost boundary. □

4.2.1.3 Obstacle Estimation

In order to anticipate the future positions of obstacles, the movement of each one is modelled as a single point in space moving linearly in discrete time steps. Their movement was indexed by k in the context of each time step of the NMPC horizon. The predicted state can be obtained using the following formulation:

$$\mathcal{P}_k = \left\{(x_k, y_k) \;\middle|\; x_k = x_0 + v_x \cdot h \cdot k,\; y_k = y_0 + v_y \cdot h \cdot k\right\} \;. \tag{4.12}$$

Here, (x_k, y_k) represents the obstacle's position at time step k, h is the predefined sampling time, and (x_0, y_0) originates from the leftmost boundary $\boldsymbol{b}$ detected through Algorithm 1. While it's true that objects can have complex trajectories, assuming linear motion within a short prediction horizon is a reasonable simplification in many cases since often vehicles can exhibit linear or nearly linear motion over short time intervals.

To estimate the velocity of the obstacle, we employ a Kalman Filter (KF) [**simon_kalman_2002**], a powerful tool for accurate state estimation using noisy sensor measurements. Traditional methods, such as differentiation of positional data, may introduce noise and errors that accumulate over time, affecting velocity estimates. In contrast, the filter mitigates these issues by effectively filtering out noise from sensor measurements.

The KF employs a constant acceleration motion model to estimate the obstacle's future positions. This model considers the x-coordinate, y-coordinate, x-velocity, y-velocity, x-acceleration, and y-acceleration of the obstacle over a time interval h. The state vector $\boldsymbol{o}$ encapsulates these variables. To enhance the accuracy of predicting obstacle positions, we integrate the filter into the prediction process [**fossen_extended_2018**]. This technique estimates the future position of the detected obstacle based on its current state. The integration is mathematically formulated in these steps:

1. State Prediction - the predicted state estimate $\hat{\boldsymbol{o}}_k^-$ and predicted error covariance $\mathbf{P}_k^-$ are computed using the following equations:

$$\begin{aligned}\hat{\boldsymbol{o}}_k^- &= \mathbf{A}\hat{\boldsymbol{o}}_{k-1} + \mathbf{B}\boldsymbol{u}_{k-1} \\ \mathbf{P}_k^- &= \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top + \mathbf{Q}_{\text{KF}}\end{aligned}$$

2. Measurement Prediction - the predicted measurement $\hat{\boldsymbol{z}}_k$ and measurement error covariance $\mathbf{S}_k$ are calculated as follows:

$$\begin{aligned}\hat{\boldsymbol{z}}_k &= \mathbf{H}_{\text{KF}}\hat{\boldsymbol{o}}_k^- \\ \mathbf{S}_k &= \mathbf{H}_{\text{KF}}\mathbf{P}_k^-\mathbf{H}_{\text{KF}}^\top + \mathbf{R}_{\text{KF}}\end{aligned}$$

3. Kalman Gain Computation - the Kalman gain $\mathbf{K}_k$ is computed using the predicted error covariance and measurement error covariance:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_{\text{KF}}^\top\mathbf{S}_k^{-1}$$

4. State Update - finally, the updated state estimate $\hat{\boldsymbol{o}}_k$ and updated error covariance $\mathbf{P}_k$ are obtained using the following equations:

$$\hat{\boldsymbol{o}}_k = \hat{\boldsymbol{o}}_k^- + \mathbf{K}_k(\boldsymbol{z}_k - \hat{\boldsymbol{z}}_k)$$
$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_{\mathsf{KF}})\mathbf{P}_k^-$$

In the above equations, $\hat{\boldsymbol{o}}_k^-$ represents the predicted state estimate, $\mathbf{P}_k^-$ is the predicted error covariance, $\mathbf{Q}$ represents the process noise covariance, and $\mathbf{R}$ is the measurement noise covariance.

4.2.1.4 Integration with Motion Planning

Here we propose a dynamic obstacle avoidance penalty method based on a NMPC problem, drawing inspiration from Artificial Potential Field (APF)s [**sheng_obstacle_2022**]. The integration involves incorporating an additional optimisation problem that is activated only when an obstacle is detected within the FoV of the ego vehicle, i.e.

$$\mathsf{FoV}_k \cap \mathcal{O} \neq \varnothing \,, \tag{4.13}$$

where $\mathcal{O}$ represents the set of detected obstacles. Therefore, the NMPC framework is combined with the previously proposed optimisation problem in Equation (4.5), allowing for adaptive obstacle avoidance during motion planning. This is summarised below in Algorithm 2.

Algorithm 2 Dynamic Obstacle Avoidance with NMPC

Input: Environment $\mathcal{E}$, current time step k, ego vehicle state $\boldsymbol{x}_k^{\mathsf{ego}}$, FoV_k, detected set of obstacle $\mathcal{O}$
Output: Optimised trajectory for motion planning

```
Compile NMPC reference tracking problem
if FoV_k ∩ O ≠ ∅ then
    Incorporate obstacle avoidance penalty terms into the NMPC objective
end if
Optimise the NMPC problem to obtain the trajectory for motion planning
Return Optimised trajectory for motion planning
```

This permits continuity in the objective function when switching from the different optimisation problems at hand, allowing for a smoother and more robust control input.

Figure 4.2 depicts an APF where obstacles are positioned at coordinates $(0, 0)$ and $(5, 5)$. The penalisation cost exponentially increases with the distance to the obstacles, effectively influencing the trajectory planning process.

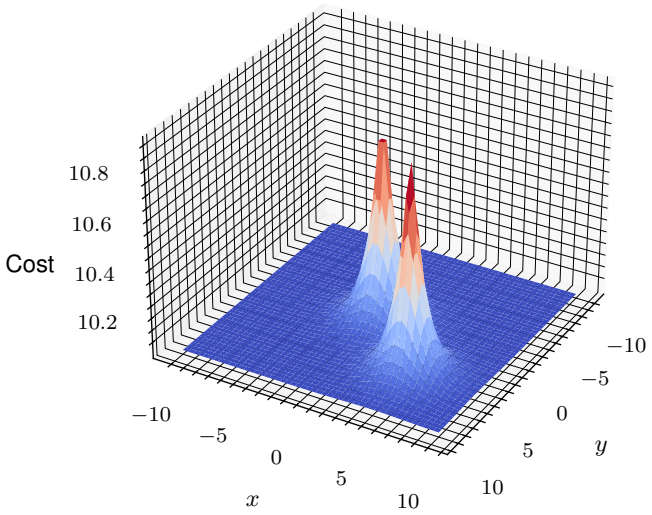


Figure 4.2: Illustration of an Artificial Potential Field, featuring dimensionless and static obstacles located at coordinates $(0, 0)$ and $(5, 5)$, respectively. For demonstrative purposes, the penalty is represented by an exponential cost function.

Accordingly, in order to enforce the obstacle constraints to an acceptable predefined tolerance, we employ a penalty method, as shown in Algorithm 3.

Algorithm 3 Dynamic Obstacle Penalty

Input: Current time step k, ego vehicle state $\boldsymbol{x}_k^{\text{ego}}$, FoV$_k$, detected obstacle $\mathcal{O}_i$
Output: Cost function with an adequate penalty

```
initiate cost function J = 0
for i = 0 to N do
    Predict obstacle position P_k                ▷ According to Equation (4.12)
    Compute its leftmost boundary b_i            ▷ According to Algorithm 1
    if available left lane for overtake in E then
        Define inner product ⟨(x_0 − x_k), (b_i − x_k)⟩
        Sum the exponential weight to cost function
    end if
end for
```

It is important to note that the determination of the condition "available left lane for overtaking in $\mathcal{E}$" is conducted indirectly. In this approach, the algorithm assesses the contour of an obstacle within an obstacle-free field of view ahead. Consequently, if such conditions are not met, the generated solution refrains from executing an overtaking manoeuvre. Instead, the ego vehicle maintains its position within the current lane, adjusting its cruising speed to ensure a safe distance from the obstacle ahead.

The proposed NMPC problem is thus formulated as follows,

$$\begin{aligned} \min_{\boldsymbol{u}(\cdot)} \quad & \sum_{k=0}^{N-1} \left[\boldsymbol{q}\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) + \boldsymbol{r}\left(\boldsymbol{x}_k, \mathcal{O}_i\right) \right] \\ \text{s.t.} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k), \\ & \boldsymbol{u}_k \in \mathcal{U}, \qquad k = 0, \ldots, N-1, \\ & \boldsymbol{x}_k \in \mathcal{X} \cap \mathcal{C}, \quad k = 0, \ldots, N-1, \end{aligned} \tag{4.14}$$

the different obstacles in the environment are indexed by i. Here, $\boldsymbol{q}$ depicts the aforementioned reference tracking penalty in (4.8). The term $\boldsymbol{r}$ conveys the exponential weight penalty

$$\boldsymbol{r}\left(\boldsymbol{x}_k, \mathcal{O}\right) = \beta^k \left\langle \left(\boldsymbol{x}_0 - \boldsymbol{x}_k\right), \left(\boldsymbol{b}_i - \boldsymbol{x}_k\right) \right\rangle \ . \tag{4.15}$$

This exponential penalty forces the trajectory to change locally where the obstacle is predicted to be, β is the base of the exponential function. This term operates as a soft constraint in the optimisation problem.

Here, it is made explicit that the objective function is parameterised in the penalty factors, allowing the ego vehicle to contour the obstacle nicely and also providing a feasible reference trajectory to the following vehicles.

Proposition 2. *The reachable set of the states of the ego is within the safe set*

$\mathcal{X} = \{\boldsymbol{x} \subset \mathbb{R}^2 | \operatorname{proj}_{xy}(\boldsymbol{x}) \cap \operatorname{proj}_{xy}(\mathcal{O}) \neq \varnothing\}$.

Proof. The distance constraint would be posed as:

$$\|\boldsymbol{b}_i - \boldsymbol{x}_k\| > 0 \ . \tag{4.16}$$

However, solving such a hard-constrained problem requires nonlinear constraints, leading to a complex non-convex MPC with multiple local minima.

To overcome this challenge, we introduce a linear soft constraint by penalising the dot product between the velocity vector and the obstacle vector. The dot product can be decomposed as

$$\left\langle \left(\boldsymbol{x}_0 - \boldsymbol{x}_k\right), \left(\boldsymbol{b}_i - \boldsymbol{x}_k\right) \right\rangle = \|\boldsymbol{x}_0 - \boldsymbol{x}_k\| \cdot \|\boldsymbol{b}_i - \boldsymbol{x}_k\| \cdot \cos\theta \ . \tag{4.17}$$

Thus, by minimising the variable θ linearly, the ego vehicle is effectively guided to contour the dynamic obstacle. □

This NMPC formulation can be posed as a non-convex optimisation problem with a linear soft constraint on safety control. Making it computationally tractable and amenable to state-of-the-art solvers.

4.2.2 Follower Vehicles

In this segment, we delve into the control strategy for the follower vehicles within the platoon. It is important to note that, in terms of their dynamic model, the state-of-the-art employed for follower vehicles remains consistent with that of the ego vehicle, as detailed in section 2.1. This common model serves as a foundation for reference tracking, building upon the principles outlined in subsection 4.2.1.

The primary objective for each individual follower vehicle is to execute reference tracking in line with the platoon's trajectory, resembling the formulation present initially in (4.5). However, an additional safety constraint comes into play: ensuring a safe distance is maintained from the preceding vehicle. To incorporate this safety consideration, the optimisation problem can be formulated as follows:

$$
\begin{aligned}
\min_{\boldsymbol{u}(\cdot)} \quad & \sum_{k=0}^{N-1} \boldsymbol{q}\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) \\
\text{s.t.} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k), \\
& \boldsymbol{u}_k \in \mathcal{U}, && k = 0, \ldots, N-1, \\
& \boldsymbol{x}_k \in \mathcal{X} \cap \mathcal{C}, && k = 0, \ldots, N-1, \\
& \quad (x_0 - x_{\text{preceder}})^2 + (y_0 - y_{\text{preceder}})^2 \geq d^2, && k = 0, \ldots, N-1.
\end{aligned}
\tag{4.18}
$$

The additional constraint is introduced to ensure that the Euclidean distance between the follower vehicle's position (x_0, y_0) and the position of the preceding vehicle $(x_{\text{preceder}}, y_{\text{preceder}})$ remains greater than or equal to the safe distance threshold d. This constraint plays a critical role in enforcing the desired separation between the follower vehicle and its preceding counterpart throughout the optimisation process. The cost function can be analysed in detail,

$$
\boldsymbol{q} = (\boldsymbol{x}_k - \boldsymbol{r}_k)^\top \mathbf{Q} (\boldsymbol{x}_k - \boldsymbol{r}_k) + (\boldsymbol{u}_k - \boldsymbol{u}_k^{\text{ref}})^\top \mathbf{R} (\boldsymbol{u}_k - \boldsymbol{u}_k^{\text{ref}}) \, . \tag{4.19}
$$

It is essential to highlight that the reference trajectory $\boldsymbol{r}$ for each follower vehicle should be derived from the ego vehicle's current position to maintain a cohesive and synchronised platoon trajectory. That is, the reference states are the ego's states at the different iterations of the horizon. Additionally, the control inputs $\boldsymbol{u}^{\text{ref}}$ are derived from the ego's optimised steering and throttle signals. This works as a a global planning for the follower vehicles.

This control strategy not only facilitates reference tracking for follower vehicles but also prioritises safety by actively maintaining appropriate separation distances, ensuring the smooth and secure operation of the platoon.

4.2.3 CACC

CACC constitutes the cornerstone of our platooning system, orchestrating the interaction between the ego and follower vehicles. Building upon the principles introduced in the background (section 2.3), CACC leverages advanced control algorithms and cooperative communication to optimise platoon performance, enhance safety, and unlock the full potential of autonomous platooning.

To understand the interactions within the platoon, we can represent the relationships between vehicles using a network representation. The adjacency matrix, denoted as $\mathbb{A}$, captures the connectivity between agents:

$$
\mathbb{A} = \begin{bmatrix}
0 & 1 & 1 & 1 & \cdots & 1 \\
0 & 0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 0 & 1 & \cdots & 0 \\
0 & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & \cdots & 0
\end{bmatrix} . \tag{4.20}
$$

Here, each row represents a vehicle in the platoon, and a "1" in a column indicates a connection between

vehicles.

The degree matrix, denoted as Δ, quantifies the number of connections each vehicle has, let the number of followers in the platoon be n_f, the degree matrix is

$$\Delta = \begin{bmatrix} n_f & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.21}$$

The Laplacian matrix, denoted L, encapsulates the interactions between vehicles within the platoon. It is defined as the difference between the degree matrix Δ and the adjacency matrix $\mathbb{A}$,

$$L = \begin{bmatrix} n_f & -1 & -1 & -1 & \cdots & -1 \\ 0 & 1 & -1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & -1 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \vdots & 0 \end{bmatrix}. \tag{4.22}$$

The Laplacian matrix L plays a pivotal role in the control strategy, enabling vehicles to adapt their speeds and following distances based on their neighbours' behaviours. By incorporating this network-based approach, our proposed algorithm ensures cooperative and adaptive control within the platoon, ultimately leading to safer and more efficient autonomous truck platooning.

