EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mechanical Engineering
Dynamics and Control Section

Online Learning for Interaction-Aware Motion Planning with Gaussian Process Model Predictive Control

Master's Thesis

By

ing. Tren Martinus Johannes Theodor Baltussen

Program: Systems and Control
Report Number: DC 2024.014
Academic Credits: 45 ECTS
Student ID: 1632345

Committee Members:
dr. ir. A.A.J. Lefeber
dr. ir. A. Katriniok
prof. dr. ir. W.P.M.H. Heemels
dr. ir. R. Tóth

Eindhoven, February 7, 2024

The research described in this MSc thesis was done in accordance with the TU/e Code of Scientific Conduct.

To Lian

Acknowledgements

This MSc thesis presents my research on online learning for interaction-aware motion planning for autonomous vehicles. This thesis is the result of a challenging, interesting, and very fun final stage of my master's studies. Firstly, I would like to thank the committee members, dr. ir. Roland Tóth and prof. dr. ir. Maurice Heemels, for their feedback and for participating in my MSc thesis defense. In particular, I would like to thank dr. ir. Roland Tóth for his valuable discussions on Gaussian processes. I also want to say thanks to my dear friends and fellow students who have supported me during my time at TU/e. Finally, I would also like to express my gratitude toward dr. ir. Erjen Lefeber and dr. ir. Alexander Katriniok. Through your guidance and support, I have developed myself in a lot of ways, both technically and personally, and I feel more than ready for what is coming next.

Thank you.

Abstract

Autonomous vehicles have the potential to increase the safety, efficiency, and availability of transportation systems. To reach this potential, significant developments in the fields of vehicle design and control are essential. Safe navigation in complex traffic scenarios requires awareness and careful consideration of interactions between different road users. However, coping with uncertain or unseen behavior in a traffic environment poses a considerable challenge that requires further research. To this end, Gaussian processes have been developed as strong function regressors that can be used to jointly predict the motion of other vehicles while considering interactions as well as uncertainty. Furthermore, learning-based model predictive control has had recent success in the motion planning of systems subject to uncertainty. This MSc thesis aims to extend the methods and capabilities of online learning-based interaction-aware model predictive control using Gaussian process prediction models for uncertain traffic scenarios. Simulation results show that online learning-based Gaussian process model predictive control is able to passively, and actively, learn the interactions between road vehicles without the need for pre-training, demonstrating its generalizability outside of training sets. The passive and active learning-based Gaussian process model predictive controllers are compared against a baseline controller in a series of simulation studies. Our interaction-aware motion planner shows improved prediction quality and demonstrates the potential of Gaussian process model predictive control and opens new doors for motion planning in uncertain and unseen traffic scenarios.

Contents

Acknowledgements **iii**

Abstract **v**

Contents **vii**

1 Introduction **1**
1.1 Interaction-Aware Motion Planning 1
1.2 Learning-based Model Predictive Control 4
1.3 Interactive Planning with Gaussian Processes 5
1.4 Related Work 7
1.5 Contributions 9

2 The Motion Planning Problem **11**
2.1 Lane Merging Scenario 11
2.2 Vehicle Modeling 12
2.3 Optimal Control Problem 23

3 Gaussian Process Prediction Model **31**
3.1 Preliminaries 31
3.2 Learned Dynamics 36
3.3 Uncertainty Propagation 39
3.4 Gaussian Process MPC 44

4 Learning-based GP-MPC **45**
4.1 MPC with a Constant Velocity Model 45
4.2 Passive Learning with GP-MPC 47
4.3 Active Learning with GP-MPC 50
4.4 Numerical Optimization 56

5 Results **59**
5.1 Experiment Design 59
5.2 Baseline MPC 62

5.3 Passive Learning with GP-MPC 65
5.4 Active Learning with GP-MPC 73
5.5 Generalizability 85
5.6 Discussion 93
5.7 Reflection 95

6 Conclusions and Recommendations **97**
6.1 Conclusions 97
6.2 Recommendations 99

References **103**

A Collision Avoidance Ellipse **109**

B Results **111**
B.1 Results of Constant Velocity MPC 112
B.2 Results of Passive Learning SPGP-MPC 113
B.3 Results of Active Learning SPGP-MPC 115
B.4 Results on Various Initial Conditions 117
B.5 Results on Altruistic Driving Behavior 118
B.6 Results on Adjust Slack Penalty Weights 119

Chapter 1

Introduction

Intelligent and autonomous vehicles have the potential to increase the safety, efficiency, and availability of transportation systems and transform transportation into a utility that is available to anyone, at any time. To realize this potential, advancements in many functionalities of vehicle autonomy are required, ranging from environment perception to coordination, motion planning, control, and human interaction. Autonomous vehicles (AVs) operate in complex dynamic environments and are faced with uncertain situations or situations that have not been encountered before. Hence, AVs require methods that generalize well to such situations before we can reach (beyond) human-level reliability and safe operation, even in complex situations [1]. In order to cope with uncertain or even unseen behavior of other road users, the research presented in this thesis aims to extend the generalizability of motion planners through online learning Gaussian process model predictive control.

In the next section, we provide an overview of the state of the art in autonomous driving methods, with a focus on decision-making and motion planning. Subsequently, in section 1.2 and section 1.3, we explore promising methods to advance motion planning for complex environments with uncertain and unseen behavior. In section 1.4, we discuss related works and identify a research gap. Finally, section 1.5 states the research objectives of this thesis and a list of contributions that address this research gap and concludes with an outline of the thesis.

1.1 Interaction-Aware Motion Planning

1.1.1 Autonomous Driving Architecture

Autonomous vehicles rely on the aforementioned functionalities, such as decision-making and planning. These functionalities are decomposed in a control architecture which receives sensory input about the AV's environment and determines control outputs that control the AV to its desired behavior. Such an architecture is also known as an autonomy stack. Schwarting *et al.* [1] distinguish three types of control architectures, detailed in Figure 1.1.

Traditional architectures (top of Figure 1.1) employ a sequential approach that has

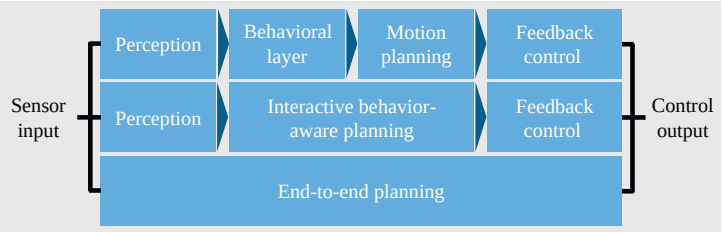

Figure 1.1: Typical control architectures for autonomous vehicles, adopted from [1].

clear interfaces between separate modules. However, traffic scenarios are characterized by complex interactions and require more advanced and integrated methods [1]. Conversely, *end-to-end planning* (bottom of Figure 1.1) integrates perception, planning as well as control [1]. End-to-end planning primarily relies on deep-learning methods to map sensor inputs directly to outputs that control the AV. While end-to-end planning is promising, the lack of hard-coded safety measures and interpretability are its biggest shortcomings [2]. As both traditional architectures and end-to-end planning have some inherent challenges, we focus on *interactive behavior-aware planning* (middle of Figure 1.1) which integrates decision-making and planning. The perception module provides a world model of the AV's environment, while the interactive behavior-aware planning module determines a suitable trajectory for low-level vehicle control. As such, behavior-aware motion planning considers behavioral aspects of other traffic participants while planning its motion and accounts for complex interactions that naturally occur in real-life traffic scenarios [1]. Interactive behavior-aware motion planning has the potential to consider complex interactions for challenging traffic scenarios with better interpretability than that of end-to-end planning. Therefore, we look into these types of motion planning methods in more detail in the next sections.

1.1.2 Interaction-Aware Motion Planning

Motion planning is a task concerned with determining an appropriate trajectory for the AV through space and time, which achieves some objective, e.g., changing lanes, or crossing an intersection, while minimizing the risk of collision and the violation of traffic rules. Socially compliant driving relies on cooperation and interactivity between vehicles and is vital for safe motion planning in cluttered, dynamic, and uncertain environments [1]. When an AV knowingly and willingly improves the combined performance of all traffic participants, we call this behavior cooperative [3]. Interactivity can be described as the interdependence of an agent's actions on other agents' actions. Cooperative and interactive decision-making is essential to achieve human-like driving behavior in AVs. It is essential that AVs are able to deduce the intentions of other road users without the need for inter-vehicle communication to successfully integrate them in traffic [1].

A survey on motion prediction and risk assessment for intelligent vehicles [4] classifies traffic motion prediction into three categories: physics-based, maneuver-based, and interaction-aware. Note that interaction- and behavior-aware planning are interchangeably

used in the literature. Physics-based motion models only rely on low-level physical behavior of the AV and are therefore only suitable for short-term motion prediction (less than one second). Maneuver-based models assume that a vehicle's motion consists of a series of maneuvers that are independent from those of other road users, and are either based on prototype trajectories or on maneuver intention estimation. Interaction-aware models, although more complex, consider the interdependence between vehicles and provide a more reliable evaluation of the risk associated with a certain motion plan [4]. As interaction-aware planning remains an open challenge [1] and an active field of research, this thesis focuses on advancing methods for interactive planning and we confine ourselves to the interactive behavior-aware planning module (Figure 1.1).

1.1.3 Iterative Planners

In motion planning, we can typically distinguish two styles of planners: sampling-based planners and iterative planners. On the one hand, sampling-based planners take a large set of candidate solutions and test their performance. Moreover, unlike iterative planners, the evaluation of sampling-based planners can be parallelized. An iterative planner, on the other hand, iteratively refines the motion plan, for example, using a gradient, or Bayesian optimization. While iterative planners yield more refined trajectories. They typically need to evaluate the motion plan significantly more times than sampling-based methods [5].

A standard approach to handle interactions is to generate predictions of the other traffic participants and plan the motion of the AV in a reactive manner without explicitly considering their interactions. Conversely, game-theoretic models can be used to account for different driving styles and/or intentions [6], and are a popular formulation for interactive planning [5]. However, their computational complexity and integration with data-driven methods remain a challenge [5]. Alternatively, modern deep learning prediction models can capture interactions by conditioning the predicted motion of other vehicles on the planned motion of the controlled vehicle. Deep-learned interactive prediction models limit the use of iterative planners due to the complexity of these predictors [5]. However, Chen *et al.* [5] present an iterative planner that is compatible with deep-learned prediction models. This method outperforms a baseline without joint optimization, as well as their baseline sampling-based planner in terms of performance and computational complexity [5], demonstrating the potential of interaction-aware and iterative planners. Such conditional formulations on the interactivity between vehicles are essential for integrated decision-making and planning [1]. While deep learning-based planners are promising, the lack of hard-coded safety measures and generalization issues are some of the challenges that still need to be addressed [2].

1.1.4 Aim of the Thesis

We aim to extend the adaptability and generalizability of interaction-aware motion planning methods to enable AVs to safely navigate in uncertain and dynamic traffic scenarios. In the following sections, we investigate existing methods for interaction-aware motion planning and we identify a gap that limits the AV's adaptability to uncertain and unseen behavior.