4.3 Implementation

This section delves into the practical implementation of the proposed algorithm, achieved through the integration of ROS with Gazebo [**sharifi_modelling_2018**] and ROS Visualization Tool (RViz) [**kam_rviz_2015**]. Gazebo is an integrated, physics-based 3D simulator in ROS that accurately models complex robots and environments while RViz is a powerful tool in the ROS ecosystem that visually represents three-dimensional data, making it easier to understand and work with data generated by sensors, actuators, and algorithms. These tools collectively enable the development, testing, and analysis of the platooning system within a controlled environment. Additionally, the Unified Robot Description Format (URDF) is crucial in robotics for accurately representing a robot's physical structure using a standardised XML-based description [**fabian_pose_2020**].

The URDF acts as a bridge between mechanical design and simulation, accurately representing the different vehicle components such as chassis, sensors and degrees of freedom. Gazebo provides a physics-based 3D simulation environment that replicates real-world dynamics and interactions. Notably,

this implementation successfully replicated a highway environment. Finally, RViz enhances data comprehension since allows visualising sensor readings.

To offer a comprehensive context, an overview of the ego vehicle's system architecture is presented in Figure 4.3. This illustration aids in understanding the broader framework.

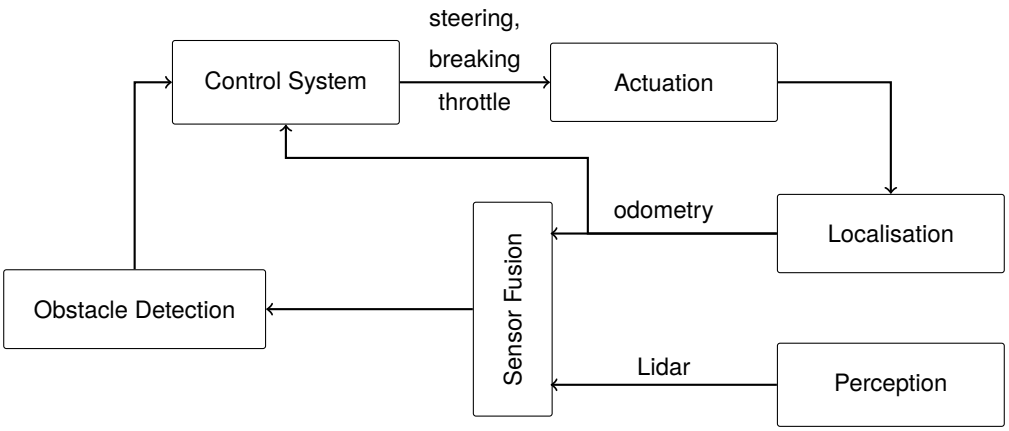


Figure 4.3: Holistic view of system architecture for the ego vehicle. Illustrating the interplay between control system, actuation, localisation, perception, sensor fusion, and obstacle detection components.

Moving forward, specific contributions within each component of the architecture are detailed, as well as further communications with the different agents. These are localisation, perception, sensor fusion and lastly the control system. Additionally, considerations about the implementation of the platoon-distributed system are also included.

4.3.1 Localisation

The primary source of localisation information for our platooning system is the odometer. The odometer measures the distance travelled by the vehicle's wheels and calculates the change in position over time. This information is valuable for tracking the agent's trajectory and maintaining an up-to-date estimate of its pose. The ROS ecosystem enables access to the odometry data of every agent, thus we achieve real-time, high-precision localisation. This data forms the basis for subsequent control actions, used individually by the ego and the follower vehicles.

4.3.2 Perception

The perception module provides the control system with an accurate description of the environment. This encompasses both moving and static objects, as well as an estimation of the road geometry and gradient that the vehicles traverse.

The perception element is developed centred around a LiDAR sensor. The sensor measures distances by sending laser pulses and calculating the time taken for the pulses to return after hitting an object. Thus providing high-resolution point cloud data, enabling the perception system to create a detailed and precise representation of the surroundings.

In this context, the Hokuyo UTM-30LX 2D laser scanner [**noauthor_utm-30lx_nodate**] is chosen to be incorporated into the ego vehicle. The specifications of this LiDAR are outlined in Table 4.1.

Table 4.1: Specifications of the Hokuyo UTM-30LX LiDAR.

Characteristic	Value
Range	0.01 m - 30 m
Scanning angular range	270°
Angular resolution	0.25°
Scanning rate	40 Hz

4.3.3 Sensor Fusion

The sensor fusion module is a critical component that combines data from multiple sources, such as odometry and LiDAR, to estimate obstacle positions accurately. Moreover, it employs advanced techniques to differentiate between static and dynamic obstacles within the environment.

To distinguish between static and dynamic obstacles in the spatial domain, the module employs an optical flow-inspired method based on LiDAR measurements over time. This involves comparing the positions of LiDAR points between consecutive scans to estimate the motion of obstacles. This motion is calculated by computing the displacement. The ideal optical flow is naturally zero for static points and different from zero for dynamic points. However, the presence of LiDAR measurement noise can highly impact this process. Therefore, empirically determined thresholds near zero and a second higher arbitrary threshold were used to classify static and dynamic obstacles respectively.

Through the integration of a KF, the sensor fusion module predicts obstacle velocities by leveraging the principles established earlier filter's formulation (see subsubsection 4.2.1.3). Furthermore, assuming the obstacle's motion always follows a constant acceleration, the system state vector is expressed as follows:

$$\begin{bmatrix} 1 & 0 & h & 0 & \frac{h^2}{2} & 0 \\ 0 & 1 & 0 & h & 0 & \frac{h^2}{2} \\ 0 & 0 & 1 & 0 & h & 0 \\ 0 & 0 & 0 & 1 & 0 & h \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.23}$$

where h is the time interval. This model expresses the x-coordinate, y-coordinate, x-velocity, y-velocity, x-acceleration, and y-acceleration of the obstacle in sequence.

In the process of system estimation, the noise covariance matrix $\mathbf{Q}_{\text{KF}}$ is defined to account for the uncertainties and noise present in the system. It encompasses the variances of position, velocity, and

acceleration in both x and y directions, such that:

$$\mathbf{Q}_{\text{KF}} = \begin{bmatrix} \sigma_{wx}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{wy}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{wvx}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{wvy}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{wax}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{way}^2 \end{bmatrix}. \tag{4.24}$$

Each covariance σ was chosen empirically, and the cross-covariance was assumed always 0. The observation matrix $\mathbf{H}_{\text{KF}}$ is formulated to estimate the velocity of the obstacle by applying the constant acceleration motion model. Consequently, the input of the observation model is the centre position of the point cloud after clustering, and its coordinates x, y serve as the design reference of the observation matrix, thus

$$\mathbf{H}_{\text{KF}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.25}$$

In the observation process, the noise covariance matrix $\mathbf{R}_{\text{KF}}$ is calculated to weigh the state correction based on the current state and the observed noise. The diagonal elements represent the variance of the x and y coordinates, reflecting the measurement noise covariance resulting from the observation noise, such that

$$\mathbf{R}_{\text{KF}} = \begin{bmatrix} \sigma_{vx}^2 & 0 \\ 0 & \sigma_{vy}^2 \end{bmatrix}. \tag{4.26}$$

Lastly, the KF is thus initialised as shown in Table 4.2.

Table 4.2: Initial setting of the Kalman Filter parameters.

Initial Setting of Parameters	
position x	read from LiDAR
position y	read from LiDAR
velocity x	0.0
velocity y	0.0
acceleration x	0.0
acceleration y	0.0

4.3.4 Control of Vehicles

The control module is built upon the NMPC formulation introduced earlier, in (4.7) for the ego vehicle and in (4.18) for the follower vehicles. To find a locally optimal solution to the Optimal Control Problem (OCP), the state-of-the-art numerical optimal control software CasADi [**andersson_casadi_2019**] is employed. The software package allows for reformulating the continuous-time optimal control problem into a NLP

problem. The resulting NLP is solved using a nonlinear Interior-Point Method (IP) solver via a direct multiple shooting combined with numerical integration.

After every NMPC step, it is a common practice to warm-start the next optimal control problem, shifting the vector of control inputs over one time instant and adding an initial guess, often the zero vector, for the last time instant. Similarly, the penalty factors are shifted, and a vector of ones is added for the last time instant.

To visualise the hierarchical structure of nodes and topics in the ROS system, one can refer to Figure 4.4, where nodes are depicted as ellipses, and topics are represented as lines connecting them. This visualisation enables us to observe the flow of information and dynamic interactions between nodes within our control system. It significantly enhances our comprehension of the vehicle's decision-making process.

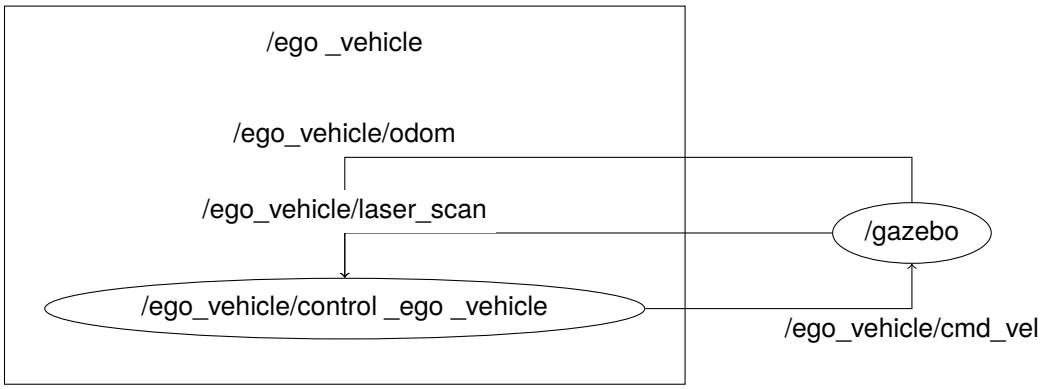


Figure 4.4: Hierarchical representation of nodes and topics in the control system via the `rqt graph` GUI tool.

4.3.5 Distributed Control

The CACC, that is the distributed control scheme used in this research work, is facilitated through ROS. The foundational principles underlying a ROS-based implementation encompass key elements such as *nodes*, *messages*, *topics*, and *services*. As depicted in Figure 4.5, the ROS master node, known as `roscore`, assumes a pivotal role in this architecture enabling inter-node communication.

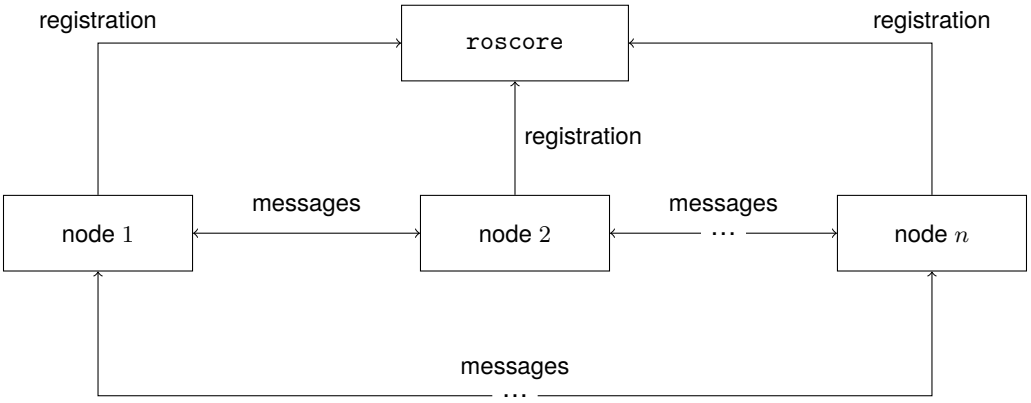


Figure 4.5: ROS architecture and node communication illustration. The `roscore` orchestrates the exchange of information among nodes, enabling collaborative coordination and data flow. Inspired and adapted from [**koubaa_coros_2015**].

A way to conceptualise the interactions within a ROS-based system is by envisioning it as a directed graph [**jiang_sign-consensus_2017**]. Recalling the introduction to graph theory performed in subsection 2.3.1, a semi-formal definition of a ROS graph is thus

Definition 4 (ROS-graph)**.** *A ROS-graph is denoted as* $\mathcal{G} := (\mathcal{N}, \mathcal{T}, \mathcal{S}, \mathcal{E}, \mathcal{D}, \mathcal{C}, \mathcal{X}, \lambda)$*, where:*

- $\mathcal{N}$ *is the set of vertices corresponding to ROS nodes;*
- $\mathcal{T}$ *the set of topics;*
- $\mathcal{S}$ *the set of services;*
- $\mathcal{C}$ *ordered set of object classes;*
- $\mathcal{X}$ *a set of labels on vertices;*
- $E \subset (\mathcal{N} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{N}) \cup (\mathcal{N} \times S) \cup (S \times \mathcal{N})$ *is a set of directed edges to represent publishing of, and subscription to, topics and provision of, and subscription to, services, respectively;*
- $D : E^- \to C^*, E^- = \mathcal{T} \cup (\mathcal{N} \times \mathcal{S}) \cup (\mathcal{S} \times \mathcal{N})$*, is a data descriptor function with* $\mathcal{C}^*$ *is a notation for finite sequences of entries from the set of a data object classes* $\mathcal{C}$*, which are used in services and topics to send information between nodes.*

Each of $\mathcal{N}, \mathcal{T}, \mathcal{S}$ *are labelled by the function* $\lambda : \mathcal{N} \cup \mathcal{T} \cup \mathcal{S} \to \mathcal{X}$*.*

Thus, in this dissertation, where vehicles communicate and coordinate, ROS acts as a facilitator. Each vehicle functions as a distinct node within the ROS ecosystem, exchanging vital information and thus orchestrating complex manoeuvres.

In the context of CACC, the communication and interaction between vehicles can be represented using a ROS graph $\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{S}, \mathcal{E}, \mathcal{D}, \mathcal{C}, \mathcal{X}, \lambda)$, likewise asserted in above. Thus $\mathcal{G}$ is such that:

- $\mathcal{N}$ represents the set of vertices corresponding to ROS nodes, including the ego vehicle and different followers in the platoon. One can mathematically define it as

$$\mathcal{N} = \{0, 1, 2, \ldots, n_f\}. \tag{4.27}$$

  Where, n_f represents the number of follower vehicles in the network, equalling a total of $1 + n_f$ ROS nodes.

- $\mathcal{T}$ is the set of topics through which vehicles communicate. Thus,

$$\mathcal{T} = \{\mathcal{T}_{\text{ego}}, \mathcal{T}_{\text{follower 1}}, \ldots, \mathcal{T}_{\text{follower } n_f}\}, \tag{4.28}$$

  these indexes take the format of the numbering of nodes established in $\mathcal{N}$.

- $\mathcal{C}$ is the set of object classes, formally expressed as:

$$\mathcal{C} = \{C_{\text{odom}}, C_{\text{control}}, C_{\text{sensor}}\} \tag{4.29}$$

  That is each class C_i within $\mathcal{C}$ defines a distinct category of information exchanged among platoon

vehicles. The specific composition of $\mathcal{C}$ recalls the odometry data, the input control commands, and the sensor data.

- $\mathcal{X}$ is the set of labels assigned to the vertices in the network, thus representing vehicle roles. Formally expressed as:

$$\mathcal{X} = \{X_0, X_1, X_2, \ldots, X_{n_f}\} \tag{4.30}$$

where X_0 represents the label assigned to the ego vehicle, X_1 and X_2 represent the label assigned to the first and second follower vehicles, respectively. X_{n_f} naturally represents the label assigned to the n_f-th follower vehicle.

- $\mathcal{E}$ is the set of directed edges in the network, representing communication relationships between nodes, i.e. different vehicles. Formally expressed as:

$$\mathcal{E} = \{(v_i, v_j) \mid v_i, v_j \in \mathcal{N}\}, \tag{4.31}$$

where, (v_i, v_j) denotes a directed edge from node v_i to node v_j within the network, and $v_i, v_j \in \mathcal{N}$ indicates that both v_i and v_j are vertices (ROS nodes) in the network.

- $\mathcal{D}$ is the data descriptor function that maps directed edges in the network, $\mathcal{E}^-$, to sequences of data object classes, $\mathcal{C}^*$, such that

$$\mathcal{D} : \mathcal{E}^- \rightarrow \mathcal{C}^* \tag{4.32}$$

The output of D is a sequence of data object classes that characterises the information content and format of messages exchanged along each directed edge.

Practically, the role of ROS is paramount in orchestrating seamless communication and cooperation among platoon vehicles. The platoon's dynamics begin with the initial assignment of roles, designating vehicles as either the ego or a follower based on predefined criteria or real-time communication with other platoon members. Each vehicle node is attributed a specific role label, as defined within $\mathcal{X}$. The platoon ego, driven by its local sensor data and control unit, maintains a constant desired velocity and adheres to the nominal path. Simultaneously, it periodically broadcasts its state, encompassing essential data such as position, velocity, and control inputs, via a designated communication channel represented by topic $\mathcal{T}_{\text{ego}}$.

In the followers' domain, these vehicles subscribe to the ego's state information, which flows through the very same topic $\mathcal{T}_{\text{ego}}$. Armed with real-time insights into the ego's behaviour, each follower autonomously computes its desired trajectory and corresponding control commands. These commands, originating from the follower's calculations, are then disseminated via an exclusive communication channel, aptly termed $\mathcal{T}_{\text{follower}}$.

Follower vehicles communicate with each other through the topic $\mathcal{T}_{\text{follower}}$. Each follower considers the control commands of the preceding follower vehicle, aiming to maintain a safe distance and synchronised motion. The data descriptor function D maps the topic edges to data object classes that include position, velocity, and other relevant information. Follower vehicles use the received control commands to adapt their velocity, acceleration, and braking based on the current traffic situation. The ego and followers adjust their behaviours to ensure safe and cooperative driving within the platoon. The system continues

to operate until a termination condition is met, such as a designated exit point or completion of a specific mission.

Figure 4.6 visually captures the essence of this communication scheme, drawing inspiration from the CACC framework in Figure 3.2, and highlights the labels and topics integral to the platoon's coordinated operation.

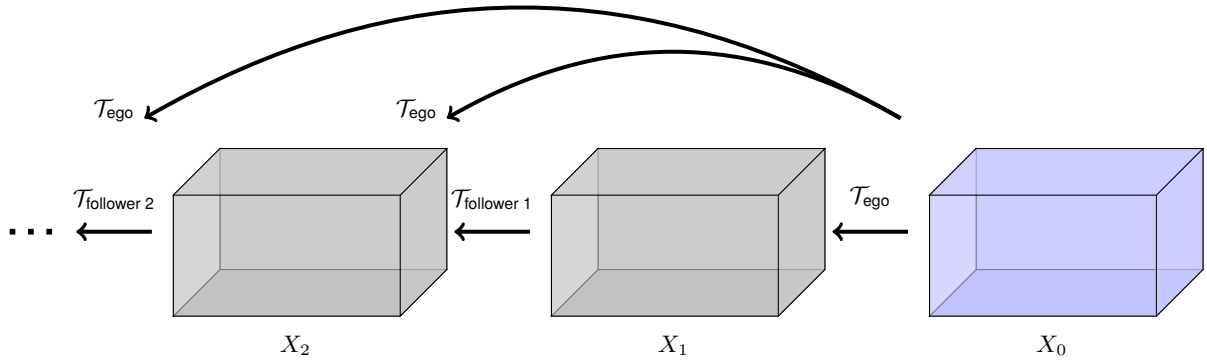


Figure 4.6: A visual representation of the information flow in the platoon dynamics within ROS. The ego vehicle is represented in *blue* and the different followers in *light grey*.

Chapter 5

Validation

In this chapter, the proposed method will be rigorously evaluated to assess its performance, effectiveness, and robustness in various scenarios. The evaluation aims to validate the approach's capability to address the challenges of lane-changing manoeuvres, obstacle avoidance, overtaking, and abrupt braking for the ego vehicle within the platoon.

5.1 Experimental Setup and Methods

The codebase and resources associated with the work presented in this dissertation are available in the following repository: GitHub Repository [1]. This repository serves as a comprehensive source for accessing the software, data, and documentation pertinent to the experiments and results discussed in this chapter.

5.1.1 Performance Metrics

The evaluation utilises a set of key performance metrics to thoroughly assess the effectiveness and efficiency of the truck platooning system proposed. These metrics include computation effort, control effort, and, consequently, energy efficiency, state stability (platoon stability), lane-change success rate, and collision avoidance rate. Computation resources and system stability are meticulously monitored. Furthermore, the ability of the system to execute successful lane changes and avoid collisions is measured. To provide context, the system's performance is compared with established baseline models, enabling a robust evaluation of its capabilities and areas for potential improvement.

5.1.2 Computational Resources

In this study, the computational resources utilised include a MacBook Pro powered by an Apple M1 Pro chip with 16GB of RAM. All experiments and simulations were conducted within a virtual machine environment, facilitated by Parallels Desktop, using Ubuntu 20.04 and ROS Noetic. It is important to note

[1] https://github.com/blourenco217/Master-Thesis-Simulation

that this experiment does not rely on GPU acceleration, ensuring compatibility with the MacBook Pro's hardware configuration.

5.1.2.1 Experiment environment

The evaluation of the proposed solution was conducted within a simulated highway environment, utilising the Gazebo simulation platform. This environment, as illustrated in Figure 5.1, provided a realistic and controlled setting for assessing the performance of our platooning system. The Gazebo simulation framework facilitated the recreation of highway scenarios, enabling comprehensive testing and analysis of the system's capabilities in a simulated but representative context.

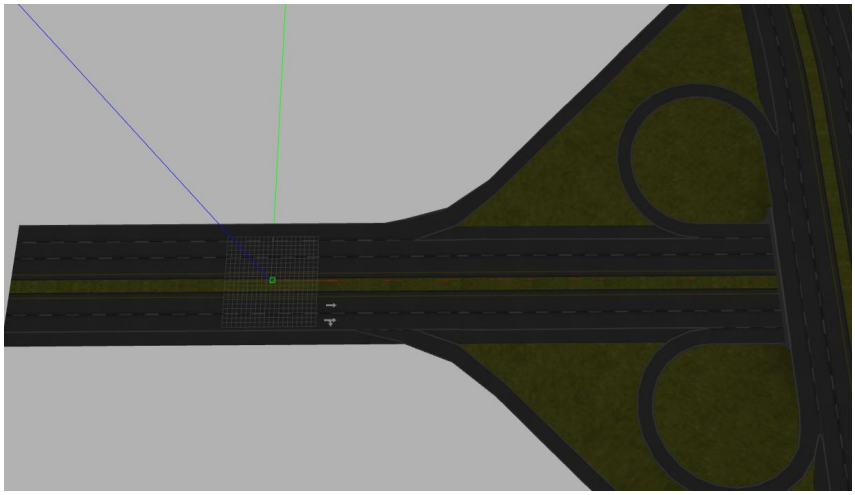
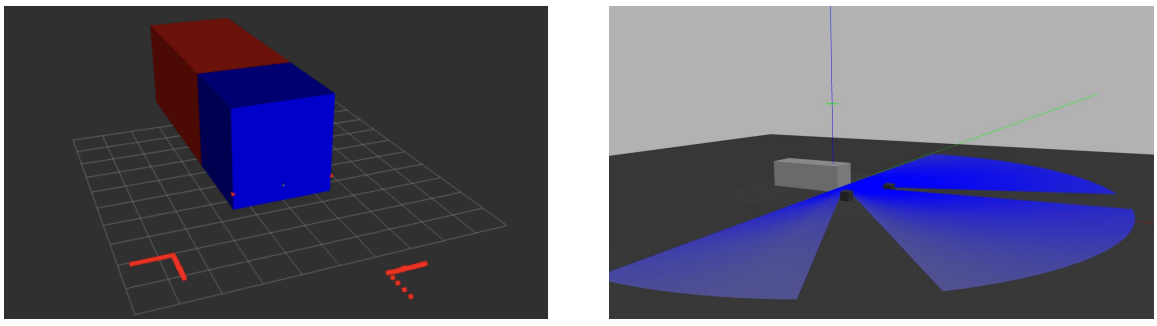


Figure 5.1: Simulated highway environment for proposed solution testing in Gazebo.

Figure 5.2 illustrates the integration of RViz and Gazebo in the simulation environment, showcasing their pivotal roles in fine-tuning LiDAR data. The simulation features a two-segment truck (tractor + trailer) navigating an environment with strategically positioned obstacles. In the RViz visualisation (Figure 5.2a), the red scatter represents the data points received from the LiDAR sensor, providing real-time insights into the surrounding environment. Simultaneously, in the Gazebo simulation (Figure 5.2b), the blue rays denote the range area captured by the LiDAR, offering a comprehensive view of the sensor's coverage. This integration facilitates precise LiDAR data analysis and optimisation, contributing to the development of an efficient platoon control system.



(a) RViz Integration. **(b)** Gazebo Simulation.

Figure 5.2: Visualization of LiDAR data integration.

5.2 Experiments

In this section, a detailed description of the experiments conducted to evaluate the proposed approach is provided. These experiments encompass a range of scenarios designed to challenge the capabilities of the approach.

During the valuation process, we performed limited parameter tuning to optimise the performance of the proposed algorithm. Specifically, we selected the state penalty matrix $\mathbf{Q}$ and the control input penalty matrix $\mathbf{R}$, both of which contribute to the objective cost function described in (4.8). The state penalty matrix was chosen as a diagonal matrix such that:

$$\mathbf{Q} = \mathrm{diag}(100, 150, 10, 5, 10, \underbrace{5, 5, \ldots, 5}_{n\text{-times}}) \tag{5.1}$$

Here, n represents the number of trailers in the controlled vehicle model (recall section 2.1). These values correspond to the penalty for the orientation of each tractor or trailer segment in the model, and the same penalty value was applied to all segments. Therefore, $\mathbf{Q}$ has a dimension of $(5+n) \times (5+n)$.

Additionally, the control input penalty matrix $\mathbf{R}$ was defined as a fixed-size matrix:

$$\mathbf{R} = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \tag{5.2}$$

Furthermore, the exponential weight penalty for the soft constraint in (4.15) was set to $\beta = 3.7$. The prediction horizon was fixed at $N = 10$, and the controller operates at a frequency of 10 Hz. Additionally, all topics and services in the ROS system are broadcasted at uniform time intervals to ensure standardisation.

All the forthcoming experiments were conducted using trucks of comparable shapes, each featuring a single trailer in a simple tractor-trailer configuration. The tractor segment has a length of $l_0 = 3.5$ m, with a hitching offset denoted as of $m_0 = 0.5$ m. The trailer's length is $l_1 = 6$ m.

The LiDAR was empirically tuned to operate within a narrow range, roughly matching the lane's width. The selection of values for the covariance matrices $\mathbf{Q}_{\text{KF}}$ and $\mathbf{R}_{\text{KF}}$ is a critical aspect of the KF design. The values of $\mathbf{Q}_{\text{KF}}$, which represent process noise, were intentionally set to low values such as

$$\sigma_x = \sigma_y = \sigma_{vx} = \sigma_{vy} = \sigma_{ax} = \sigma_{ay} = 0.01\ . \tag{5.3}$$

These low values reflect our desire to have a high confidence in the accuracy of the predicted state of the system, as it relies on the dynamic model's accuracy.

Conversely, the values of $\mathbf{R}_{\text{KF}}$, which correspond to measurement noise, were set higher, specifically

$$\sigma_{vx} = \sigma_{vy} = 0.1\ . \tag{5.4}$$

This decision was driven by the nature of sensor measurements, which inherently contain more noise and

uncertainty. By assigning higher values to $\mathbf{R}$, we ensure that the Kalman filter places greater emphasis on sensor measurements and adapts to changes in the observed data. This balance between process noise and measurement noise is crucial in achieving optimal filter performance, ensuring that the filter effectively fuses the predictions from the dynamic model with real sensor measurements to provide an accurate and reliable estimate of the system's state.

It is important to note that our goal was not to fine-tune the NMPC extensively but rather to assess its overall performance within this specific experimental setup.

5.2.1 Lane Changing Manoeuvres

In order to assess the proposed approach's capability to perform basic lane-changing manoeuvres while maintaining platoon integrity and stability, a series of experiments were conducted. These tests aimed to evaluate the approach's performance across different platoon configurations, focusing on two scenarios: one with a platoon composed of three trucks and another with four trucks. This was performed on similar-shaped trucks with only one trailer - i.e. a tractor-trailer configuration.

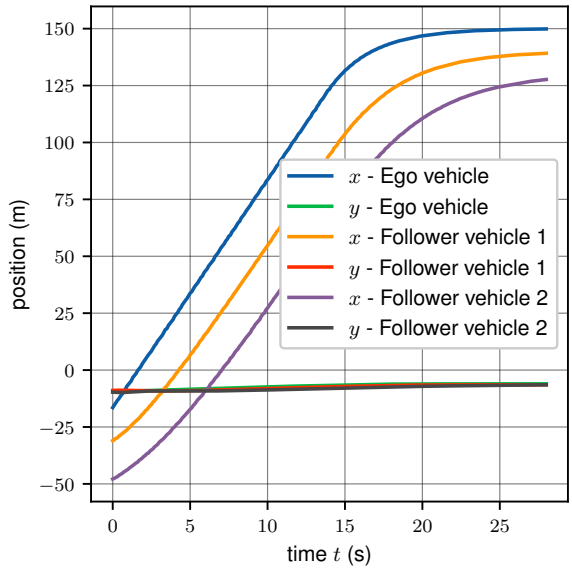
Figure 5.3 presents comprehensive performance metrics for the basic lane-changing manoeuvre. Particular emphasis is given to the evolution of position over time (Figure 5.3a and Figure 5.3c), as well as the velocity profile over time (Figure 5.3b).

Furthermore, Figure 5.4 presents a sequence of snapshots from the Gazebo simulation, providing a visual representation of the platoon's dynamic movement. These snapshots correspond to the same experimental run as the previously displayed plots.