1.2 Learning-based Model Predictive Control

Receding horizon control, or model predictive control (MPC), lends itself well to handling state and input constraints of complex systems with multiple inputs and outputs [7] and has been established as the prime methodology for constrained control [8]. Model predictive control relies on a dynamic model to forecast the behavior of the system and iteratively optimize this forecast to determine the best decision [7], for example, what trajectory the AV should follow to progress towards its goal, while respecting specific constraints that follow from traffic laws, other traffic participants and social factors. Due to recent advancements in solvers for nonlinear constrained optimization, MPC is capable of motion planning [1]. Recent works, such as [5, 9–11], utilize MPC for interaction-aware motion planning for AVs. Next, we provide a brief introduction to stochastic MPC and how it can leveraged to adapt to uncertain and unseen behavior.

1.2.1 Stochastic Model Predictive Control

Deterministic MPC is unable to proactively cope with system changes, which can increase uncertainty even more, leading to control performance degradation and potential constraint violation. Thus, online learning of system uncertainty and regular adaptation of uncertainty descriptions, via system re-identification or Bayesian inference, is crucial for maintaining the MPC performance for uncertain systems. Although MPC has seen a lot of attention in the literature for the control of deterministic as well as stochastic systems, the presence of uncertainty is still a major challenge that is receiving considerable attention [7].

By explicitly incorporating a probabilistic description of model uncertainty into an optimal control problem, stochastic model predictive control (SMPC) aims to guarantee robust stability and performance of the closed-loop system in a probabilistic sense. Constraint satisfaction for all possible realizations of the disturbance can lead to unnecessary conservatism and, indeed, is impossible if the probability distribution has infinite support. However, such a probabilistic formulation enables SMPC to handle *chance constraints* which require constraints to be satisfied with a user-defined probability [12].

1.2.2 Dual Control

In particular, with the recent successes in the field of machine learning, and the availability of increased sensing and computational capabilities, there is an increasing interest in learning and data-driven control methods [8]. Passive learning methods use data from online observations to improve the model of the system. However, in passive learning the control policy does not consider the informativity of this data when selecting the control inputs.

In active learning or dual control, a controller aims to improve overall control performance by reducing the system uncertainty in a control-oriented manner, which is known as the exploration-exploitation trade-off, or optimal simultaneous identification and control [8]. The dual control paradigm can be used to control the inputs to an uncertain dynamic system. These inputs have a probing effect for active uncertainty learning, as well

as a directing effect for controlling the dynamic system. Stochastic MPC strategies with dual control effect can hold promise for applications in which, possibly abrupt, unknown changes in the system dynamics can compromise the performance, reliability, and safety of the controlled system [12], such as AVs operating in uncertain and dynamic environments.

1.2.3 Active Learning-Based MPC

The selection of the dual control inputs can be categorized into two classes: implicit and explicit dual control. Implicit dual control relies on approximate dynamic programming, and hence, it is computationally expensive and limited to a specific class of problems, whereas explicit dual control artificially probes the system through a reformulation of the MPC problem. Designing the probing effect of inputs to learn specifically about the control-relevant uncertainty of the system, rather than the general system uncertainty, remains largely an open area of research. Another important challenge in explicit dual control arises from the natural conflict between the objectives of learning and control, which makes tuning these controllers challenging [12, 13]. To this end, Soloperto *et al.* [13] propose a novel active learning MPC framework to overcome these shortcomings.

Learning-based MPC uses learning techniques to adapt the online MPC optimization problem at each time step. Hewing *et al.* [8] provide an overview of recent developments in the field of learning-based MPC. Most interactive learning-based MPC methods implement passive learning. Furthermore, learning-based MPC and dual control in constraint systems remain an open question [8]. To bridge this gap, we aim to develop active learning-based MPC methods that can exploit interaction-aware motion planning for autonomous vehicles.

1.3 Interactive Planning with Gaussian Processes

Integrated decision-making and motion planning are essential for the safe and reliable operation of autonomous vehicles in interaction-driven traffic scenarios. Furthermore, online identification, i.e., learning, is crucial to adapt to unseen situations or behavior [1], and active learning could improve the control performance of the motion planner by safely exploring the state space. In addition, active learning can be utilized to perturb other traffic participants, something that is very natural for human drivers. For example, the AV could safely nudge toward the lane center to indicate that it wants to merge into the adjacent lane. Examples of such interaction-driven traffic scenarios include a lane merging scenario, a left turn at an intersection, or entering and exiting a roundabout or traffic circle.

1.3.1 Lane Merging Scenario

In this thesis, we focus on a lane merging scenario, where the AV aims to merge on a target lane on which two other vehicles are driving. In this scenario, the AV has to decide not only when to merge, it also has to decide where to merge. For example, ahead of both vehicles, in between the vehicles, or behind both vehicles. Consequently, this problem requires both decision-making (where to merge) and planning (when to merge). Furthermore, the vehicles

have to interact on a social level and the AV has to consider who is going to yield or not. Such a scenario could lend itself well as a first step to actively learning interaction dynamics, for it can be characterized by strong interactions and relatively long observation times.

1.3.2 Gaussian Processes Modeling

While MPC is a promising paradigm that enables interpretable control, it relies on a sufficiently descriptive system model for the prediction of future states to optimize control performance and satisfy constraints. Hence, system modeling is a critical element for the success of such control systems. For interaction-aware motion planning, we need to properly model the other traffic agents' reactive behavior before we can exploit it for planning [5].

Model descriptions can be subject to great uncertainty, originating, e.g., from insufficient data, restrictive model classes, parametric uncertainties, or the presence of unmodeled disturbances [8]. In the class of probabilistic models, a distinction is made between parametric and nonparametric uncertainty. Classical system identification focuses on parametric models where the underlying process is assumed to lie in a pre-defined model class and the task is to infer the appropriate parameter values. However, especially for complex systems, nonparametric techniques appear to be more promising [14].

Nonparametric probabilistic learning models are typically based on Gaussian process (GP) regression as these models are both flexible and computationally tractable. Within the past two decades, GPs have been developed as powerful function regressors [15] and GP regression is the most commonly employed technique in learning-based control [8]. One can think of a Gaussian process as a Gaussian random variable that is generalized to functions. A Gaussian process is a mathematical object that describes a distribution over functions, as such, it is characterized by an expected function and a covariance function. This notion of covariance can be used to directly assess the uncertainty of the prediction model [8]. A strong advantage of GPs is that it is possible to approximate the probability of collision when the future states of a vehicle can be represented by such a probability distribution [4]. Through the joint conditioning of the motion of road users, GPs can learn complex interactions between vehicles [10]. As such, Gaussian processes are a strong candidate to learn uncertain and unknown interactions with between the AV and other vehicles online.

1.3.3 Online Learning

Gaussian processes have been successfully used to learn complex dynamics in automotive applications that require fast sampling times and high-fidelity control [16, 17]. Hewing *et al.* [16] pre-compute the covariance of the GP based on the solution of the previous time-step, such that this covariance remains fixed during optimization. Although this enables real-time computations on embedded hardware, such an approach will limit interaction-aware planning. In particular for active learning, where we need to optimize the covariance to exploit the uncertainty of the prediction model, and determine exploratory control inputs that affect the uncertainty. As an AV shares its environment with other traffic participants which the AV cannot control, the reactive behavior of these other agents needs to be taken

into account in motion planning [5]. Accordingly, interactions can be modeled by indirect control over another vehicle [1], which, in the case of GP-MPC, leads to joint optimization via GPs [10]. Moreover, GP models can be adopted in MPC to actively learn the system dynamics [13, 18]. For example, [19] actively learns various single-agent control systems using GPs. However, as far as we know, interactive GP-based MPC has not yet been leveraged for active, nor passive, online learning for interaction-aware motion planning.

1.4 Related Work

This work relates several research fields, namely, interaction-aware motion planning, learning-based model predictive control, and Gaussian process modeling. We leverage GP-based prediction models in a learning-based MPC scheme to learn uncertain or unseen behavior of other road users in a lane merging scenario. In this section, we provide an overview of the related works and identify a gap in the current literature. In section 1.4.1 we provide briefly discuss alternative methods to interaction-aware motion planning. Subsequently, we discuss the current state of the art in GP-MPC for interaction-aware motion planning, and active learning methods in a lane merging scenario, in section 1.4.2 and section 1.4.3, respectively. Finally, we identify a research gap that this thesis aims to bridge to advance interaction-aware motion planning for uncertain and unseen traffic scenarios, in section 1.4.4.

1.4.1 Game-Theoretic and Deep Learning Methods

In addition to GP-based MPC, alternative methods have been used to solve interaction-aware motion planning. As mentioned before, game-theoretic approaches are a popular method to formulate interactions [5]. As such, Evens *et al.* propose a penalty method for interaction-aware planning using generalized potential games [20]. Furthermore, Liu *et al.* [6] use a leader-follower game controller to model the interactions in an MPC-based motion planner for forced lane merging. In order to optimize the interactive behavior, they use imitation learning to approximate the leader-follower game controller by a neural network [6]. However, [6] optimizes over a space of sample trajectories rather than purely optimizing the control inputs. While these methods are promising, computational complexity remains a challenge [5]. Conversely, Chen *et al.* [5] propose a novel iterative method that uses homotopy candidates to solve multiple quadratic program MPCs. Their framework utilizes deep-learned prediction models that capture the interactions between vehicles [5]. As game theoretic and deep learning-based approaches are limited by their complexity [5] and generalizability [2], respectively, we attempt to advance GP-MPC methods for interaction-aware motion planning in the hope that they can resolve these challenges.

1.4.2 Gaussian Process Predictions

Gaussian processes have been used to directly predict the future motion of other vehicles [9–11]. As such, Bethge *et al.* [11] present a multi-modal GP-MPC to predict the future motion of other vehicles on intersections. Gaussian processes have also been used to

learn interactions between vehicles in an overtaking maneuver in autonomous racing [9, 10]. Brüdigam *et al.* [9] sample from a GP-based prediction model to construct tightened half-space constraints for a linear SMPC. However, [9] limits the behavior of the target vehicle such that it cannot *weave* and can only move in one lateral direction. Moreover, [9, 11] sample from a GP based on the current state and do not jointly optimize the control actions and predictions, making their predictions reactive, rather than interactive. Zhu *et al.* [10] present an interactive motion planner that jointly optimizes the control inputs and predictions of the target vehicle. Furthermore, [10] does not pose any limitations on the behavior of the target vehicle. While [9] uses online training, [9] does not actively incentivize exploration, and their method is limited to passive learning. Furthermore, the work of [10, 11] only uses past observations from a fixed training set. The use of GPs to directly predict the driving behavior of other road users in a lane merging scenario is not found in the literature.

1.4.3 Actively Learning Interactions

Model predictive path integral control has been used to actively learn interactions in a lane merging scenario, outperforming passive learning counterparts [21]. Here, Knaup *et al.* [21] utilize a Merge-Reactive Intelligent Driver Model [22] in a particle filter to predict the future motion of other vehicles. However, they make use of a sampling-based planner, which yields less refined trajectories, as discussed in section 1.1.