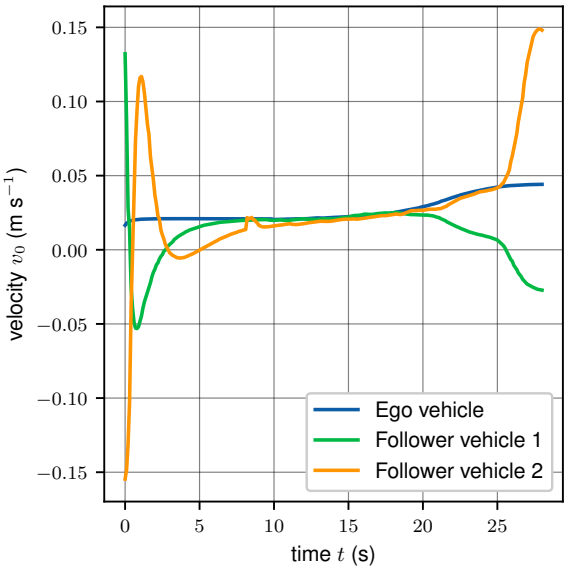
Analogously, Figure 5.5 offers a comprehensive overview of performance metrics during a lane-changing manoeuvre involving four vehicles. Notable aspects include the evolution of vehicle positions over time (Figure 5.5a and Figure 5.5c) and the corresponding velocity profiles (Figure 5.5b). While Figure 5.6 captures key moments from a 14-second video clip, portraying a coordinated lane-changing manoeuvre within a platoon of four vehicles. These visual snapshots align with the experimental scenario explored through the previously detailed graphs.

The provided plots showcase performance metrics for lane-changing manoeuvres in platoons of three and four trucks. In the lane-changing manoeuvre, the Time Evolution graphs (Figure 5.3a and Figure 5.5a) illustrate smooth execution with distinct phases, allowing for comparisons across different platoon sizes. The Velocity Profile graphs (Figure 5.3b and Figure 5.5b) highlight dynamic behaviours, revealing acceleration and deceleration patterns that contribute to overall platoon stability. The position tracking graphs (Figure 5.3c and Figure 5.5c) demonstrate the platoon's ability to maintain its desired formation during the manoeuvre, reinforcing the robustness of the proposed algorithm. These images collectively emphasise the efficacy of the control approach, demonstrating consistent and reliable performance across varying platoon sizes in executing complex lane-changing manoeuvres.

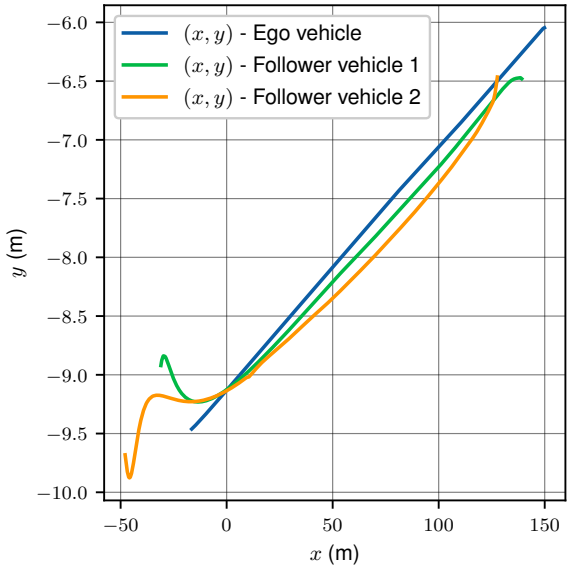
The experiments demonstrated that the proposed approach successfully managed basic lane-changing makeovers within the platoon for both three-truck and four-truck configurations. The control effort exhibited efficiency, and the platoon's stability was well-maintained throughout the manoeuvres. Sequential images showcasing the lane-changing performance of the platoon revealed smooth



(a) Time evolution of lane changing.



(b) Velocity profile during lane changing.



(c) Position tracking during lane changing.

Figure 5.3: Performance metrics for lane changing manoeuvres on a three trucks platoon configuration.

transitions between lanes. The ego vehicle exhibited controlled acceleration and deceleration while maintaining safe distances from neighbouring vehicles. These simpler experience makes it promising for further and more complex approaches, such as overtaking manoeuvres.

5.2.2 Static and Dynamic Obstacle Avoidance

A stationary obstacle was placed along the trajectory of the platoon to evaluate how well the approach detects and avoids such obstacles. The ego vehicle initiated avoidance manoeuvres upon detecting the obstacle in its path.

Figure 5.7 presents a set of performance metrics given the described scenario. The images include

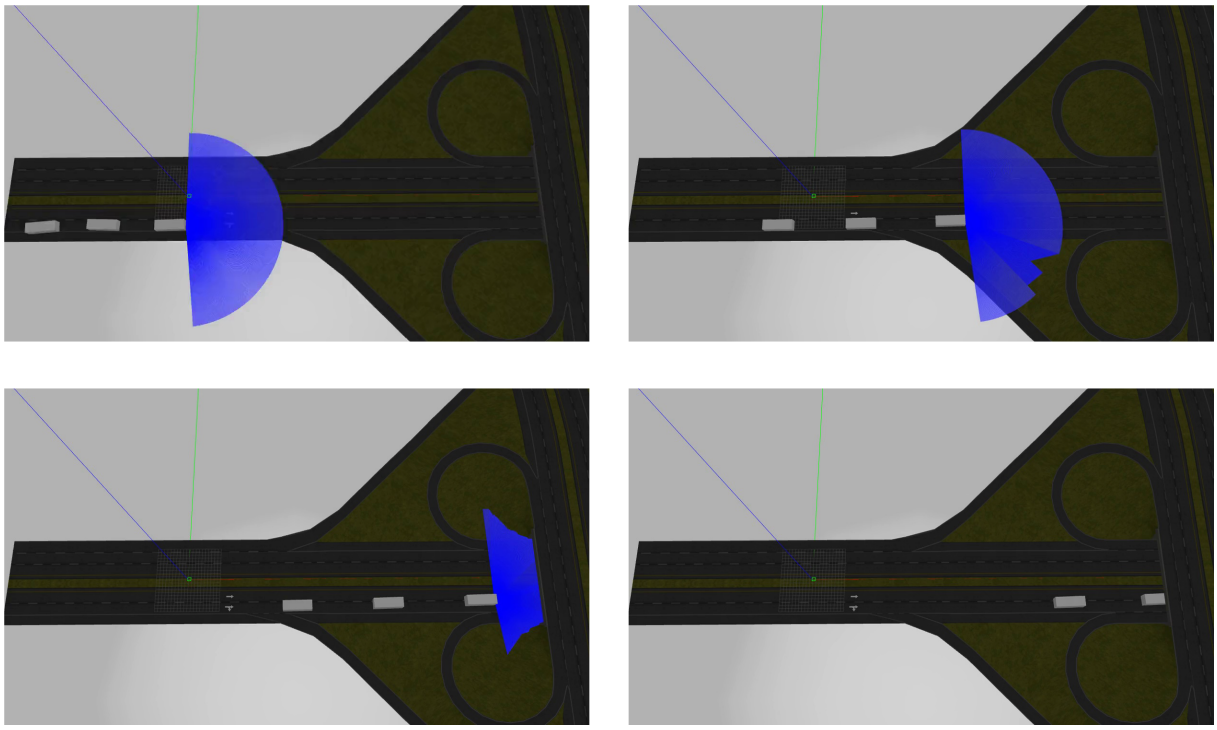


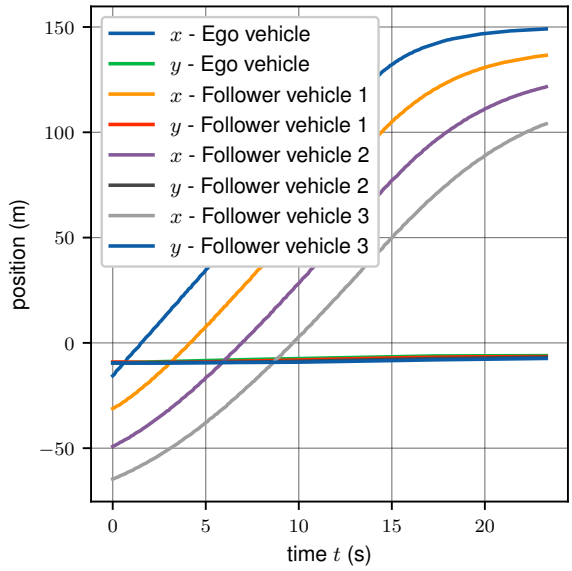
Figure 5.4: Sequential images illustrating a successful lane-changing manoeuvre. This set of four equally spaced frames, captured from a 16-second clip, depicts the lane-changing performance of the ego vehicle (equipped with LiDAR) and two following vehicles.

the time evolution of vehicle positions (Figure 5.7a) and velocities (Figure 5.7b), as well as the tracked positions (Figure 5.7c) during these experiments.

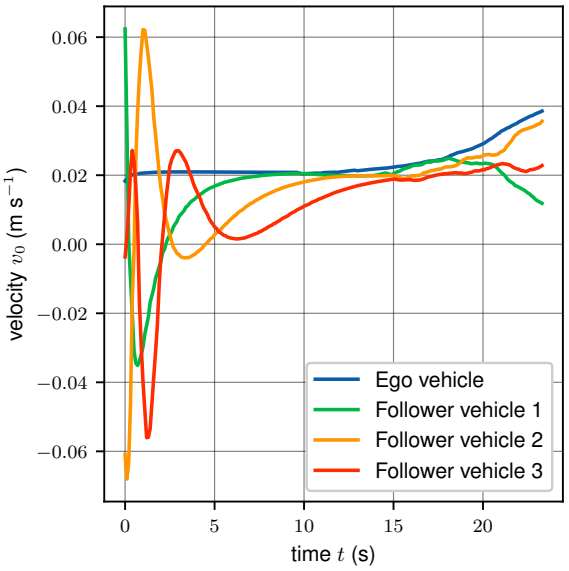
In Figure 5.7a, notably, the x position demonstrates a synchronised behaviour, indicating that the vehicles maintain a constant inter-vehicle distance throughout the manoeuvre. Meanwhile, the y position reveals a ripple effect originating around second 4 for the ego vehicle, propagating this divergence from the nominal trajectory to the follower vehicles. This observation underscores the algorithm's proficiency in orchestrating a cohesive platoon, ensuring that all vehicles faithfully follow the lead vehicle's trajectory.

The velocity profiles depicted in Figure 5.7b showcase the algorithm's capability to adapt vehicle speeds when confronted with static obstacles. Specifically, at second 4, the ego vehicle, at second 5, the first following vehicle, and finally, at second 7, the last vehicle in the platoon react to the presence of the obstacle by smoothly decelerating. This orchestrated speed reduction aligns with the ego vehicle's detection and deviation from its initial nominal trajectory, thereby preventing any abrupt changes in the platoon's movement.

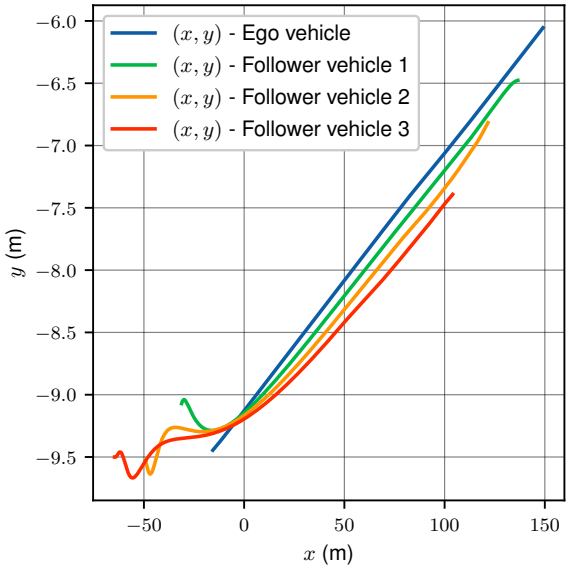
In the context of Figure 5.7c, we witness the meticulous tracking of the platoon's vehicle positions over time. Here, the algorithm's effectiveness in circumnavigating the obstacle is particularly evident. The discernible "jump" in the trajectory corresponds to the precise avoidance of the obstacle, demonstrating the algorithm's capacity for decisive and obstacle-aware manoeuvres. The oscillations observed in the trajectories of the follower vehicles just before overtaking the obstacle stem from the sudden trajectory shift executed by the ego vehicle. These oscillations highlight the algorithm's ability to navigate the platoon seamlessly through complex scenarios while maintaining the desired safety margins.



(a) Time evolution of lane changing.



(b) Velocity profile during lane changing.



(c) Position tracking during lane changing.

Figure 5.5: Performance metrics for lane changing with on a four trucks platoon configuration.

In the sequential snapshots presented in Figure 5.8, we witness the platoon's coordinated manoeuvre around a static obstacle. This series of images showcases the algorithm's ability to effectively guide the platoon's vehicles smoothly and efficiently navigating past the obstacle. These snapshots demonstrate the algorithm's real-time decision-making and trajectory adjustments, ensuring the safe and obstacle-free progression of the platoon.

The next experiment delves into the assessment of the same proposed algorithm, this time in the context of dynamic obstacle avoidance. To accurately represent the obstacles, a kinematic bicycle model was employed, and their movements were controlled using a PID controller. Additionally, a reference trajectory for each vehicle was planned by incorporating randomisation techniques to determine the desired lane.

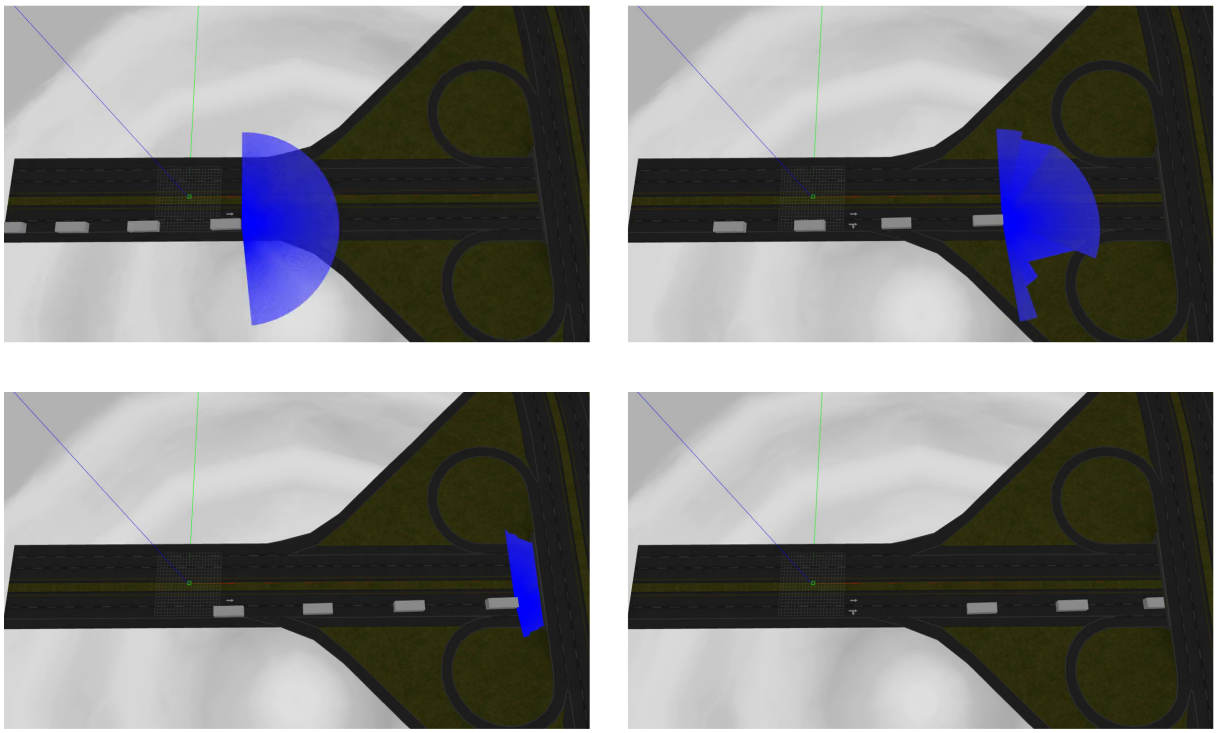


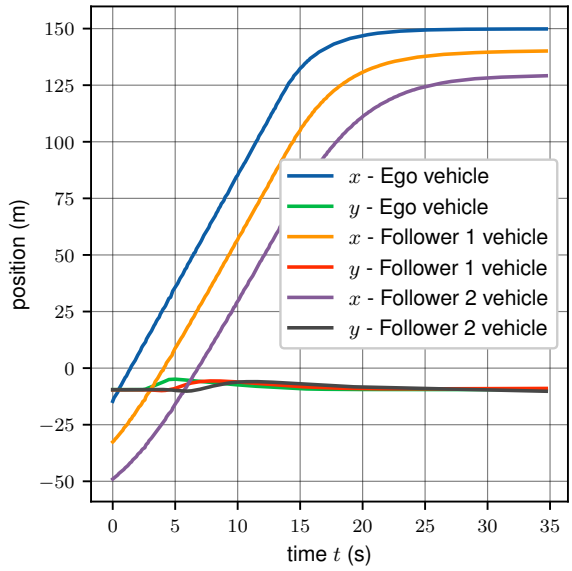
Figure 5.6: Snapshot from a 14-second video clip showcasing lane-changing manoeuvres . In this frame, the ego vehicle, equipped with LiDAR, leads a platoon of three following vehicles as they execute a coordinated lane change.

As the platoon encounters dynamic obstacles in its path, the algorithm's performance is examined to ensure its adaptability to changing scenarios. Figure 5.9 presents the set of performance metrics of the present experience.

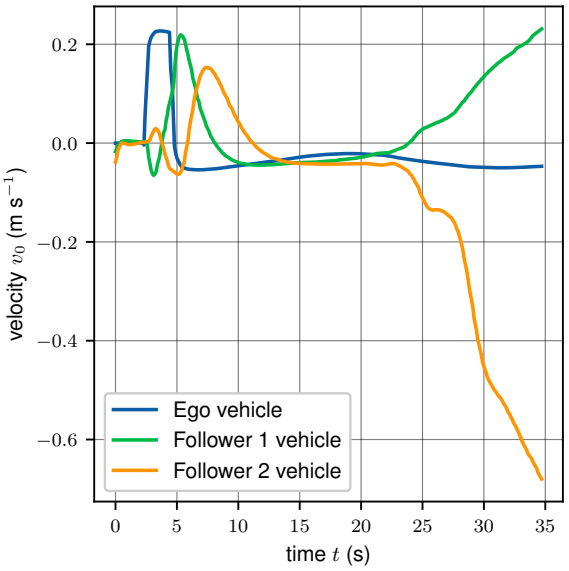
In Figure 5.9a it becomes apparent that the ego vehicle and the follower vehicle respond synchronously to the dynamic obstacle. The plot highlights the parallel behaviour of their positions along the x-axis, indicating that the vehicles maintain a constant distance from each other. Additionally, on the y-axis it demonstrates a *ripple effect* caused by the obstacle's detection at approximately second 4 for the ego vehicle. This *ripple* propagates along the path, illustrating how the vehicles effectively follow each other's trajectories.

In Figure 5.9b the ego vehicle exhibits a smooth deceleration as it approaches the obstacle at approximately second 4. This deceleration results in a downward slope in the velocity curve, indicating a controlled reduction in speed. Following successful obstacle avoidance, the ego vehicle smoothly accelerates to regain its initial speed. Importantly, the follower vehicles also adjusts its velocity to maintain platoon coherence during the ego vehicle's manoeuvre, demonstrating a controlled reduction in speed and subsequent acceleration.

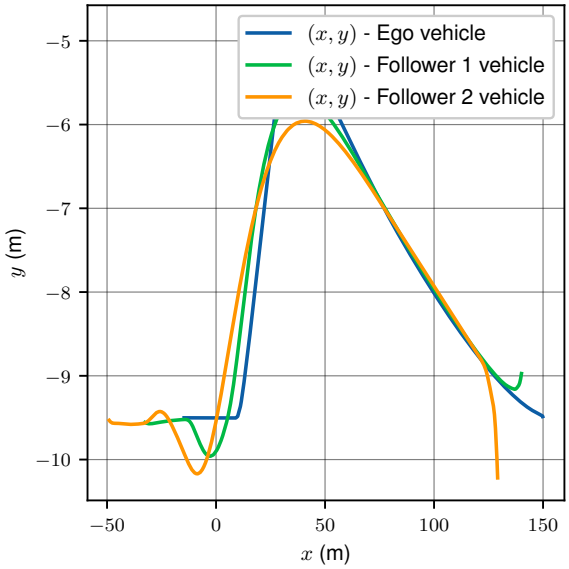
Figure 5.9c emphasises how effectively the vehicles contour the dynamic obstacle. Notably, a *jump* in the trajectory corresponds to the avoidance of the obstacle, showcasing the algorithm's ability to orchestrate this manoeuvre seamlessly. The slight oscillation in the trajectory of the follower vehicles just before overtaking the obstacle is a result of the ego vehicle's sudden trajectory change. This analysis confirms the algorithm's capacity to navigate dynamic obstacles while ensuring platoon coherence.



(a) Time evolution the platoon's position.



(b) Velocity profile.



(c) Position tracking.

Figure 5.7: Performance metrics for a static obstacle avoidance scenario with a three vehicles platoon configuration.

In Figure 5.10 we denote the sequential snapshots from this manoeuvre.

5.2.2.1 Baseline

The obstacle avoidance performance of the system was compared with a well-established baseline model, discussed in the state-of-the-art segment (see section 3.1). This comparison allows for a robust evaluation of the system's capabilities, highlighting areas for potential improvement.

To assess its performance, a quadratic barrier function was employed represented as follows:

$$h(\boldsymbol{p}_{\text{ego}}) = \frac{1}{2} \left\| \boldsymbol{p}_{\text{ego}} - \boldsymbol{b} \right\|^2 - \frac{1}{2} r^2 \, . \tag{5.5}$$

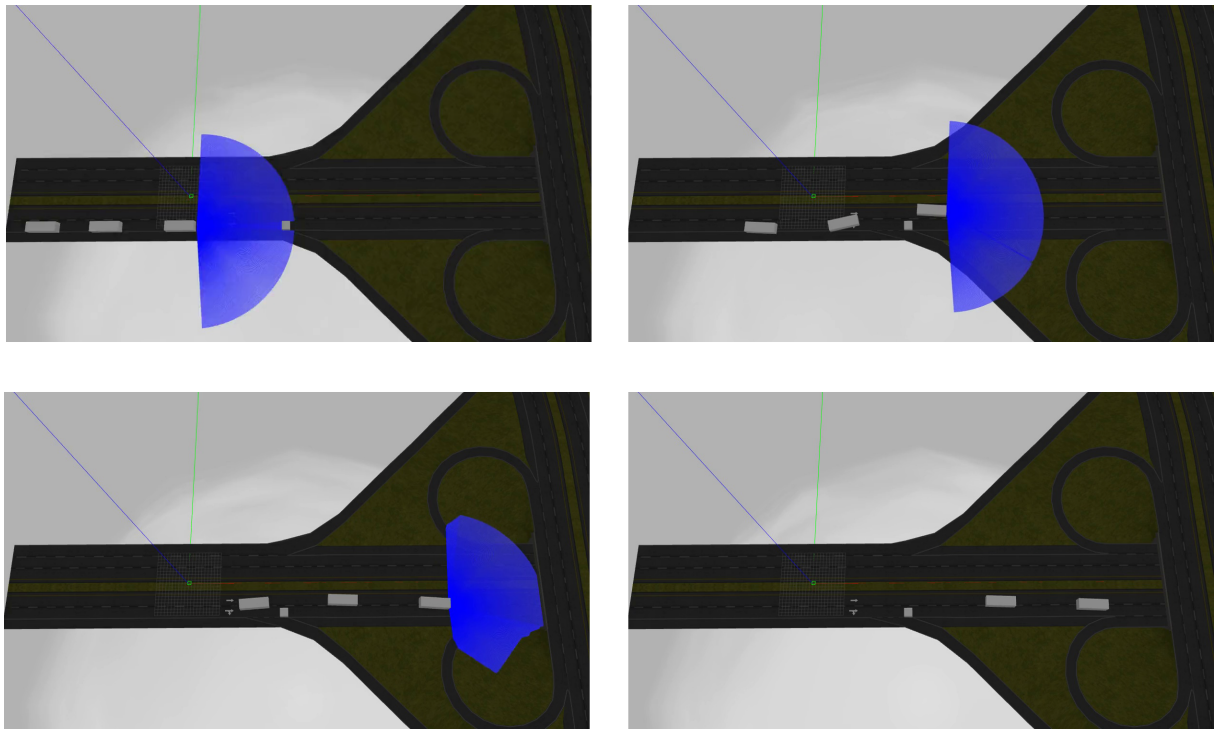


Figure 5.8: Snapshot from a 13-second video clip showcasing platoon of vehicles navigating past a static obstacle. In this frame, the ego vehicle, equipped with LiDAR, leads a platoon of two following vehicles as they execute a coordinated lane change. The static obstacle is a *light-grey* cube placed on the rightmost lane of the highway environment.

Here, p_{ego} denotes the pose of the ego vehicle, while b represents the leftmost boundary of the obstacle. The parameter γ acts as the convergence rate for the CBF and was tuned to a fixed value, independent of the ego vehicle's position, set at $\gamma = 0.7$. This choice strikes a balance between the feasibility domain and achieving the fastest possible convergence. Additionally, recalling problem formulation in (3.2) the objective function was altered by the addition of a quadratic cost of the slack variable ω, that is, by the sum of

$$\psi\left(\omega_k\right) = \omega_k^2. \tag{5.6}$$

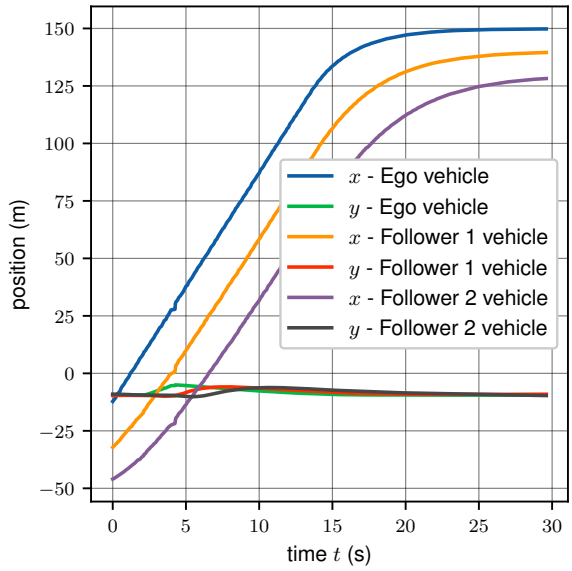
The reference tracking function maintained the same framework as in the proposed algorithm in (4.8) such that

$$q\left(x_k, u_k\right) = \left(x_k - r_k\right)^\top \mathbf{Q}(x_k - r_k) + u_k^\top \mathbf{R} u_k. \tag{5.7}$$

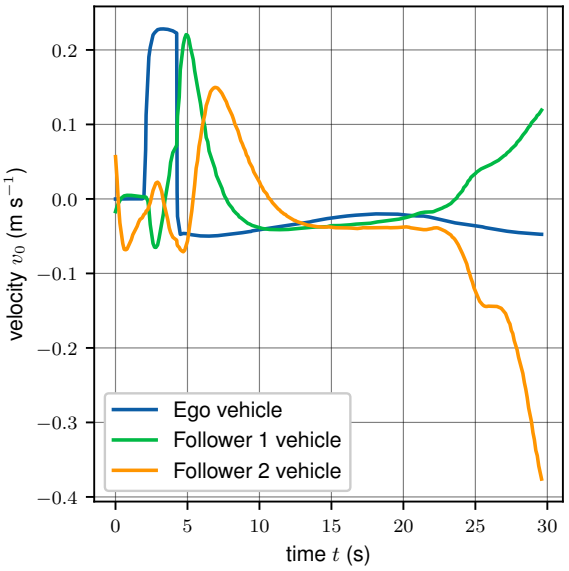
However, to ensure integrity and convergence the NMPC optimisation problem had to be re-tuned with higher weight values. While matrix $\mathbf{R}$ remained unaltered, matrix $\mathbf{Q}$ had to be assigned higher weights on the segments' orientation, such that

$$\mathbf{Q} = \operatorname{diag}(100, 150, 10, 5, 10, \underbrace{10, 10, \ldots, 10}_{n \text{ times}}) \tag{5.8}$$

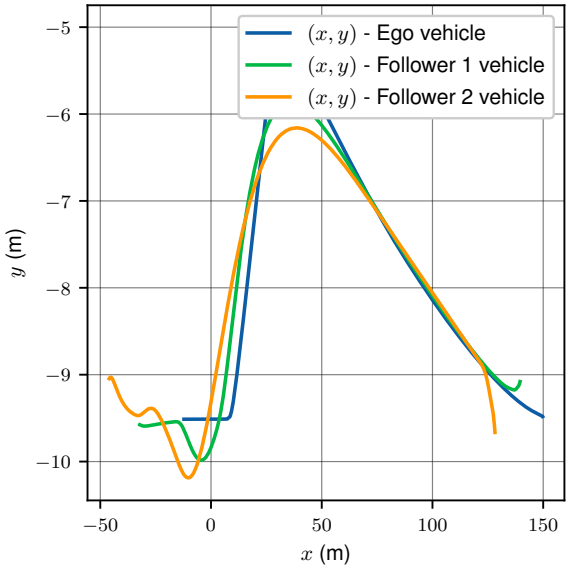
This adjustment had to be applied the segments orientation otherwise it would diverge from its desired orientation, ensuring the MPC's effectiveness in achieving the desired outcomes. The predicted horizon remained the same, i.e. $N = 10$.



(a) Time evolution the platoon's position.