1.4.4 Research Gap

While several works have made an attempt to advance interaction-aware motion planning using Gaussian process-based model predictive control, there is undiscovered potential in online and active learning for uncertain and unseen traffic scenarios. Although Brüdigam *et al.* [9] use online observations to train a GP-based prediction model, they consider limited interaction dynamics of the target vehicle and do not jointly optimize the predicted motion of the interacting vehicles. Furthermore, [9] lacks an explicit learning incentive in their MPC formulation, i.e., learning of the unknown dynamics is done passively. Whereas [10] jointly optimizes the motion of the interacting vehicles, they only consider offline training data for learning. The literature shows potential for active learning methods [6, 13, 19, 21, 23]. However, active learning-based MPC methods for interaction-driven traffic scenarios remain an open field of research whose relevance and potential are promising.

In conclusion, passive and active online learning has been shown to exploit interactions between vehicles and improve performance in lane merging scenarios. Furthermore, Gaussian processes are successful in predicting complex dynamics using past observations, and have been used for autonomous overtaking. However, GPs have not been exploited in lane merging scenarios. Moreover, neither active nor passive online learning with interactive GP-MPC has yet been used for interaction-aware motion planning.

1.5 Contributions

In summary, autonomous vehicles need to leverage interaction-aware motion planning in order to cope with uncertain and unseen behavior of other road users. Complex interaction-driven traffic scenarios, such as lane merging, could benefit from interaction-aware motion planning that can cope with uncertain and unseen behavior to extend the generalizability of the motion planner. Furthermore, dual control can be utilized to actively perturb other vehicles and explore the state space to improve the AV's understanding of its environment.

To this end, Gaussian process-based MPC lends itself well to online learning in addition to its offline capabilities [23]. Since other works have demonstrated the interactive prediction capabilities with offline learning [10], this study focuses on online learning-based GP-MPC to predict the behavior of other agents. While Gaussian process regression and model predictive control have been shown successful in identifying interactions between vehicles, they have not yet been used to learn these interactions in complex traffic scenarios online, neither passively nor actively. As providing theoretical guarantees for stochastic MPC is a major challenge [24], we confine ourselves to a proof of concept through extensive simulation studies. We perform simulation studies in a lane merging scenario to investigate the potential of online and active learning-based GP-MPC and present a qualitative and quantitative analysis of the GP-MPC as an interactive motion planner.

1.5.1 Research Objective

With the aim of this thesis to develop an interaction-aware motion planner that can cope with uncertain as well as unseen behavior of other vehicles, set the following research objectives are set:

- Extend the capabilities of Gaussian process model predictive control for passive, and active, online learning-based interaction-aware motion planning.
- Verify the performance and safety of the online learning Gaussian process model predictive controller against a baseline controller in a simulated, interactive lane merging scenario.

1.5.2 Statement of Contributions

The aim of this thesis is to develop an interaction-aware motion planner that can cope with uncertain as well as unseen behavior of other vehicles. We attain the aforementioned objective of extending the methods for passive and active online learning-based model predictive control for interaction-aware motion planning for autonomous vehicles and bridge the previously identified research gap through the following contributions. (i) This thesis presents an interaction-aware motion planner that utilizes GP-MPC to learn the interactions between vehicles online and jointly optimize their motions. To the author's best knowledge, this is the first work that learns these interactions online, without any pre-training. This demonstrates the potential of GP-MPC and its ability to adapt to truly unseen behavior. (ii) The

active learning framework for MPC by Soloperto *et al.* [13] is employed to actively explore the state space and perturb other road users with the GP-MPC motion planner. (iii) In addition, this thesis presents a novel MPC-based algorithm that enables the active learning framework from [13] with sparse GP-based [25] predictions. (iv) Lastly, a novel extension of the Intelligent Driver Model [26] is introduced to model the interactive driving policy of another vehicle. This Interactive Intelligent Driver Model smoothly switches between driving modes through an activation function which can be dependent on the state of one or more vehicles. This interactive driver model enables the modeling of complex interactions in simulation studies.

1.5.3 Outline of the Thesis

Firstly, Chapter 2 formulates the problem that is studied in this work, namely, lane merging. Here, we model the lane merging scenario and define policies for the other traffic participants, including a novel extension of the Intelligent Driver Model [26]. Chapter 2 concludes by formalizing the motion planning problem into a finite horizon optimal control problem. Secondly, in Chapter 3 we present the GP-based prediction models that are used to model and indirectly control the future motion of the target vehicle. Chapter 3 provides a brief introduction to GPs and how they can be utilized as prediction models. Subsequently, Chapter 4 details three MPC methods that are studied in this work. Firstly, a baseline MPC that uses a constant-velocity prediction model to predict the future motion of the target vehicle. Secondly, we extend this MPC by including the GP-based prediction model from Chapter 3 in an attempt to improve the predictions and, hence, the performance of the MPC. Thirdly, we use the GP-based prediction model to actively learn the interactions between the AV under control, and the target vehicle. The results of the baseline CV-MPC, and the passive and active online learning GP-MPCs for various test cases are presented in Chapter 5. We investigate the effects of various parameters and aim to provide new insights into the working mechanisms of these methods. Finally, Chapter 6 concludes this thesis and discusses the findings of the research, and provides an outlook for future research.

Chapter 2

The Motion Planning Problem

In Chapter 1, we identify Gaussian process-based MPC as a strong candidate for interaction-aware motion planning in uncertain traffic scenarios, like that of a lane merging scenario. As discussed in section 1.1, motion planning is a task concerned with computing a safe trajectory for the AV to follow. In this chapter, we construct a mathematical optimization problem of the lane merging scenario in order to synthesize a such a trajectory using MPC.

Firstly, we define the lane merging scenario that is considered in this study. Secondly, we discuss the simulation models for the vehicles. Furthermore, in order to devise an interactive driving policy for the Following vehicle, we propose a novel extension of the Intelligent Driver Model (IDM). We conclude this chapter by formulating the motion planning problem as an optimal control problem. Subsequently, Chapter 3 proposes three prediction models that are used to anticipate the behavior of the target vehicles. The MPC algorithms, detailed in Chapter 4, exploit these prediction models to solve the motion planning problem.

2.1 Lane Merging Scenario

This study utilizes a lane merging scenario to investigate the potential of online learning GP-MPC for motion planning. The AV under control, which is referred to as the Ego vehicle, is forced to merge into a target lane since its current lane is closing. The target lane is occupied by other road users. In this scenario, three vehicles compose the scene: two target vehicles, a Follower and a Leader, that are driving in the target lane, and the Ego vehicle that is driving in the merge lane. The Ego and Follower start at the same longitudinal position, next to one another. They are approaching the Leader with a higher speed than that of the Leader, which is assumed to drive at a constant velocity. The Ego vehicle is forced to merge onto the target lane before the end of the merge lane. While doing so, it has to decide if it will merge in between or behind the target vehicles. To this end, the Ego vehicle needs to consider the current and future positions of the target vehicles to plan its motion that will complete the merge effectively. Furthermore, we assume to have access to a map of the road layout, and that the states of the target vehicles at the current time can be measured without noise. A schematic overview of the scenario is found in Figure 2.1.

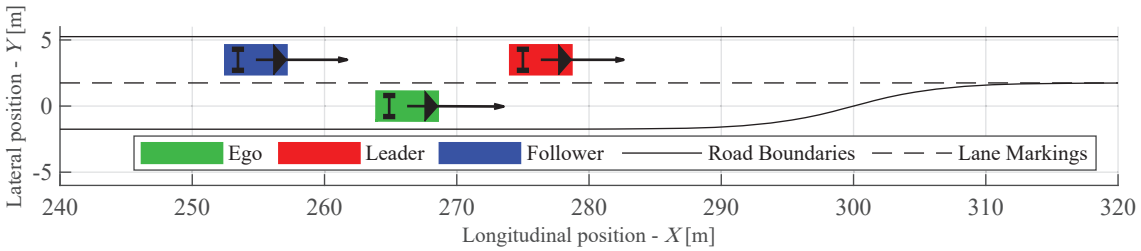

Figure 2.1: Snapshot of the lane merging scenario near the merge point.

2.2 Vehicle Modeling

Now that we have defined the lane merging scenario, we have to construct a simulation model that describes how the different vehicles behave. Firstly, section 2.2.1 describes the vehicle model that is used to simulate the Ego, Follower and Leader. The Follower and Leader are modeled as a closed-loop system, while the inputs for the Ego result from the MPC, detailed in Chapter 4. As mentioned before, the Leader maintains a constant velocity. To simulate interactive driving behavior, the Follower's driving policy is modeled by an Intelligent Driver Model [26]. To this end, section 2.2.2 introduces these models and proposes a novel extension to existing Intelligent Driver Models. In section 2.2.3, we explore various discretization methods of the dynamics to incorporate them in an MPC algorithm. Finally, we discuss some considerations for the sample time of the discretized model.

2.2.1 Kinematic Bicycle Model

All vehicles in the scenario are modeled using a kinematic bicycle model for the evolution of their state. Furthermore, for collision avoidance, they are geometrically modeled as a rectangle with a length L and a width W. The parameters of these vehicles are adopted from a Ford Escape/Kuga and are seen in Table 2.1. For the sake of simplicity, the physical dimensions of all road users are assumed to be the same. The Ego vehicle is being controlled by various proposed MPC algorithms presented Chapter 4. Furthermore, the inputs of the Follower are modeled by a novel interactive variant of the Intelligent Driver Model. Finally, the Leader is assumed to maintain its initial speed throughout the scenario.

Table 2.1: Vehicle and Road Parameters.

Parameter	Value [m]
Length (L)	4.6
Width (W)	2.2
Wheelbase (l)	2.7
Distance to front axle (l_f)	1.35
Distance to front axle rear (l_r)	1.355
Track width (t_w)	1.6
Lane width (W_l)	3.5

System Dynamics The vehicles are modeled by a kinematic bicycle model. This model is a simple kinematic representation of the vehicle dynamics that respects the nonholonomic constraints of a vehicle with sufficient accuracy for motion planning. The kinematic bicycle model lumps two tires of an axle into one. These lumped tires are assumed to roll without any slip. Consequently, the vehicle rotates around an instantaneous center of rotation (IC) which is fully determined by the geometry of the vehicle and the steering angle. Specifically, in this work, the inputs to the kinematic bicycle model are the forward acceleration and steering angle rate. Let us define the state vector $\mathbf{x}$ as:

$$\mathbf{x}(t) = \begin{bmatrix} X(t) \\ Y(t) \\ v(t) \\ \psi(t) \\ \delta(t) \end{bmatrix}, \tag{2.1}$$

where X and Y are the longitudinal and lateral positions of the rear axle in the world frame, respectively. The longitudinal velocity of the rear axle is denoted by v, ψ is the heading angle of the vehicle with respect to the world's longitudinal axis, and δ is the steering angle of the front wheel with respect to the vehicle's longitudinal axis. The global continuous time is denoted by t. The input vector is defined as:

$$\mathbf{u}(t) = \begin{bmatrix} a(t) \\ r(t) \end{bmatrix}, \tag{2.2}$$

where a is the longitudinal acceleration of the rear axle, and r is the steering angle rate. A schematic overview of the kinematic bicycle model is provided in Figure 2.2, below.