(b) Velocity profile.



(c) Position tracking.

Figure 5.9: Performance metrics for a dynamic obstacle avoidance scenario with a three vehicles platoon configuration.

Figure 5.11 along with the additional snapshots provided in the Appendix C offer a comprehensive view of the real-time obstacle avoidance performance.

In order to evaluate the real-time performance of the proposed approach, we conducted a comparison with an established baseline model. The objective was to assess the efficiency and effectiveness of our approach in obstacle avoidance scenarios. The results, presented in Table 5.1, highlight key performance metrics, including computational time and the number of iterations required for successful manoeuvre execution.

Note that the time indicated in the table encompasses the entire set of iterations specified.

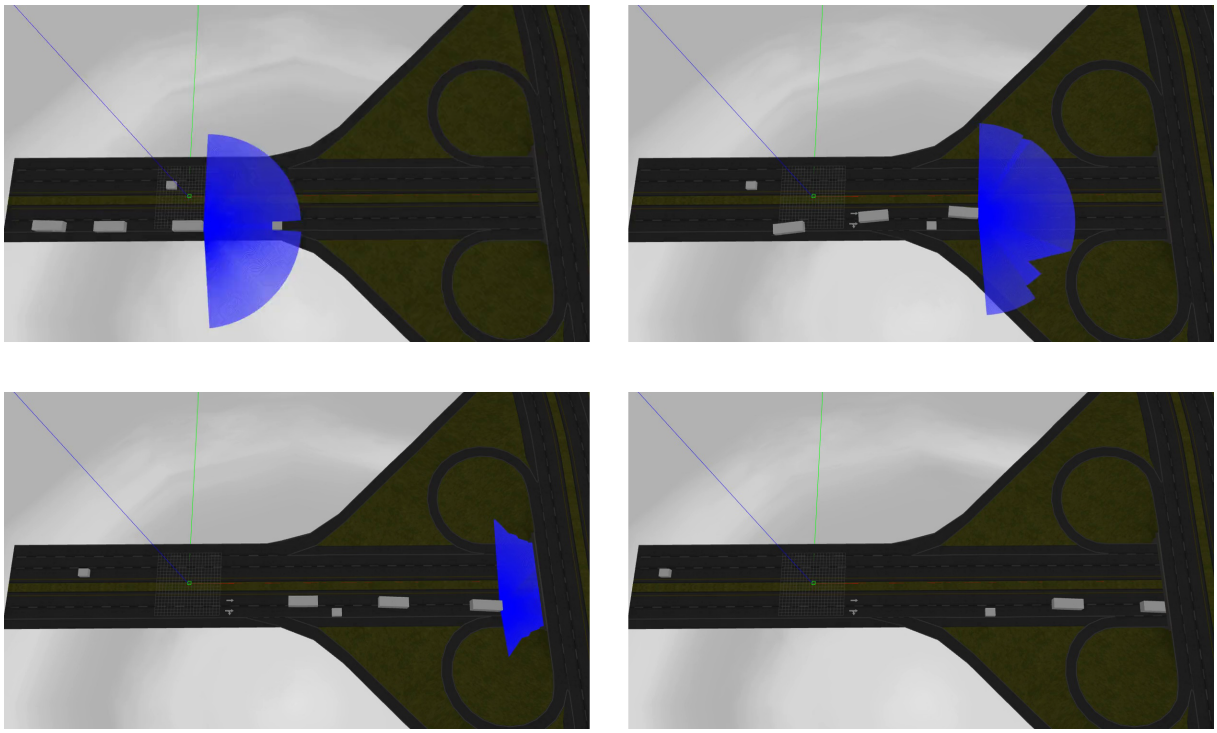


Figure 5.10: Snapshot from a 16-second video clip showcasing a platoon of vehicles navigating past a static obstacle. In this frame, the ego vehicle, equipped with LiDAR, leads a platoon of two following vehicles as they execute a coordinated lane change. The dynamic obstacle being overtaken is a*light-grey* cube on the rightmost lane of the highway environment that is independently controlled through a PID.

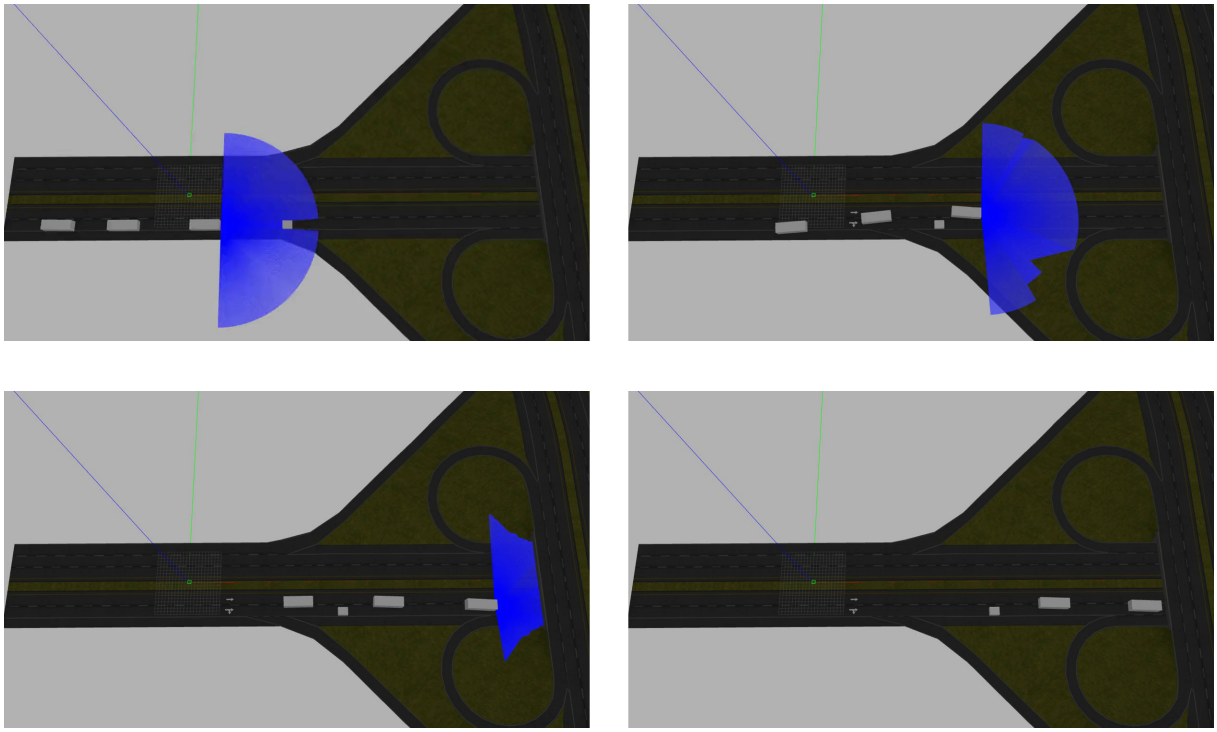


Figure 5.11: Snapshot from a 14-second video clip showcasing platoon of vehicles navigating past a dynamic obstacle. Here, the baseline NMPC controller was employed.

These results underscore the significance of real-time performance in obstacle avoidance scenarios. Both the CBFs method and the Penalty method demonstrate commendable efficiency, with comparable

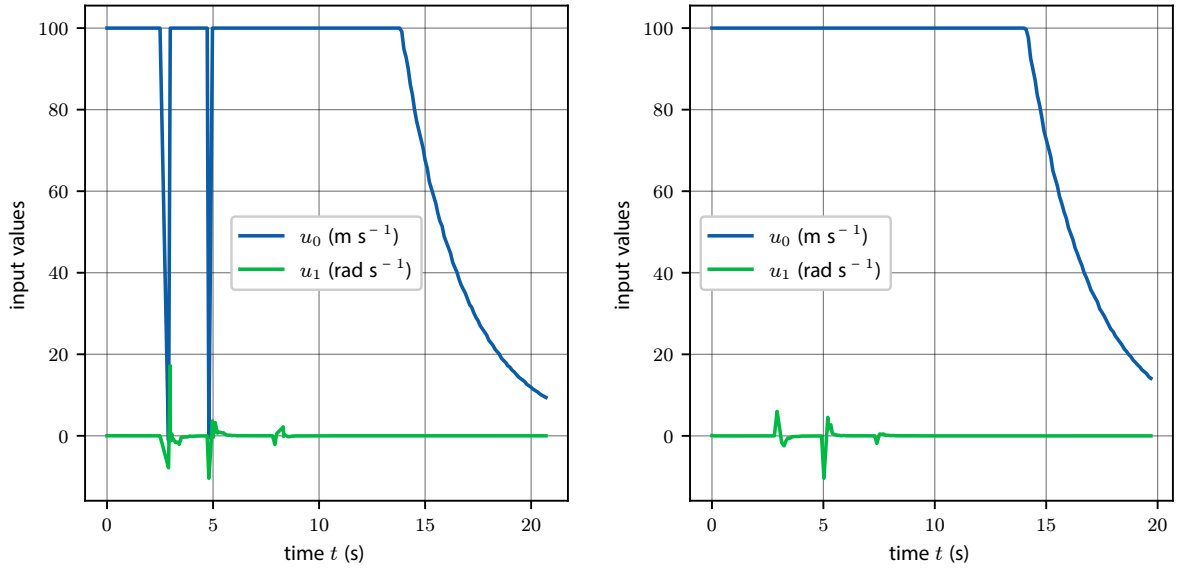
Table 5.1: Comparison of the computational time, in seconds, and iteration count between the CBFs method and the Penalty method. These metrics offer insights into the real-time capabilities of both approaches in overcoming obstacles.

	CBFs method	**Penalty method**
Computational time (sec)	2.1384	2.0098
Iterations	11	12

computational times. However, the Penalty method requires slightly more iterations to complete the manoeuvre.

In the context of real-time applications, where rapid decision-making is paramount, minimising the computational time and maintaining a reasonable iteration count are crucial factors. The small difference in iteration count suggests that both methods offer feasible solutions within practical timeframes. However, the faster computational time of the Penalty method may provide an edge in scenarios where split-second decisions are required for obstacle avoidance.

In our comparative analysis, we explore the magnitude of input signals, offering insights into the control strategies employed by both the baseline and the novel proposed methods. Figure Figure 5.12 provides a side-by-side comparison of the input signal magnitudes, shedding light on their respective characteristics and performance.



(a) Baseline method.

(b) Novel method.

Figure 5.12: Comparative analysis of the input signals magnitude for the baseline methods vs. the novel proposed method.

In the presented comparative analysis notable differences between the novel method and the baseline approach emerge, offering valuable insights into their control strategies. Firstly, when examining acceleration, a distinct characteristic stands out. In the novel method, acceleration exhibits a linear behaviour at the beginning of the ego vehicle's movement. Conversely, in the baseline approach, acceleration shows two steep valleys. These valleys are a direct result of the simulation's behaviour,

where the platoon briefly halts to evaluate and compute its trajectory. This momentary stoppage creates the observed acceleration pattern. On the other hand, when assessing the steering rate, both experiments showcase striking similarities. This similarity suggests that the steering rate profiles in both methods are relatively consistent, emphasising their stable control behaviour during the obstacle avoidance manoeuvre.

Furthermore, it's worth noting that the nonlinearity in the optimisation process primarily arises from the CBF constraints. These constraints introduce complexity to the optimisation process, as they dictate the vehicle's behaviour to ensure safety. Despite this increase in complexity, it's crucial to highlight that the reduction in constraint horizons, achieved through careful algorithm design, tends to dominate any rise in complexity resulting from the introduction of additional optimisation variables ω_k. This optimisation strategy underscores the effectiveness of both methods in navigating complex obstacle avoidance scenarios while maintaining platoon coherence and safety.

5.2.3 Abrupt Braking of the Ego Vehicle

In the last experiment, we tested the response of the platoon's integrity to the abrupt braking of the ego vehicle. The platoon tested consisted of four trucks: one ego vehicle and three followers. Similar to previous experiments, all vehicles in this scenario had a similar tractor-trailer configuration.

The results in Table 5.2 provide insight into the response time of each following vehicle in the platoon to the ego vehicle's abrupt braking. The table records the time, in seconds, taken by each follower to detect and respond to the ego vehicle coming to a stop. Remarkably, the recorded times indicate almost instantaneous responses. The responses, captured in fractions of a second, demonstrate the platoon's

Table 5.2: Response times, in seconds, of following vehicles to ego vehicle's abrupt braking.

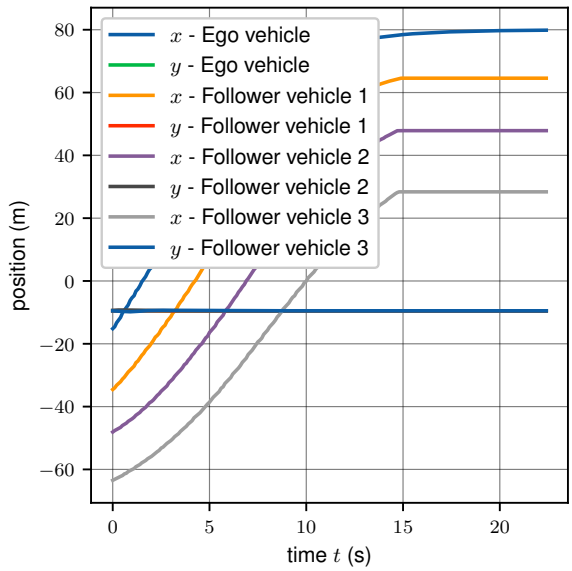
Follower 1	Follower 2	Follower 3
1.846e-05	1.692e-05	2.885e-05

rapid adaptability to sudden changes in the ego vehicle's motion.

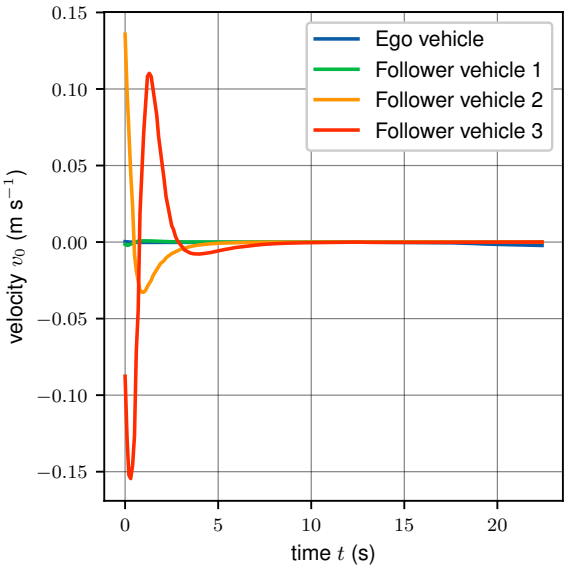
Additionally, Figure 5.13 presents a comprehensive set of performance metrics for this scenario. The images include the time evolution of vehicle positions (Figure 5.13a) and velocities (Figure 5.13b), as well as the tracked positions (Figure 5.13c) during the experiments.

The parallel evolution of the x-coordinate in Figure 5.13a indicates that the safety distance constraint is consistently maintained throughout the movement. In 5.13b, we observe that all vehicles in the platoon come to a halt at approximately the same time, around second 7. At this point, their velocities converge to zero. And finally, Figure 5.13c illustrates that the vehicles come to a complete stop, with distances between them remaining constant. This experiment reinforces the platoon's capability to respond swiftly to abrupt changes in the ego vehicle's motion, showcasing its potential in critical real-world situations.

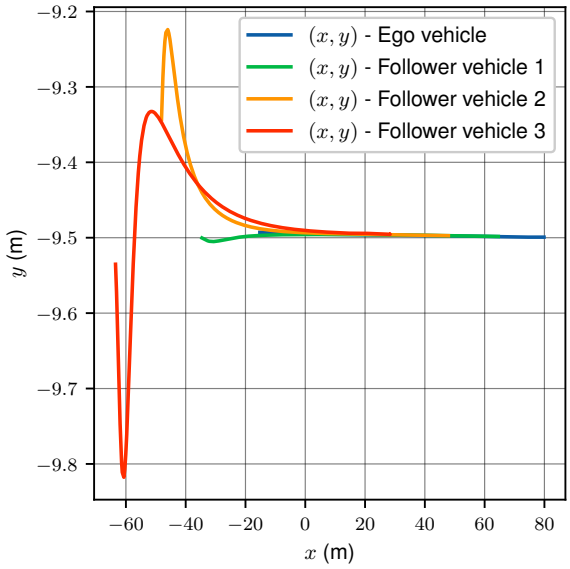
Figure 5.14 presents a snapshot from a 12-second video clip, providing a visual representation of the platoon's coordinated braking manoeuvre.



(a) Time evolution the platoon's position.



(b) Velocity profile.



(c) Position tracking.

Figure 5.13: Performance metrics for a abrupt braking of the ego vehicle scenario with platoon of three vehicles.

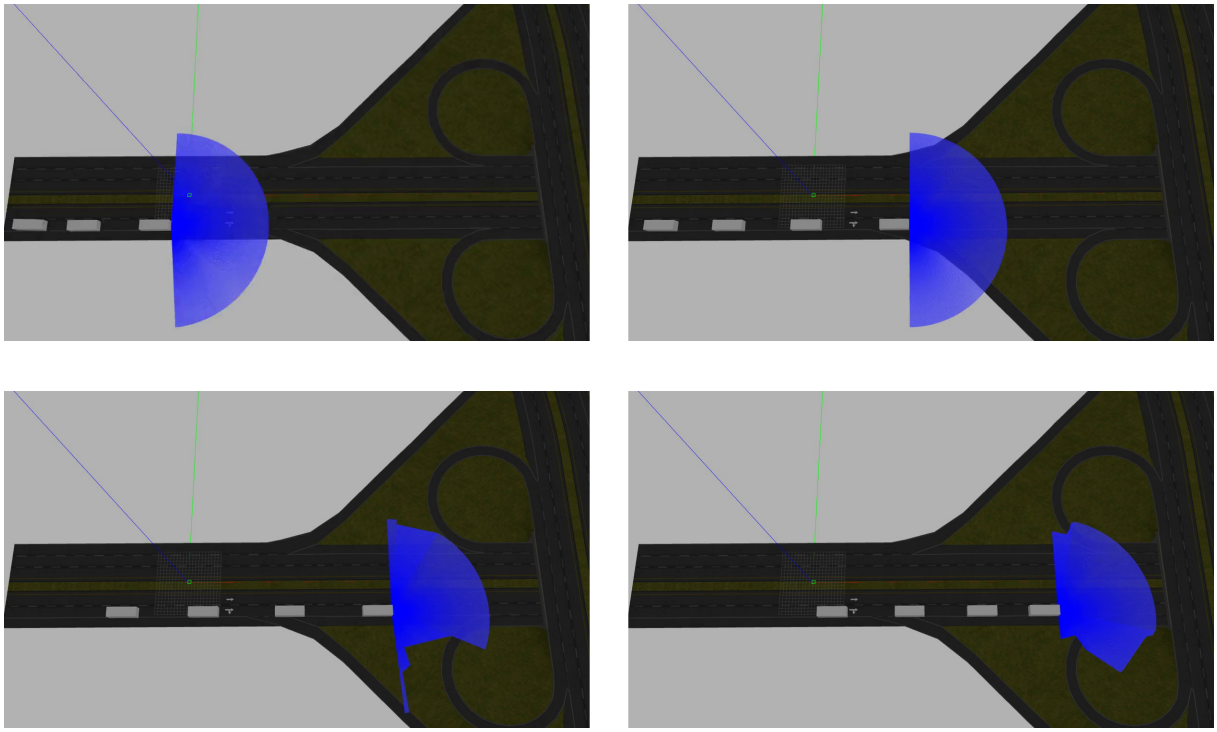


Figure 5.14: Snapshot from a 12-second video clip showcasing platoon of vehicles. In this frame, the ego vehicle, equipped with LiDAR, leads a platoon of two following vehicles. In this experience the ego vehicle is disconnected from its controller leading to an abrupt brake.

Chapter 6

Discussion

This chapter delves into a comprehensive discussion of the research findings and insights gained throughout this study. The preceding chapters have focused on the development and implementation of a novel control algorithm, leveraging NMPC to enable autonomous platooning. While these chapters have presented the technical details and results of the proposed approach, the discussion segment provides a platform for a deeper analysis of these findings and their implications.

6.1 Control of Nonlinear Systems

One of the fundamental challenges encountered in this research was the optimal control of highly nonlinear systems, particularly in the context of a general n-trailer model. Nonlinearity inherently introduces complexities that demand innovative solutions. Throughout the development of the NMPC framework, numerous decisions were made to address these challenges effectively. The results highlight the algorithm's adaptability and ability to tackle the inherent nonlinearities, reinforcing its suitability for real-world platoon control applications.

The choice of discretisation techniques plays a pivotal role in controlling nonlinear systems. In our approach, the RK4 was chosen over the simpler Euler method. This decision was driven by the intricate dynamics of the general n-trailer model, characterised by its nonlinear behaviour. RK4's higher-order accuracy proved essential in capturing the system's nuances and enhancing predictive accuracy. The adoption of RK4 contributed significantly to improved real-time performance and manoeuvre execution in the presence of nonlinearities.

One of the key innovations in the NMPC approach is the incorporation of soft constraints within the objective function. Unlike traditional methods that apply nonlinear distance constraints directly in the optimisation problem, soft constraints were implemented, representing a linear operation. This strategic shift yielded notable advantages, including improved computational efficiency and manoeuvre stability. By introducing soft constraints, the NMPC framework strikes a balance between obstacle avoidance and platoon coherence, offering a practical solution for real-time platoon control.

In conclusion, this discussion underscores the pivotal role of NMPC in addressing the challenges of

controlling nonlinear systems. The decisions made throughout the development process, from handling nonlinearities to selecting appropriate discretisation techniques and implementing soft constraints, collectively contribute to the algorithm's adaptability and robustness in real-world platoon control scenarios. These insights mark significant progress in the realm of autonomous truck platooning, bringing us closer to realising safer, more efficient, and intelligent transportation systems.

6.2 Real-World Applications

The practical applications of autonomous platooning extend beyond the realms of research and into real-world scenarios with transformative potential. This section explores some key domains where the proposed algorithm holds promise.

6.2.1 Overtaking

Overtaking other vehicles on the road is a common manoeuvre that requires careful coordination and control to ensure safety and efficiency. From the experiment of "Dynamical Obstacle Avoidance" where the baseline method was evaluated in respect to the novel method, one notable advantage of our proposed approach is the ability to execute overtaking manoeuvre with smoother velocity profiles compared to traditional methods, as demonstrated in the benchmarks.

In many benchmark scenarios, abrupt changes in velocity are often observed, particularly when the platoon encounters obstacles or navigates through complex environments. These sudden velocity changes, while effective in obstacle avoidance, may not be ideal for overtaking scenarios.

Smooth, gradual changes in velocity during overtaking can provide several benefits:

1. Enhanced Safety: Smoother velocity profiles reduce the risk of abrupt changes that might catch other road users off guard. This enhances overall road safety, especially when overtaking slower-moving vehicles.

2. Passenger and Load Comfort: Sudden changes in velocity can be uncomfortable for passengers and cargo, particularly in commercial transportation. Smoother overtaking manoeuvres improve the comfort of those on board.

3. Fuel Efficiency: Gradual changes in velocity can contribute to improved fuel efficiency. Abrupt changes often result in unnecessary fuel consumption, while smoother manoeuvres optimise energy usage.

4. Traffic Flow: Smoother overtaking can lead to improved traffic flow and reduced congestion. Predictable, gradual manoeuvres allow for better coordination among vehicles on the road.

By prorating smoother velocity profiles during overtaking, our proposed approach aligns with the broader goals of enhancing road safety, passenger comfort, and fuel efficiency. It also contributes to a more harmonious traffic flow, benefiting both platooning and other road users.

Incorporating such considerations into platoon control strategies can pave the way for safer and more

efficient overtaking manoeuvres, ultimately contributing to the broader adoption of platooning in real-world scenarios.

6.2.2 Emergency Braking

In the context of real-world applications, our proposed approach opens up new possibilities and challenges for platoon management, particularly in emergency situations like abrupt braking. Traditionally, platooning research has often focused on maintaining relatively large inter-vehicle distances, typically between 5 to 10 meters, to mitigate the risk of rear-end collisions, especially during emergency braking scenarios. The SARTRE project [**bergenhem_challenges_nodate**], for instance, conducted controlled experiments with professional truck drivers in lead vehicles and observed inter-vehicle distances in this range, broadcasting packets to all trailing vehicles. Additionally, they emphasised disbanding the platoon in the event of braking emergencies as a safety measure.

However, our approach introduces a unique perspective by reducing the inter-vehicle distance to 2.5 meters. This reduced distance not only provides aerodynamic benefits but also results in increased average fuel savings for the entire platoon. Nonetheless, at such short following distances, the risk of rear-end collisions during abrupt braking is still present, albeit mitigated by the platoon's automated control system. This approach allows the platoon to react swiftly to emergencies and stop within a driver's typical response time, akin to real-life scenarios involving imminent collisions, sudden obstacles, or unexpected slowdowns. This capability is crucial for collision-free emergency braking in practical applications.

The "Abrupt Braking of the Ego Vehicle" experiment, as discussed in Chapter 5, showcased the platoon's ability to halt with a driver's response time. This capability is vital in real-life situations such as imminent collisions, sudden obstacles, or unexpected slowdowns.

In [**noauthor_active_nodate**] was obtained empirically that the reaction time for a human driver can vary from 0.3 s to 1.2 s. And while the simulations naturally do not account for latency in real-world communication systems, the demonstrated capabilities highlight the potential of our approach in improving road transportation efficiency and safety.

Chapter 7

Conclusion

In this thesis, we have delved into the realm of platoon control systems, focusing on the development and evaluation of a novel approach for real-time obstacle avoidance. Platoon-based transportation systems have garnered significant attention for their potential to revolutionise road transport by improving fuel efficiency, reducing emissions, and enhancing overall traffic management. However, the successful implementation of platoon systems hinges on the ability to navigate complex environments while ensuring safety, feasibility and efficiency.

Foremost, a comprehensive exploration of the state-of-the-art in platoon control is introduced while revealing existing challenges and opportunities for improvement. We identified the need for more robust obstacle avoidance strategies and set out to design a novel solution that addresses these challenges.

The proposed approach to real-time obstacle avoidance leverages NMPC to enable platoons to respond to dynamic obstacles in their path. This approach stands out for its ability to seamlessly integrate with existing platoon control systems and adapt to rapidly changing road conditions.

One of the cornerstones of our approach is the use of penalty soft constraints on the optimisation problem to define safety constraints, ensuring that vehicles within the platoon maintain safe distances from obstacles while preserving platoon coherence. We meticulously designed an objective function, demonstrating their effectiveness in preventing collisions in real-time simulations. Our predictive controller optimises trajectory planning, enabling platoons to navigate around obstacles while ensuring minimal disruption to traffic flow.

To rigorously evaluate the effectiveness of our proposed approach, we conducted a series of experiments in simulation environments that replicated real-world scenarios. These experiments encompassed a wide range of challenges, from static obstacle avoidance to abrupt braking scenarios. Through detailed analysis and comparison with established baseline models, we demonstrated the superior performance of our approach in terms of safety, efficiency, and real-time decision-making.

This chapter provides a summary of the contributions then discusses future research directions.

7.1 Key Findings and Contributions

Our research yielded several key findings and contributions:

1. Enhanced Safety: Our approach significantly enhances the safety of platoon-based transportation systems by effectively preventing collisions with obstacles. This is achieved through the predictive control, and real-time decision-making.

2. Improved Efficiency: We have shown that platoons equipped with our approach can efficiently navigate complex environments while maintaining platoon coherence. This efficiency contributes to reduced fuel consumption and emissions, aligning with sustainability goals.

3. Real-Time Adaptability: Our approach demonstrates remarkable real-time adaptability, capable of responding swiftly to dynamic obstacles and unforeseen challenges. This adaptability positions platooning as a viable solution for contemporary transportation needs.

4. Smoother Velocity Profiles: In overtaking scenarios, our approach excels in providing smoother velocity profiles compared to traditional methods. This enhances overall road safety, passenger comfort, and fuel efficiency.

7.2 Future Work

While the present research represents a significant step forward in platoon control systems, there are several avenues for future exploration and improvement. Continued research and development should focus on scalability, integration with autonomous vehicles, and extensive real-world testing. The following discussions outline these potential areas of exploration and improvement:

1. Cost-Coupled Optimisation Problem: this would involve each agent, including the ego vehicle and followers, autonomously computing and possessing knowledge of its individual cost functions, which could encompass various metrics such as fuel consumption, traffic flow optimisation, and safety considerations - likewise the current approach. By exchanging cost-related information among neighbouring agents through a novel algorithm, the cooperative optimisation problem aims to collectively minimise the overall cost of the platoon, resulting in a more efficient and collaborative system. Further investigation would be needed to fully understand the potential benefits of this approach, optimise its implementation and compare its competitive performance with the current method.

2. Enhanced Perception with 3D LiDAR Integration: A promising avenue for future work involves the integration of 3D LiDAR sensors into the lead truck of the platoon. While our experiments have already showcased the effectiveness of LiDAR in obstacle detection, the transition to 3D LiDAR can provide a more comprehensive view of the environment, enabling better obstacle recognition and avoidance strategies.

3. Advanced Communication Protocols: Building on our insights into long-distance communication challenges between follower vehicles and the ego vehicle, future research can focus on the development of more advanced communication protocols. These protocols should not only

enhance data transmission reliability but also support real-time decision-making and coordination among platoon members.

4. Fine-Tuning of MPC Parameters: While our experiments have demonstrated the feasibility and efficiency of our proposed approach, there is still room for extensive tuning of the MPC parameters. Future work can delve deeper into parameter optimisation to further refine the control strategies, ultimately leading to even smoother and more efficient platoon operations.
5. Graphics Processing Unit (GPU) Integration for Real-Time Control: To augment the real-time control capabilities of platoon systems, future research can explore the integration of GPUs. Leveraging GPU technology can significantly improve tracking performance, enabling real-time obstacle detection and avoidance. This advancement can further enhance the safety and adaptability of platoon-based transportation systems.