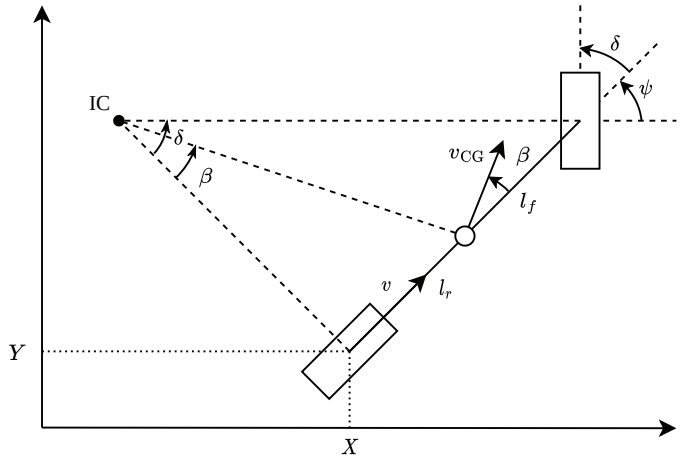

Figure 2.2: Schematic view of the kinematic bicycle Model used for dynamical modeling of the Ego and target vehicles.

The continuous-time dynamics of the kinematic bicycle model, with the rear axle as its point of reference, is defined as follows:

$$\dot{\mathbf{x}}(t) = f_c\left(\mathbf{x}(t), \mathbf{u}\left(t\right)\right), \tag{2.3a}$$

where:

$$\dot{X}(t) = v(t)\cos\left(\psi\left(t\right)\right), \tag{2.3b}$$
$$\dot{Y}(t) = v(t)\sin\left(\psi\left(t\right)\right), \tag{2.3c}$$
$$\dot{v}(t) = a(t), \tag{2.3d}$$
$$\dot{\psi}(t) = \frac{v(t)}{l}\tan\left(\delta\left(t\right)\right), \tag{2.3e}$$
$$\dot{\delta}(t) = r(t). \tag{2.3f}$$

Note that the position of the vehicle is defined with respect to the vehicle's rear axle, which simplifies the dynamics and the involved trigonometry significantly. In contrast to using the vehicle's center of gravity as a point of reference, the body side slip angle (β), that is the angle between the velocity of the center of gravity (or centroid) and the vehicle's longitudinal axis, is not part of the dynamics. Therefore, we can reduce the state of the system which improves online optimization. As we omit dynamics, we assume that the center of gravity coincides the the centroid. The velocity at the center of gravity can be computed afterward as:

$$v_{\mathrm{CG}}(t) = \frac{v(t)}{\cos\left(\beta\left(t\right)\right)}, \tag{2.4}$$

and the body side slip angle as:

$$\beta(t) = \arctan\left(\frac{l_r}{l}\tan\left(\delta\left(t\right)\right)\right), \tag{2.5}$$

where l denotes the vehicle's wheelbase, and l_r is the distance from the center of gravity to the rear axle. Henceforth, we use an index j to refer to a specific agent in the scenario. The Ego, Follower, and Leader are denoted by $j = 1, 2, 3$, respectively.

2.2.2 Intelligent Driver Model

In this section, we first introduce the Intelligent Driver Model (IDM). Secondly, we discuss an extension that considers lateral interactions, namely the Merge-Reactive IDM (MR-IDM). Subsequently, we present a novel extension of the IDM to include interactions with approaching traffic agents, the Interactive IDM. Finally, we apply this extension to the MR-IDM to construct an Interactive Merge-Reactive IDM (I-MR-IDM) that considers approaching traffic agents as well as lateral interactions.

The Intelligent Driver Model [26] is a widely used car-following that has been extended several times. These models use the positions, velocities and accelerations of a *reference*

vehicle to formulate a policy for the acceleration of the following vehicle. These models primarily focus on the preceding vehicle in the same lane. Although [27] proposes a variant of the IDM which additionally considers succeeding vehicles, they do not consider approaching vehicles in adjacent lanes. The Merge-Reactive IDM by [22] considers merging vehicles in adjacent lanes, however, this model assumes that the merging vehicle has passed the Follower. These variants do not consider approaching traffic that influences the driving behavior of the Following vehicle. Moreover, most driver models aim to mimic normal driving behavior, while challenging driving scenarios i.e., edge cases, can help distinguish good planning algorithms from great ones. In order to generate such interactive driving behavior of the Following vehicle, a new variant of the IDM is proposed.

The Intelligent Driver Model with Constant Acceleration Heuristic Specifically, in this study, we extend the IDM with Constant Acceleration Heuristic (IDM-CAH) by [28] which uses a heuristic to reduce unrealistic decelerations of the original IDM [26]. The driving policy of the Follower is characterized by a total of twelve parameters. Eight of these parameters are fixed and are listed in Table 2.2, below.

Table 2.2: Parameters of the Intelligent Driver Model of the Follower.

IDM Parameter	Value
Free acceleration exponent (δ_{IDM})	4 [-]
Jam distance (s_0)	2 [m]
Maximum acceleration ($a_{\max}$)	4 [m/s^2]
Desired deceleration ($b_{\max}$)	3 [m/s^2]
Coolness factor (c)	0.99 [-]
Look back time T_{lookback}	0.4 [s]
Smoothing factor β	2 [-]
Lateral reactivity ζ	2.5 [-]

The original IDM acceleration function reads:

$$a_{\text{IDM}}(s^j, v^1, \Delta v^j) = \frac{\mathrm{d}v^1}{\mathrm{d}t} = a_{\max}\left[1 - \left(\frac{v^1}{v_{\text{ref}}}\right)^{\delta_{\text{IDM}}} - \left(\frac{s^*(v^1, \Delta v^j)}{s^j}\right)^2\right], \tag{2.6}$$

where $\Delta v^j = v^1 - v^j$, and $s^j := X^j - X^1 - L$ is the gap between the reference vehicle and the Follower. Since both the Ego vehicle and the Leader could serve as the IDM model's reference vehicle, depending on the current state, j denotes the index of the reference vehicle i.e., the current *leader*. The maximum acceleration parameter $a_{\max}$ is the maximum acceleration that is employed by the IDM. The free acceleration exponent δ_{IDM} denotes how the acceleration scales with the velocity. In the nominal IDM, the reference velocity v_{ref} is equal to the nominal desired velocity:

$$v_{\text{ref}} := v_{\text{nom}}. \tag{2.7}$$

The effective desired safety gap s^* is defined as:

$$s^*(v^1, \Delta v^j) = s_0 + v^1 T_{\text{ref}} + \frac{v^1 \Delta v^j}{2\sqrt{a_{\max} b_{\max}}}. \tag{2.8}$$

Here, the headway time T_{ref} is equal to the nominal desired headway time:

$$T_{\text{ref}} := T_{\text{nom}}. \tag{2.9}$$

The nominal desired velocity and headway time are defined in Chapter 5, and are specific for each specific test case. The desired deceleration parameter $b_{\max}$ is the *comfortable* deceleration that is employed by the IDM. Note that the actual maximum deceleration output from the IDM can exceed this value in case it has to prevent a collision. Note that $b_{\max}$ is typically set to 2 [m/s^2], however, we increase it to generate more adversarial driving behavior, similar to [6].

The Constant Acceleration Heuristic (CAH) determines, for given values of the gap s^j between the reference vehicle and Follower, the Follower's velocity v^1, the reference vehicle's velocity v^j, and its acceleration a^j, the maximum acceleration a_{CAH} leading to no crashes and is given by:

$$a_{\text{CAH}}\left(s^j, v^1, v^j, a^j\right) = \begin{cases} \frac{\left(v^1\right)^2 \tilde{a}^j}{\left(v^j\right)^2 - 2s\tilde{a}^j} & \text{if } v^j\left(v^1 - v^j\right) \leq -2s^j\tilde{a}^j, \\ \tilde{a}^j - \frac{\left(v^1 - v^j\right)^2 \Theta\left(v^1 - v^j\right)}{2s^2} & \text{otherwise,} \end{cases} \tag{2.10}$$

where $\tilde{a}^j = \min\left(a^j, a_{\max}\right)$, and Θ is the Heaviside step function. The IDM-CAH limits the decelerations of the original IDM as follows:

$$a_{\text{IDM-CAH}}\left(s^j, v^1, v^j, \Delta v^j, a^j\right) = \begin{cases} a_{\text{IDM}} & a_{\text{IDM}} \geq a_{\text{CAH}}, \\ (1-c)a_{\text{IDM}} + c\left[a_{\text{CAH}} + b_{\max}\tanh\left(\frac{a_{\text{IDM}} - a_{\text{CAH}}}{b_{\max}}\right)\right] & \text{otherwise.} \end{cases} \tag{2.11}$$

For more details and motivations for this construction, please refer to [28].

The Merge-Reactive Intelligent Driver Model When a merging vehicle is cutting in front of the Follower, the original IDM is prone to extreme decelerations that are not acceptable (nor possible) in real-world applications [28]. Although the IDM-CAH reduces the unreasonable decelerations during changes of the reference vehicle of the IDM, it still fails to acknowledge the merging vehicle until it becomes its leader [22], as the IDM and IDM-CAH do not feature any lateral awareness or reactivity. Conversely, the Merge-Reactive IDM (MR-IDM) [22] is a novel extension of the IDM-CAH [28] that considers merging agents into the Following vehicle's lane. The MR-IDM uses the visual angle between the Follower and the vehicle that is trying to merge. Firstly, we compute the absolute distances from the Follower to both rear corners of the reference vehicle:

$$d_1^j, d_2^j = \sqrt{(s^j)^2 + (\zeta \Delta Y^j \pm W/2)^2}, \tag{2.12}$$

where $\Delta Y^j = Y_1 - Y_j$, and ζ is a tuning factor to adjust the lateral reactivity. Recall that W is the width of the reference vehicle. Using this absolute distance, the MR-IDM maps the relative longitudinal and lateral position to an effective longitudinal position for the IDM-CAH:

$$s_e^j = \frac{W}{2} \sqrt{\frac{\left(d_1^j + d_2^j\right)^2 - W^2}{W^2 - \left(d_1^j - d_2^j\right)^2}}. \tag{2.13}$$

Subsequently, this effective gap is used as an input for the IDM-CAH:

$$s^j := s_e^j. \tag{2.14}$$

The MR-IDM computes the acceleration of the IDM-CAH with *both* the Ego vehicle and the leader as reference vehicles using the effective distance from (2.13). The reference vehicle that induces the largest deceleration determines the output of the MR-IDM:

$$a_{\text{MR-IDM}}\left(\mathbf{x}^1, \mathbf{x}^0, \mathbf{u}^0, \mathbf{x}^2, \mathbf{u}^2\right) = \min\left(a_{\text{IDM-CAH}}\left(s_e^j, v^1, v^j, \Delta v^j, a^j\right) \text{ for } j = 0, 2\right) \tag{2.15}$$

Consequently, as the Ego vehicle merges into the target lane, the MR-IDM will gradually transition its reference vehicle to the Ego vehicle.

The Interactive Intelligent Driver Model Next, we propose the Interactive Intelligent Driver Model (I-IDM), a general extension to an IDM that is used to characterize the driving policy of the Follower which is interacting with the preceding Leader, as well as with the approaching Ego vehicle in the adjacent lane.

The Follower could change its behavior as the Ego vehicle approaches. The Follower could show social behavior, for example, by increasing the gap allowing the Ego vehicle to merge in-between the Follower and Leader. Conversely, it could close the gap between the Follower and Leader further, thereby showing more adversarial behavior. However, regardless of its behavior, we assume that it will keep the properties of an IDM, namely a collision-free, smooth, and interpretable policy [28]. Whether it opens or closes a gap, it will try to keep some distance from the Leader. This idea is exploited by making the parameters of the IDM state-dependent.

Intuitively, the I-IDM is composed of multiple IDMs that represent different driving styles. For the sake of simplicity, we consider the case where the Follower starts with nominal parameters and switches to adversarial parameters as the Ego vehicle approaches. Without loss of generality, we use a logistic function to activate the parameter change when the Ego approaches the Follower:

$$\alpha = \frac{1}{1 + \ \exp\left(-1/\beta\left(T_{\text{lookback}} v^1 + s^0 + L\right)\right)}, \tag{2.16}$$

where β is a smoothing factor, and T_{lookback} is the time factor that the Follower looks behind him such that the activation function is at a factor of $\alpha = 0.5$ when the approaching

vehicle is $s^j + L = T_{\text{lookback}} v^1 [\text{m}]$ behind the Follower. In turn, the value of the activation function is used to compute a convex combination of the parameters of the two driving styles. Accordingly, we adjust the reference headway time and reference velocity of the IDM as follows:

$$T_{\text{ref}} := (1 - \alpha)\, T_{\text{nom}} + \alpha T_{\text{act}}, \tag{2.17a}$$

$$v_{\text{ref}} := (1 - \alpha)\, v_{\text{nom}} + \alpha v_{\text{act}}, \tag{2.17b}$$

where T_{act} and v_{act} are the *active* headway time and the velocity of the Interactive IDM, e.g., that attempt to close the gap, respectively. The nominal and interactive reference velocity and headway time are defined in Chapter 5, and are specific for each specific test case. The interaction between the different vehicles is governed by an activation function that changes the parameters of the IDM such that the behavior of the Follower smoothly transitions from one driving style to another. As such, one can extend an IDM to an interactive IDM by computing the IDM parameters according to (2.16) and (2.17). Subsequently, we discuss some properties, extensions, and limitations of this novel Interactive IDM.

Continuity of Interactive Intelligent Driver Model The I-IDM's acceleration is a continuous function, provided that the underlying IDM version is continuous. Note that the IDM-(CAH) is a continuous function that maps the relative positions, velocities, and accelerations of the Follower and its reference vehicle to an acceleration of the Follower [28]. Firstly, the I-IDM can be regarded as a time-varying convex combination of IDMs. The preservation of smoothness is ensured by the convex combination of these IDMs, as the convex combination of two smooth functions results in a smooth function. Furthermore, the I-IDM incorporates a smooth, state-dependent activation function that adjusts the parameters of the IDM. In conclusion, given that the composition of functions maintains smoothness, it can be deduced that the resultant I-IDM also produces a continuous acceleration.

Extension to Interactive Merge-Reactive IDM The MR-IDM is extended by including state-dependent parameters that are governed by an activation function (2.16) to model the interactions between different agents in the scene, as described above. In the remainder of this study, we model the Follower by a novel Interactive Merge Reactive Intelligent Driver Model (I-MR-IDM). This I-MR-IDM transitions between two parameter settings for the MR-IDM as per (2.17). The resulting model has both the lateral reactivity of the MR-IDM, as well as the state-dependent parameters of the I-IDM. However, the Interactive-IDM can employ different variants of the IDM.

Limitations of the MR-IDM As mentioned before, the MR-IDM only considers merging vehicles that have passed the Follower. The MR-IDM only acknowledges the Ego vehicle as a reference vehicle when it is beside the Follower. Consequently, a jump in the dynamics may occur when the Follower suddenly starts reacting to the merging agent. To circumvent

this behavior, the MR-IDM needs to be properly tuned such that it is not reacting to a passing vehicle, but only to a merging vehicle. Recall that the MR-IDM uses the visual angle with the merging agent to project the position of the merging agent to an effective longitudinal distance for the underlying IDM. To have a smooth transition of the reference vehicle from the leader to the merging agent, the effective distance should transition smoothly. Nevertheless, potentially strong decelerations are mitigated by the constant acceleration heuristic of the IDM-CAH. The focus of this work is on motion planning and the I-MR-IDM is developed to generate relevant and sufficiently realistic driving behavior. Since the I-MR-IDM works sufficiently well, further analysis of this model is outside the scope of this work. In future work, the I-MR-IDM could be analyzed in more detail.

2.2.3 Discretization

As the motion planning problem is continuous by nature, we initially want to solve a continuous-time optimal control problem (OCP). However, analytical solutions to these problems are limited to special classes of problems. Alternatively, indirect methods typically result in the formulation of a boundary-value problem. Conversely, direct methods convert the infinite-dimensional continuous-time OCP to a finite-dimensional OCP. By discretizing the problem first, we reduce the problem to an initial-value problem, which, in turn, can be optimized numerically [7].

Direct Methods Direct methods are very successful and most widely used in MPC [7]. These methods are supported by sophisticated and dedicated solvers, like IPOPT [29]. Therefore, we employ a direct method for the MPC algorithms which are discussed in Chapter 4. For these direct methods, we describe the behavior of the Ego vehicle being controlled by a discrete-time dynamical system model with a constant sampling time T_s. To this end, we can define a sampled discrete-time state vector $\mathbf{x}_k$, which denotes the state vector at time $t_k = kT_s$:

$$\mathbf{x}_k := \mathbf{x}\left(t_k\right). \tag{2.18}$$

Analytical Solution The solution of an MPC is typically a piecewise-constant trajectory with segments of T_s seconds composed of a sequence of control inputs. The analytical solution to the differential equation in (2.3), for a constant input $\mathbf{u}_k$ over an integration step of T_s seconds, is computed using the symbolic computation library of Wolfram Mathematica [30]. The analytical solution of the continuous-time dynamical system for piecewise-constant

inputs reads:

$$X_{k+1} = X_k + 2l\cos\left(\psi_k + \frac{\left(a_k T_s^2 + 2v_k T_s\right)\tan\left(\delta_k\right)}{4l}\right)\cot\left(\delta_k\right)\sin\left(\frac{\left(a_k T_s^2 + 2v_k T_s\right)\tan\left(\delta_k\right)}{4l}\right), \tag{2.19a}$$

$$Y_{k+1} = Y_k + 2l\cos\left(\psi_k\right)\cot\left(\delta_k\right)\sin^2\left(\frac{\left(a_k T_s^2 + 2v_k T_s\right)\tan\left(\delta_k\right)}{4l}\right) + l\cot\left(\delta_k\right)\sin\left(\psi_k\right)\sin\left(\frac{\left(a_k T_s^2 + 2v_k T_s\right)\tan\left(\delta_k\right)}{4l}\right), \tag{2.19b}$$

$$v_{k+1} = v_k + T_s a_k, \tag{2.19c}$$

$$\psi_{k+1} = \psi_k + \frac{\left(a_k T_s^2 + 2v_k T_s\right)\tan\left(\delta_k\right)}{2l}, \tag{2.19d}$$

$$\delta_{k+1} = \delta_k + T_s r_k. \tag{2.19e}$$

Although there exists an analytical solution to the continuous-time dynamical system for piecewise-constant inputs, this solution is rather complicated. Since the system is used in an optimization scheme, we need to compute derivatives of this solution. To this end, numerical integration can be used to approximate the solution to the continuous-time system with sufficient accuracy, while simplifying the expressions of the solution and its derivative significantly. Two discretization methods for numerical integration are considered, namely the Forward Euler and Runge-Kutta 4 methods. These methods are validated by the analytical solutions to the continuous-time dynamical system for piecewise-constant inputs.

Forward Euler Method The numerical integration of the continuous-time dynamical system in (2.3) can be approximated by the Forward Euler method, which can be obtained through a first-order Taylor Expansion of a (vector) function $\mathbf{x}$ around t:

$$\mathbf{x}(t + T_s) = \mathbf{x}(t) + T_s\ \dot{\mathbf{x}}(t) + \mathcal{O}\left(T_s^2\right). \tag{2.20}$$

Recall the continuous-time dynamical system (2.3):

$$\dot{\mathbf{x}}(t) = f_c\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right)\right), \tag{2.21}$$

By substituting (2.3) for the first derivative of $\mathbf{x}(t)$ and by omitting second-order terms, we obtain the Forward Euler approximation:

$$\mathbf{x}\left(t + T_s\right) \approx \mathbf{x}(t) + T_s\ f_c\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right)\right). \tag{2.22}$$

Let us introduce a discrete-time variable $k \in \mathbb{N}$:

$$\mathbf{x}\left(\left(k+1\right)T_s\right) \approx \mathbf{x}(kT_s) + T_s\ f_c\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right)\right). \tag{2.23}$$

Employing the short-handed notation that we introduced earlier, we have:

$$\mathbf{x}_{k+1} \approx \mathbf{x}_k + T_s \; f_c\left(\mathbf{x}_k, \mathbf{u}_k\right). \tag{2.24}$$

This yields the following discrete-time dynamical system:

$$X_{k+1} = X_k + T_s v_k \cos\left(\psi_k\right), \tag{2.25a}$$
$$Y_{k+1} = Y_k + T_s v_k \sin\left(\psi_k\right), \tag{2.25b}$$
$$v_{k+1} = v_k + T_s a_k, \tag{2.25c}$$
$$\psi_{k+1} = \psi_k + T_s \, \frac{v_k}{l} \tan\left(\delta_k\right), \tag{2.25d}$$
$$\delta_{k+1} = \delta_k + T_s r_k. \tag{2.25e}$$

Runge-Kutta Method Alternatively, the solution to the nonlinear differential equations from (2.3) can be approximated with the classical fourth-order Runge-Kutta (RK4) method [7]. For a constant input of $\mathbf{u}_k$ over one sample time T_s, the system is discretized as follows:

$$\mathbf{x}_{k+1} \approx f\left(\mathbf{x}_k\right) := \mathbf{x}_k + \frac{T_s}{6}\left(h_1 + 2h_2 + 2h_3 + h_4\right) + \mathcal{O}\left(T_s\right)^5 \tag{2.26a}$$
$$\tag{2.26b}$$

for $k = 0, 1, 2, 3, \ldots$, using:

$$h_1 = f_c\left(\mathbf{x}\left(t_k\right), \mathbf{u}\left(t_k\right)\right), \tag{2.26c}$$
$$h_2 = f_c\left(\mathbf{x}\left(t_k\right) + T_s \frac{h_1}{2}, \mathbf{u}\left(t_k\right)\right), \tag{2.26d}$$
$$h_3 = f_c\left(\mathbf{x}\left(t_k\right) + T_s \frac{h_2}{2}, \mathbf{u}\left(t_k\right)\right), \tag{2.26e}$$
$$h_4 = f_c\left(\mathbf{x}\left(t_k\right) + T_s k_3, \mathbf{u}\left(t_k\right)\right). \tag{2.26f}$$

Note that the discrete-time system uses four evaluations of the continuous-time dynamical system f_c in (2.3) to approximate the integration of the dynamics.