Additionally, further investigation into the broader implications of platoon-based transportation on traffic management, energy efficiency, and urban planning is warrented. Collaborative efforts involving academia, industry stakeholders, policymakers, and urban planners will be crucial in realising the full potential of platoon-based transportation systems. The future of transportation holds immense promise, and continued innovation and research will play a pivotal role in shaping that future.

Appendix A

Explanation of the 4th Order Runge-Kutta Method

In this appendix, we provide an explanation of the 4th Order Runge-Kutta (RK4) method, which is a numerical integration technique widely used to solve Ordinary Differential Equations (ODE).

The RK4 method is an iterative process that approximates the solution of an ODE by evaluating the derivative at multiple points within a time interval and combining them to obtain an accurate estimate of the solution at the next time step.

The general formula for the RK4 method is as follows:

$$
\begin{aligned}
k_1 &= h \cdot f(t_n, y_n), \\
k_2 &= h \cdot f(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}), \\
k_3 &= h \cdot f(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}), \\
k_4 &= h \cdot f(t_n + h, y_n + k_3), \\
y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),
\end{aligned}
$$

where:

- t_n represents the current time,
- y_n is the current approximation of the solution at time t_n,
- h is the step size, which determines the time increment between each iteration,
- $f(t, y)$ is the function representing the derivative of the solution.

Appendix B

Khachiyan's algorithm for Minimum-Volume Enclosing Ellipsoids

Khachiyan's algorithm is an iterative algorithm that computes the minimum-volume enclosing ellipsoid (MVEE) of a given set of points in high-dimensional space. The algorithm starts with an initial ellipsoid and iteratively refines it until convergence. The key idea is to iteratively update the centre and shape of the ellipsoid to minimise its volume while still containing all the given points.

The algorithm can be summarised as follows:

1. Given a set of n points $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ in d-dimensional space.
2. Initialise an ellipsoid $\mathcal{E}_0$ with centre $\mathbf{c}_0$ and shape matrix $\mathbf{A}_0$.
3. Repeat until convergence:
   (a) For each point $\mathbf{x}_i$, compute its Mahalanobis distance d_i from the current ellipsoid $\mathcal{E}_t$.
   (b) Update the centre of the ellipsoid as the weighted average of the points:
   $$\mathbf{c}_{t+1} = \frac{\sum_{i=1}^{n} w_i \mathbf{x}_i}{\sum_{i=1}^{n} w_i}$$
   where $w_i = \frac{1}{\sqrt{1+d_i^2}}$.
   (c) Update the shape matrix of the ellipsoid as:
   $$\mathbf{A}_{t+1} = \frac{\sum_{i=1}^{n} w_i \left(\mathbf{x}_i - \mathbf{c}_{t+1}\right)\left(\mathbf{x}_i - \mathbf{c}_{t+1}\right)^\top}{\sum_{i=1}^{n} w_i}$$
4. Output the final ellipsoid $\mathcal{E}_t$ as the minimum-volume enclosing ellipsoid.

Appendix C

Simulation Results

This appendix section provides additional snapshots based on the same experimental scenarios covered in the main body of the dissertation, which include overtaking scenarios, with both static and dynamic obstacles, respectively Figure C.1 and Figure C.2, and lastly Figure C.3 from the baseline comparison performed.

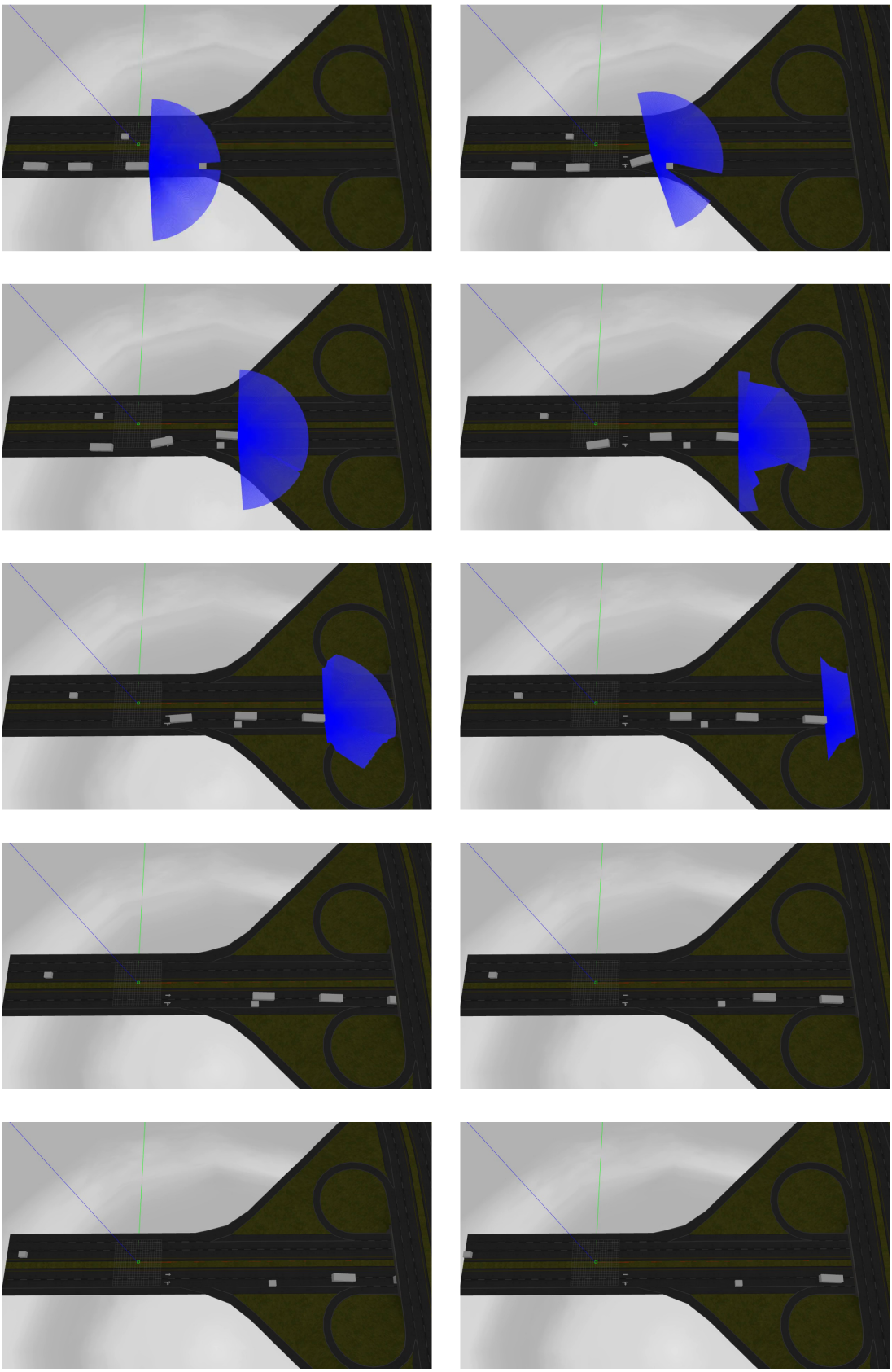


Figure C.1: In-depth snapshots of Figure 5.8.

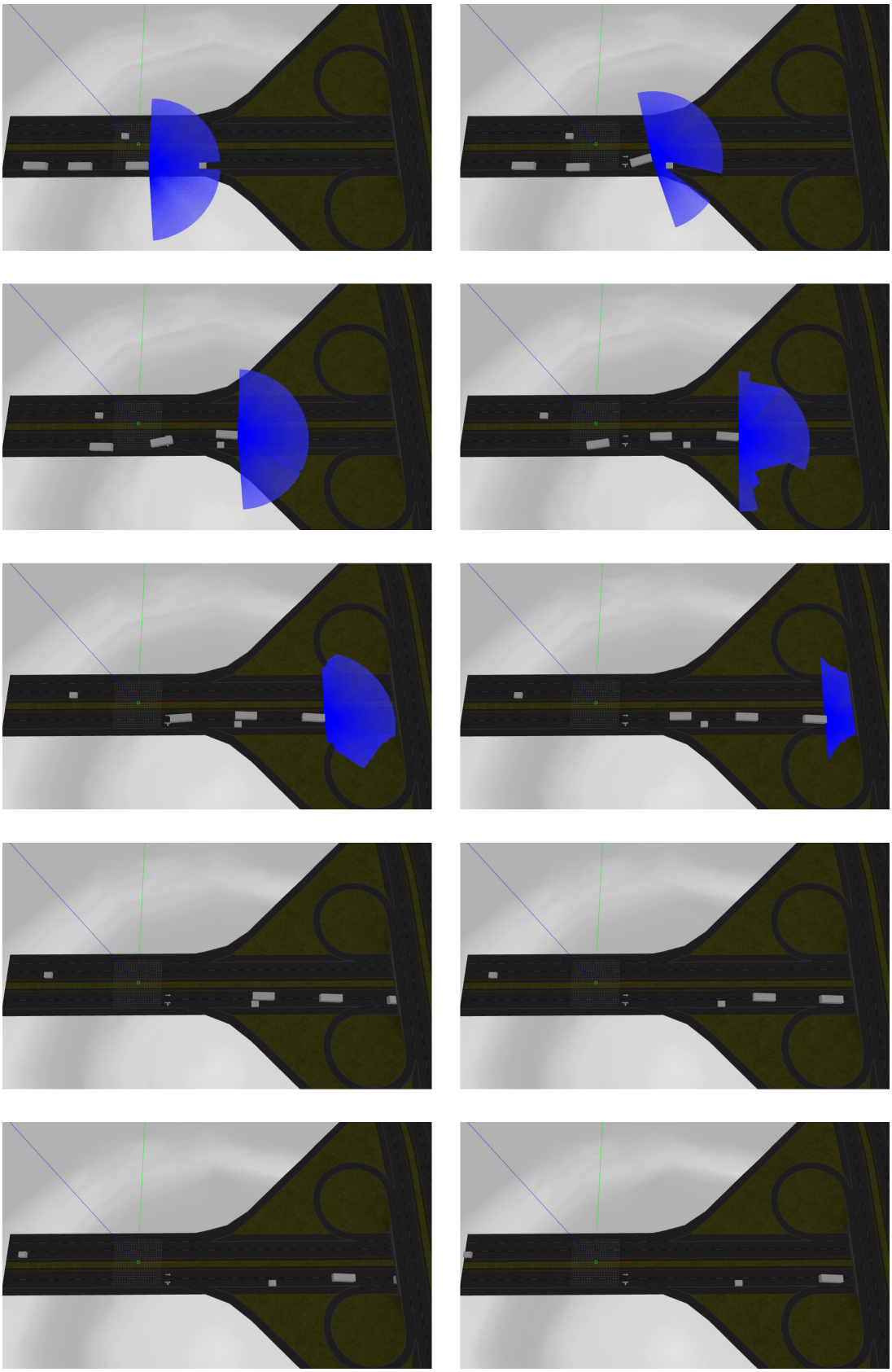


Figure C.2: In-depth snapshots of Figure 5.10.

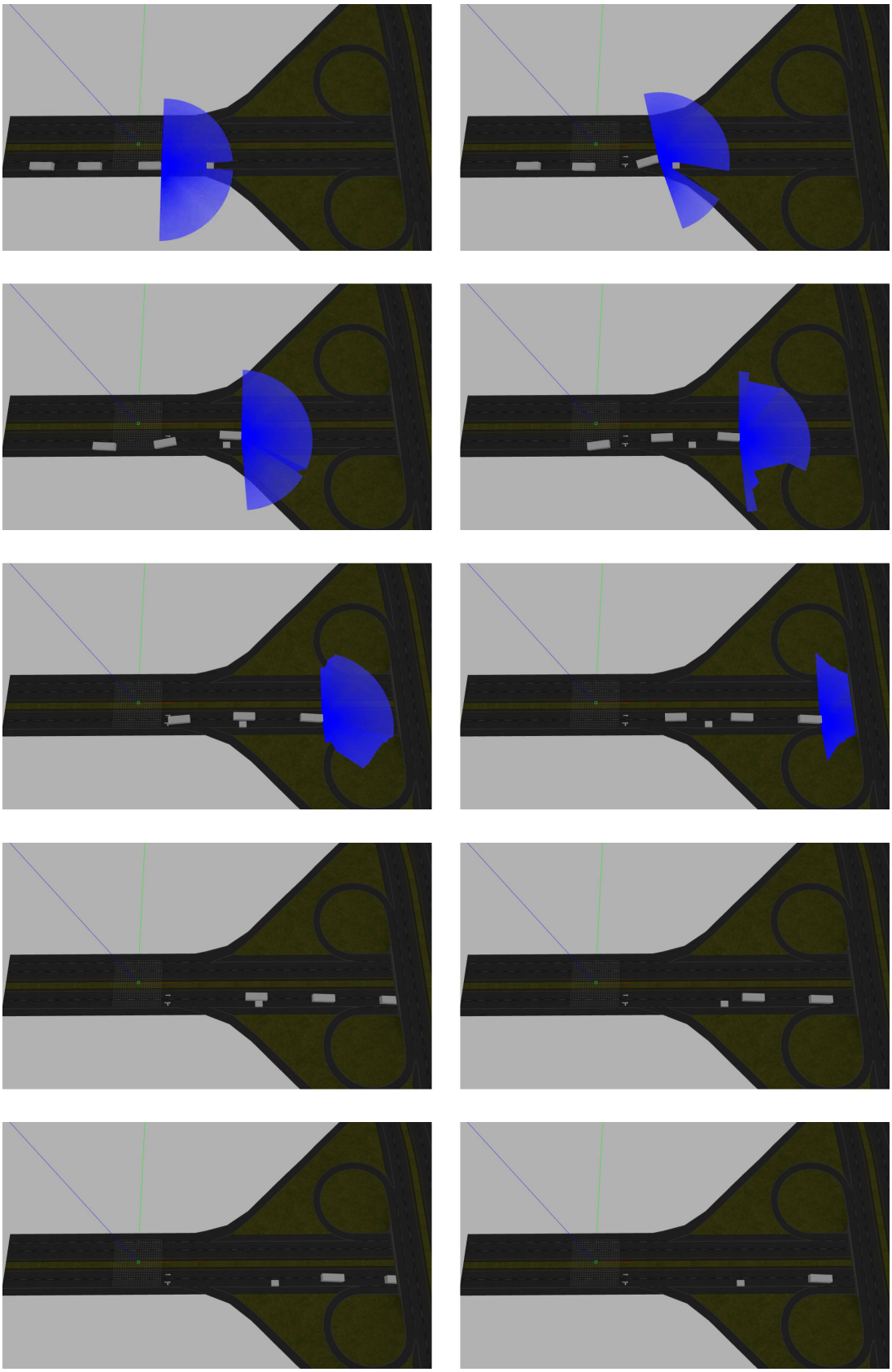


Figure C.3: In-depth snapshots of Figure 5.11.