Validation of Discretization Methods The Forward Euler and Runge-Kutta discretization methods are validated using the analytical solution to the system's differential equation in (2.3). We integrate the system with a constant acceleration of $a = 5$ [m/s^2] and constant steering angle of $\psi = 5$ [deg.] for 10 seconds using the different integration methods. Table 2.3 shows the local truncation errors of the Forward Euler and Runge-Kutta 4 methods with respect to the analytical solution for three different sampling times. Note that the approximations of the linear subequations are exact for all methods and all sampling times.

The Forward Euler method is a computationally cheap discretization method. However, it requires a very small sample time T_s to have acceptable accuracy. as the local truncation

Table 2.3: Local Truncation Error of Integration of Continuous-Time Dynamics

Step Size [s]	State	Forward Euler - RMS Error	RK4 - RMS Error
1e0	X [m]	2.0e1	9.9e-3
	Y [m]	1.3e1	1.5e-2
	ψ [rad.]	4.8e-2	9.7e-16
1e-1	X [m]	2.1e0	9.4e-7
	Y [m]	1.2e0	1.4e-6
	ψ [rad.]	4.7e-3	4.9e-16
1e-2	X [m]	2.1e-1	9.4e-11
	Y [m]	1.2e-1	1.4e-10
	ψ [rad.]	4.7e-4	7.0e-17

error is $\mathcal{O}\left(T_s^2\right)$. The classic fourth-order Runge-Kutta method (RK4) provides a good trade-off between computational complexity and accuracy, with a local truncation error in $\mathcal{O}\left(T_s^5\right)$ [31]. Henceforth, the system dynamics in (2.3) are discretized using RK4, and the system (2.26) will be used to simulate all vehicles as well as the Ego's prediction model.

Discrete-Time Intelligent Driver Model Although the IDMs, discussed in section 2.2.2, are policies that map continuous-time variables to an acceleration, they are to be discretized in a digital simulation environment. To this end, the trajectories of the states and inputs are composed of piecewise-constant signals. The inputs to the IDM are piecewise-constant and therefore, also the resulting acceleration from the IDM is a piecewise constant signal.

2.2.4 Sample Time

The system's sample time has to be carefully designed such that discretization errors are within acceptable limits while limiting the computational burden. Moreover, MPC using direct methods typically provides point-wise constraint satisfaction. Hence, a sufficiently high sampling frequency is required to have proper inter-sample constraint satisfaction. Due to the small integration error of the RK4 method, the sample time is primarily limited by the collision avoidance constraints. As this study focuses on a proof of concept, we do not consider real-time implementation. However, real-time implementation imposes strict constraints on the sample time, in addition to the considerations discussed next.

Computational Complexity In Chapter 4 some novel MPC algorithms will be introduced that are relatively complex, compared to nominal MPC. This should be considered when selecting the sample time to keep them consistent throughout the different methods, while maintaining computational tractability for the most complex algorithms. At a smaller sample time, we need to extend the prediction horizon N to have the same horizon time in the continuous time. This leads to increased complexity of the OCP. Moreover, sampling at a too high frequency can lead to poor convergence and numerical errors with the GP.

Inter-Sample Constraint Satisfaction Next to a lower bound, we also have an upper bound on the sample time. In case we have a too low sampling frequency, the Ego vehicle can be behind a target vehicle at time k, then it could *pass* this vehicle and be in front of the same target vehicle at time $k+1$ while satisfying all collision avoidance constraints at the sample times. However, the continuous-time system would have collided with the target vehicle. This is a critical consideration when selecting the appropriate sample time.

Although no attempt is made to guarantee inter-sample constraint satisfaction, we can approximate the minimally required sample frequency to prevent an inter-sample collision. If we only consider the longitudinal distance, then the sample time must be sufficiently small such that a target vehicle cannot be passed provided a maximum allowed velocity and the minimal length of the vehicles in the scene.

Approximation of Required Sample Time When we consider an operational design domain with a maximum velocity difference of 15 [m/s], then a sampling frequency of $f_s = 4$ [Hz] is able to account for vehicles with a length of $L \geq 3.75$ [m]. Considering the vehicles in our scenario are of length $L = 4.62$ [m], this is sufficient to guarantee inter-sample constraint satisfaction when the concerned vehicles have the same heading angle (ψ) and lateral position (Y). Moreover, we have some margin to accommodate slight offsets in heading angle and lateral position. A sampling frequency of $f_s = 4$ [Hz] shows adequate constraint satisfaction with and without a relative heading between the vehicles, and is henceforth employed in the remainder of this study.

Horizon Time Furthermore, we have to consider a desired prediction horizon time in continuous time, e.g., 3 seconds into the future. A too small horizon time can compromise both performance as well as safety. While a too long prediction horizon can also lead to poor control policy if those predictions are inaccurate. As mentioned before, extending horizon length leads to increased computational complexity and solve times. Motion prediction models typically use a prediction horizon in the order of several seconds [32]. In our case, to obtain a prediction time of 3 seconds with a sampling frequency of 4 Hz, we would need a prediction horizon of length $N = 12$ which is found to be acceptable considering the complexity of the problem.

2.3 Optimal Control Problem

In the previous section, we constructed the models for the Ego vehicle as well as the target vehicles. The solution to the motion planning problem is a trajectory i.e., a sequence of states through space and time, that the Ego vehicle is to follow to safely navigate in the traffic while it interacts with the other vehicles. We proceed by formulating the motion planning problem as a finite horizon optimal control problem (OCP). To this end, we consider a multi-layer autonomy architecture where we focus on high-level planning. It is assumed that we have perfect knowledge of the Ego vehicle and its environment, and that the computed trajectory can be perfectly tracked by a lower-level control layer. The aim of

the MPC algorithm, detailed in Chapter 4, is to determine the appropriate input sequence that navigates the Ego vehicle through the traffic while minimizing some objective function and satisfying a number of constraints that aim to provide a degree of safety and comfort.

In this section, we first define the objective function that we use to tune the output of the motion planner. Secondly, we construct a set of constraints that aim to enforce collision avoidance, road boundaries, and traffic laws. Conclusively, these ingredients are combined to construct the OCP that is parsed to the MPC algorithm. As we assume perfect knowledge of the Ego, we can plan its motion perfectly. Furthermore, we assume that the Leader maintains a constant velocity. As for the motion prediction of the Follower, we propose a novel Gaussian process-based prediction model in Chapter 3.

2.3.1 Control Objective

We aim to develop an interaction-aware GP-MPC motion planner that can adapt to uncertain and unseen behavior. The focus of this work is to analyze the potential shortcomings of a constant velocity prediction model, and the potential advantages of a Gaussian process prediction model, for the predictions of other road users. In an attempt to compare these models in an objective and fair manner, the objective function is designed to be agnostic and unbiased toward any of the prediction models.

Causality Dilemma One could argue that a reference trajectory is the product of the motion planning problem. Therefore, providing any bias toward decision-making through the construction of the objective function could be considered invalid in the sense that this construction contains part of the solution. As such, in this work, we adopt an objective function that is unbiased and the incentive to merge into the target lane follows from the current state of the scenario, not from the objective function. This automatically implies that neither of the prediction models is utilized to its full potential as it can be expected that the objective function that provides the *best* performance — regardless of its definition — is dependent on the prediction model that is used. However, it does provide a true comparison of the different prediction models and their strengths and weaknesses.

Objective Function In an attempt to limit such any bias from the objective function bias, this objective function simply incentivizes the Ego vehicle to maintain its initial velocity and stay in its lane, until the merge lane closes and the MPC decides we should merge. Firstly, deviations from Ego's initial velocity are penalized. Secondly, the objective function aims to limit the heading angle and steering angle. Thirdly, the objective function limits the control inputs and their rate of change. Accordingly, the reference $\mathbf{x}^r(k)$ is defined as:

$$\mathbf{x}^r(k) = \begin{bmatrix} 0 & 0 & v^0(0) & 0 & 0 \end{bmatrix}^\top. \tag{2.27}$$

We do not steer the Ego to a particular lateral position, rather, we devise a non-convex objective function to allow driving in the merge lane as well as the target lane, such that the merge point is defined by the solution of the problem. Accordingly, we define the

function $m(X) : \mathbb{R} \to \mathbb{R}$ that describes the lane center of the merge lane as a function of the longitudinal position and coincides with the center of the target lane after the merge lane has fully closed:

$$m(X) = \frac{W_l}{1 + e^{-0.3(X-300)}}, \tag{2.28}$$

here, the merge point is set at 300 m, and the factor 0.3 is used to control the gradient of the lane center as a function of the longitudinal position X. As such, we prevent any bias toward one of the two lanes through the construction of the objective function. Furthermore, we define the world coordinate frame to have its origin at $(X, Y) = (0, 0)$ and the longitudinal X-axis to be along the lane center of the merge lane (before it starts closing). This coordinate frame is depicted in Figure 2.1. The resulting primary objective function reads:

$$\begin{aligned} J(\mathbf{x}(k), \mathbf{u}(k-1), \mathbf{U}_k) = &\left\|\mathbf{x}_{N|k} - \mathbf{x}_k^r\right\|_P^2 + \left(Y_{N|k} - W_l\right)^2 P_Y \left(Y_{N|k} - m\left(X_{N|k}\right)\right)^2 \\ &+ \sum_{i=0}^{N-1} \left\|\mathbf{x}_{i|k} - \mathbf{x}_k^r\right\|_Q^2 + \left(Y_{i|k} - W_l\right)^2 Q_Y \left(Y_{i|k} - m\left(X_{i|k}\right)\right)^2 \\ &+ \left\|\mathbf{u}_{i|k}\right\|_R^2 + \left\|\boldsymbol{\Delta}\mathbf{u}_{i|k}\right\|_S^2, \end{aligned} \tag{2.29}$$

where $P, Q, S \succeq 0$ are positive (semi-) definite weighting matrices, $R \succ 0$ is a positive definite weighting matrix, and $P_Y, Q_Y \geq 0$ are non-negative weighting coefficients. The prediction of the state $\mathbf{x}$ at time step $k + i$ at the current time t_k is denoted by $\mathbf{x}_{i|k}$, and

$$\mathbf{U}_k = \left(\mathbf{u}_{0|k}, \ldots, \mathbf{u}_{N-1|k}\right) \tag{2.30}$$

is the sequence of control inputs over the horizon length of $N \in \mathbb{N}_+$ at time t_k.

2.3.2 Constraints

To derive a trajectory for the Ego vehicle that is reasonably safe, comfortable, and socially acceptable for both its occupants and other road users, a set of state and input constraints is introduced with the intention of delimiting the solution.

State and Input Constraints Firstly, let us define a set of constant state constraints. These partially govern the road boundaries and limit the heading angle and steering angle of the Ego vehicle to 0.2618 [rad.] $\approx$ 15 [deg.]. Furthermore, a set of box input constraints limit the steering acceleration and steering angle rate to 5 [m/s^2] and 0.0873 [rad./s] $\approx$ 5 [deg./s], respectively:

$$\mathbf{u}_{\min} \leq \mathbf{u}_{i|k} \leq \mathbf{u}_{\max}, \quad \text{for } i = 0, \ldots, N-1 \tag{2.31a}$$

$$\mathbf{x}_{\min} \leq \mathbf{x}_{i|k} \leq \mathbf{x}_{\max}, \quad \text{for } i = 0, \ldots, N, \tag{2.31b}$$

where

$$\mathbf{u}_{\min} = \begin{bmatrix} -5 \\ -0.0873 \end{bmatrix}, \qquad \mathbf{u}_{\max} = \begin{bmatrix} 5 \\ 0.0873 \end{bmatrix}, \tag{2.32a}$$

$$\mathbf{x}_{\min} = \begin{bmatrix} -\infty \\ -\infty \\ 0 \\ -0.2618 \\ -0.2618 \end{bmatrix}, \qquad \mathbf{x}_{\max} = \begin{bmatrix} \infty \\ W_l + \frac{W_l - W}{2} \\ 37.5 \\ 0.2618 \\ 0.2618 \end{bmatrix}. \tag{2.32b}$$

Note that the lower bound on the lateral position is varying and is, therefore, governed by a function that we introduce next.

Closing Merge Lane Secondly, as the right lane boundary is dependent on the longitudinal position, this constraint is enforced through a separate constraint function:

$$\begin{aligned} &h_r\left(\mathbf{x}_{i|k}\right) = m(X_{i|k}) - Y_{i|k} + \frac{W - W_l}{2}, \\ &h_r\left(\mathbf{x}_{i|k}\right) \leq 0, \quad \text{for } i = 0, \ldots, N, \end{aligned} \tag{2.33}$$

where the right road boundary is parallel to the lane center of the merge lane, defined in (2.28), W and W_l denote the vehicle width and the lane width, as defined in Table 2.1, respectively. Figure 2.1 shows a schematic overview of the scenario.

Collision Avoidance Collision avoidance is governed by a safety ellipse that constrains the centroid of the target vehicles to be outside an ellipse surrounding the Ego vehicle. This ellipse is defined by the following constraint function:

$$\begin{aligned} &h_c\left(\mathbf{x}_i^0, \mathbf{x}_i^j\right) = -\frac{\left(c_{x,i}^1 - c_{x,i}^0\right)^2}{\mathcal{E}_{c,A}^2} - \frac{\left(c_{y,i}^1 - c_{y,i}^0\right)^2}{\mathcal{E}_{c,B}^2} + 1 \\ &h_c\left(\mathbf{x}_i^0, \mathbf{x}_i^j\right) \leq 0, \quad \text{for } i = 0, \ldots, N, \end{aligned} \tag{2.34}$$

where $c_{x,i}^j$ and $c_{y,i}^j$ denote the longitudinal and lateral component of the geometric center of the j^{th} vehicle at prediction step i, respectively. The major and minor semi-axis of the ellipse are denoted by $\mathcal{E}_{c,A}$ and $\mathcal{E}_{c,B}$, respectively. This ellipse accounts for the size of the Ego vehicle and the target vehicle. Furthermore, the relative heading angle between the vehicles can be accounted for by enlarging the ellipse such that under a maximum allowed heading angle, no collision shall occur provided that our prediction model is correct.

Firstly, the minor semi-axis of the ellipse is fixed such that the vehicles can safely pass one another in adjacent lanes without unnecessary interference. The minor semi-axis length is determined by the vehicle width W, with some additional margin:

$$\mathcal{E}_{c,B} = W + 0.82 = 3 \text{ [m]}. \tag{2.35}$$

Secondly, the major semi-axis of the ellipse is determined by a maximum assumed relative angle between the concerned vehicles. Considering the state constraints limiting the Ego vehicle to a heading angle of $|\ \psi_{\max}\ | = 0.2618$ [rad.] ≈ 15 [deg.], assuming the target vehicles drive in a straight line and that the target has a zero heading angle. Subsequently, one can determine the minimal major semi-axis length of the ellipse as:

$$\mathcal{E}_{c,A} = 10.47 \text{ [m]}. \tag{2.36}$$

The major semi-axis length of the ellipse is calculated using a heuristic, for details on this heuristic refer to Appendix A. For further reference, the eccentricity of the ellipse and its effect on the control performance is researched in detail in [33].

Note that this ellipse does not consider any social driving behavior. This is accounted for by a softly constrained *social ellipse*. This ellipse is typically larger than the nominal safety ellipse. To promote social driving whenever possible, let us define the social collision avoidance constraint as:

$$\begin{aligned} h_s\left(\mathbf{x}_i^0, \mathbf{x}_i^j\right) &= -\frac{\left(c_{x,i}^1 - c_{x,i}^0\right)^2}{\mathcal{E}_{s,A}^2} - \frac{\left(c_{y,i}^1 - c_{y,i}^0\right)^2}{\mathcal{E}_{s,B}^2} + 1 \\ h_s\left(\mathbf{x}_i^0, \mathbf{x}_i^j\right) &\leq 0, \quad \text{for } i = 0, \ldots, N, \end{aligned} \tag{2.37}$$

where the minor semi-axis is kept the same to allow vehicles to pass one another:

$$\mathcal{E}_{s,B} = \mathcal{E}_{c,B} = 3 \text{ [m]}. \tag{2.38}$$

The major semi-axis of the social ellipse is extended to promote the Ego vehicle to keep a social distance whenever possible:

$$\mathcal{E}_{s,A} = 20 \text{ [m]}. \tag{2.39}$$

Expanded Collision Avoidance Ellipse As mentioned before, Chapter 3 expands on the prediction models that provide a predicted trajectory of the Following vehicle. These predictions can be used to anticipate the future positions of the Follower and account for this to prevent a collision in the future. However, the collision avoidance constraints in (2.34) are deterministic and therefore assume perfect knowledge of the future states of the target vehicles. Still, these constraints can be extended to account for a degree of uncertainty in these predictions of the other vehicles.

Similar to [10], a minimum covering ellipse is used to represent the target vehicle. Considering that the predictions of the Follower over the horizon are random variables, collision avoidance is governed in a probabilistic sense, such that we have constraint satisfaction with a user-defined probability p_x:

$$\Pr\left(h_c\left(\mathbf{x}_i^0, \mathbf{x}_i^j\right) \leq 0\right) \geq p_x, \quad \text{for } i = 0, \ldots, N. \tag{2.40}$$

Such probabilistic constraints, or chance constraints, can be transformed into deterministic constraints by means of constraint tightening. Linear constraints allow for the separation

of the stochastic and deterministic components of a chance constraint, as such, they can be adopted in a deterministic optimal control problem, as seen in [34]. Generally, nonlinear constraints cannot be decomposed into deterministic and stochastic components and need some form of approximation to be cast into a tractable optimal control problem.

To account for the uncertainty in the predicted position of the target vehicle over the horizon, the semi-axes of these ellipses can be scaled to account for the variance in the longitudinal and lateral positions. Zhu *et al.* [10] consider longitudinal and lateral uncertainty while performing an overtaking maneuver on a race track. However, in the case of a lane merging scenario, the road users have to abide by written and unwritten traffic laws. Hence, we can safely assume that the target vehicle in the target lane will remain in its lane and therefore assume no uncertainty in the lateral motion of the target vehicle. By expanding the major axis of the ellipse to account for longitudinal uncertainty, we obtain a deterministic constraint that approximates constraint satisfaction with a user-defined probability, with the assumption that the calculated probability distribution is correct:

$$\begin{aligned} h_e\left(\mathbf{x}_i^0, \mathbf{x}_i^1, \Sigma_i^{X^1}, \sigma\right) &= -\frac{\left(c_{x,i}^1 - c_{x,i}^0\right)^2}{\left(\mathcal{E}_{c,A} + \sigma\sqrt{\Sigma_i^{X^1}}\right)^2} - \frac{\left(c_{y,i}^1 - c_{y,i}^0\right)^2}{\mathcal{E}_{c,B}^2} + 1 \\ h_e\left(\mathbf{x}_i^0, \mathbf{x}_i^1, \Sigma_i^{X^1}, \sigma\right) &\leq 0, \quad \text{for } i = 0, \ldots, N, \end{aligned} \tag{2.41}$$

where $\Sigma_i^{X^1}$ denotes the covariance of the longitudinal position of the Follower X^1 at prediction step i. The probability of stochastic constraint satisfaction can be tuned with $\sigma \geq 0$, such that the approximate probabilistic constraints are enforced with a probability of:

$$\Pr\left(h_c\left(\mathbf{x}_i^0, \mathbf{x}_i^j\right) \leq 0\right) \geq \operatorname{erf}\left(\frac{\sigma}{\sqrt{2}}\right), \tag{2.42}$$

where

$$\operatorname{erf}(z) = \frac{z}{\sqrt{2}} \int_0^z e^{-t^2} \mathrm{d}t \tag{2.43}$$

and σ is the number of standard deviations which are accounted for.

A relatively large uncertainty in the longitudinal direction will result in a very eccentric ellipse, this would lead to additional conservatism either in the longitudinal or in the lateral direction. Although this could be a potential drawback of such an expanded ellipse in lane merging scenarios, it is found that the expanded ellipses work sufficiently well. The implementation of the expanded collision avoidance constraints is discussed in Chapter 4.

2.3.3 Prediction Models

In order to find a solution to the OCP that considers the future states of the target vehicles, we need some predictions that provide a belief of the states of the Ego as well as the target

vehicles. Firstly, we assume to have perfect knowledge of the Ego's dynamics and therefore its predictions:

$$\mathbf{x}^0_{i|k+1} = f^0\left(\mathbf{x}^0_{i|k}, \mathbf{u}_{i|k}\right) = f\left(\mathbf{x}^0_{i|k}, \mathbf{u}_{i|k}\right), \tag{2.44}$$

where f^0 denotes the nominal prediction model for the Ego vehicle, f is the sampled dynamical system defined in (2.26) — recall that the f is the sampled approximation of f_c from (2.3). Here, $\mathbf{x}^0_{i|k}$, denotes the prediction of state $\mathbf{x}^0$ at future time $t = (k+i)\,T_s$, at current time $t = kT_s$.

As stated in section 2.1, we assume that the Leader maintains its initial velocity, zero heading, and steering angle, hence, and remains in its lane. Accordingly, its predictions are also true:

$$\mathbf{x}^2_{i|k+1} = f^2\left(\mathbf{x}^0_{i|k}, \mathbf{0}\right) = f\left(\mathbf{x}^0_{i|k}, \mathbf{0}\right), \tag{2.45}$$

As a baseline, we consider a constant velocity prediction model for both the Leader as well as the Following vehicle:

$$\bar{v}^1_{i+1|k} = \bar{v}^1(k), \text{ for } i = 0, \ldots, N-1, \tag{2.46}$$

where $v^1_{0|k} = v^1(k)$. Hence, the predictions of the longitudinal position read as follows:

$$\bar{X}^1_{i+1|k} = \bar{X}^1_{i|k} + T_s v^1_{0|k}, \text{ for } i = 0, \ldots, N-1. \tag{2.47}$$

Conclusively, the nominal prediction model of the target vehicles is:

$$\mathbf{x}^j_{i+1|k} = f^j\left(\mathbf{x}^j_{i|k}\right) = A\mathbf{x}^j_{i|k}, \quad \text{for } i = 0, \ldots, N-1, \text{ and } j = 1, 2, \tag{2.48}$$

where the system matrix A is defined as:

$$\begin{bmatrix} 1 & 0 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.49}$$

where $\bar{\mathbf{x}}^1_{0|k} = \mathbf{x}^1(k)$.

Recall that the Follower does not maintain a constant velocity, rather, it is controlled by the novel Interactive Merge-Reactive Intelligent Driver Model, presented in section 2.2.2. One can imagine that in this case, a constant velocity (CV) prediction model cannot capture these dynamics. To this end, Chapter 3 introduces a Gaussian process-based prediction model that exploits Bayesian inference to improve its predictions of the Follower.

2.3.4 Model Predictive Control

By combining the ingredients defined in the section above, the motion planning problem can formulated as a finite horizon optimal control problem (OCP):

$$\min_{\mathbf{U}_k, \boldsymbol{\epsilon}_k} J\left(\mathbf{x}^0(k), \mathbf{u}(k-1), \mathbf{U}_k\right) \tag{2.50a}$$

$$\text{s.t. } \mathbf{x}^0_{i+1|k} = f^0\left(\mathbf{x}^0_{i|k}, \mathbf{u}_{i|k}\right), \qquad i = 0, \dots, N-1 \tag{2.50b}$$

$$\mathbf{x}^1_{i+1|k} = f^1\left(\mathbf{x}^1_{i|k}\right), \qquad i = 0, \dots, N-1 \tag{2.50c}$$

$$\mathbf{x}^2_{i+1|k} = f^2\left(\mathbf{x}^2_{i|k}\right), \qquad i = 0, \dots, N-1 \tag{2.50d}$$

$$\mathbf{x}^0_{min} \leq \mathbf{x}^0_{i|k} \leq \mathbf{x}^0_{max}, \qquad i = 0, \dots, N \tag{2.50e}$$

$$\mathbf{u}_{min} \leq \mathbf{u}_{i|k} \leq \mathbf{u}_{max}, \qquad i = 0, \dots, N-1 \tag{2.50f}$$

$$h_c\left(\mathbf{x}^0_{i|k}, \mathbf{x}^1_{i|k}\right) \leq \boldsymbol{\epsilon}_{i|k,1}, \qquad i = 0, \dots, N \tag{2.50g}$$

$$h_c\left(\mathbf{x}^0_{i|k}, \mathbf{x}^2_{i|k}\right) \leq \boldsymbol{\epsilon}_{i|k,2}, \qquad i = 0, \dots, N \tag{2.50h}$$

$$h_s\left(\mathbf{x}^0_{i|k}, \mathbf{x}^1_{i|k}\right) \leq \boldsymbol{\epsilon}_{i|k,3}, \qquad i = 0, \dots, N \tag{2.50i}$$

$$h_s\left(\mathbf{x}^0_{i|k}, \mathbf{x}^2_{i|k}\right) \leq \boldsymbol{\epsilon}_{i|k,4}, \qquad i = 0, \dots, N \tag{2.50j}$$

$$h_r\left(\mathbf{x}^0_{i|k}\right) \leq 0, \qquad i = 0, \dots, N \tag{2.50k}$$

$$\mathbf{x}^j_{0|k} = \mathbf{x}^j(k), \qquad j = 1, 2, 3. \tag{2.50l}$$

The motion planning problem is solved using nonlinear model predictive control. As such, the OCP (2.50) is solved in a receding horizon fashion: at each discrete time step denoted as k, the OCP is solved using numerical optimization. The solution to this OCP yields a piecewise-constant input sequence over the control horizon, the first input of this sequence is applied to the system and is held constant by means of a zero-order hold function. Subsequently, the states of all vehicles in the scenario evolve for one sample time according to the discretized system dynamics (2.26). Then, the OCP is reformulated and solved for the subsequent time step $k+1$. With this problem formulation in place, we proceed to devise a Gaussian process-based prediction model for the Follower in Chapter 3. Subsequently, we exploit this prediction model in several GP-based MPC algorithms, detailed in Chapter 4.

Chapter 3

Gaussian Process Prediction Model

Chapter 2 formalizes the motion planning problem and concludes with a finite horizon optimal control problem that is solved using model predictive control. As a baseline, we predict the future states of the Follower with a constant velocity model. However, such a prediction model does not always suffice, as seen in Chapter 5. Furthermore, active learning requires an interaction-aware prediction model. In order to better anticipate the Follower's behavior in the motion planning problem, we want to capture any deviations from this velocity, which we refer to as residual dynamics. These residual dynamics are approximated by a Gaussian process model that is updated online to learn the interactive behavior of the Follower.

Chapter 1 discusses the potential of Gaussian processes (GP) and their relevance in learning-based MPC. This chapter proposes a GP-based prediction model for the trajectory of the Follower. Firstly, a brief introduction to GPs is provided in section 3.1. Secondly, we derive an expression for the expected value of GP-based prediction model. Finally, we extend this model to a stochastic model that considers uncertainty in its predictions through the covariance of the predictions. The GP-based MPC exploits this covariance, which is detailed in Chapter 4. Throughout this chapter, we consider two variants of Gaussian process regression. Firstly, we consider the full Gaussian process in its classical form. Secondly, we consider a Sparse Pseudo-Input GP (SPGP), which aims to reduce the complexity of the Gaussian process regression.

3.1 Preliminaries

As just mentioned, we provide a brief introduction to the relevant preliminaries and nomenclature of Gaussian process regression that is used in this work. For a comprehensive explanation and more details on Gaussian processes, please refer to [35].

3.1.1 Stochastic Processes

A stochastic process is a mathematical object $d(\mathbf{z})$ that consists of an experiment with a probability measure $\Pr(d(\mathbf{z}))$ defined on a sample space $\mathcal{S}$ and a function that assigns a sample function $\hat{d}(\mathbf{z}, s)$ to each sample of $s \in \mathcal{S}$. These sample functions are a function of $\mathbf{z} \in \mathcal{Z}$, the input to the stochastic process, where $\mathcal{Z}$ denotes the input space of the stochastic process. The input to the process $\mathbf{z}$ can be, but is not necessarily, time. To this end, we can take a sample s from the stochastic process $d(\mathbf{z})$ and we obtain a probability on this sample function $\Pr\left(\hat{d}(\mathbf{z}, s)\right)$. Loosely speaking, we can think of a stochastic process as a generalization of a probability distribution of variables.

Gaussian Processes Gaussian processes are a specific class of stochastic processes. Just like Gaussian variables, whether scalar or vector-valued, are characterized by a value for its mean and covariance, Gaussian processes are characterized by a mean function and a covariance function. As a result, we have a Gaussian distribution over functions:

$$d(\mathbf{z}) \sim \mathcal{GP}\left(\bar{d}(\mathbf{z}), k\left(\mathbf{z}, \mathbf{z}'\right)\right), \tag{3.1}$$

where the mean function $\bar{d}(\mathbf{z})$ and the covariance function $k(\mathbf{z}, \mathbf{z}')$ of $d(\mathbf{z})$ are defined as:

$$\bar{d}(\mathbf{z}) = \mathbb{E}\left[d(\mathbf{z})\right] \tag{3.2a}$$

$$k\left(\mathbf{z}, \mathbf{z}'\right) = \mathbb{E}\left[\left(d(\mathbf{z}) - \bar{d}(\mathbf{z})\right)\left(d\left(\mathbf{z}'\right) - \bar{d}\left(\mathbf{z}'\right)\right)^{\top}\right], \tag{3.2b}$$

where $(\mathbf{z}, \mathbf{z}')$ is an input pair at which we evaluate the covariance function. In this study, we limit ourselves to a scalar output of a real Gaussian process, such that $d(\mathbf{z}) : \mathbb{R}^{n_z} \to \mathbb{R}$.

Prior Assumptions Let us consider an input vector $\mathbf{z} \in \mathbb{R}^{n_z}$ and a mapping $d(\mathbf{z})$ to an output $\mathbf{y} \in \mathbb{R}$. Assuming we are agnostic to the process $d(\mathbf{z})$, we assume a mean function of zero. We can draw random sample functions from this 'zero mean' *prior* distribution, i.e., in the absence of data or experience. To this end, we assume a Gaussian prior distribution based on our expert knowledge of the process $d(\mathbf{z})$:

$$d(\mathbf{z}) \sim \mathcal{N}\left(\mathbf{0}, k\left(\mathbf{z}, \mathbf{z}'\right)\right). \tag{3.3}$$

As we will see later, this does not limit the prediction capabilities of the GP. Moreover, using a deterministic mean function $\bar{d}(\mathbf{z})$ is trivial: we apply a *zero mean* GP to the difference between the observations and the fixed mean function $\bar{d}(\mathbf{z})$.

Bayesian Inference During *training*, we observe M pairs of inputs and outputs from the GP $(\mathbf{Z}_j, \mathbf{y}_j)$ and store them in a training set $\mathcal{D} = \{(\mathbf{Z}_j, \mathbf{y}_j) \mid j = 1, \ldots, M\}$. Given a new input $\mathbf{z}$ and this training set, or *experience*, we can improve our predictions of the output $\mathbf{y}$. For example, when the input $\mathbf{z}$ is similar to previous observations, we may expect that the output is similar to the output of the associated observations. On the other hand, when the

input $\mathbf{z}$ is very different from previous observations, we can still make a prediction, however, the covariance of this prediction is likely to be larger. Bayes' theorem is at the heart of this inference:

$$\Pr(d \mid \mathcal{D}) = \frac{\Pr(\mathcal{D} \mid d)\ \Pr(d)}{\Pr(\mathcal{D})}, \tag{3.4}$$

and states that our prediction of d given the observations $\mathcal{D}$ can be computed by multiplying the likelihood of the process d given observations $\mathcal{D}$ by our prior assumption on d and dividing it by the probability of our observations $\mathcal{D}$. Here, the likelihood of d given $\mathcal{D}$ which is designed by the user based on data or expert knowledge, is:

$$\mathcal{L}(d \mid \mathcal{D}) = \Pr(\mathcal{D} \mid d) \tag{3.5}$$

Gaussian processes are particularly attractive as the computations required for this inference and learning become relatively easy [35]. Given a *prior distribution* $\Pr(d)$ and a training set with data of previous observations $\Pr(\mathcal{D})$, we can compute a closed-form expression of the *posterior distribution* $\Pr(d \mid \mathcal{D})$, which is also a Gaussian process. This simply means that with a Gaussian prior assumption of the functions we expect, and some understanding of the process in the form of data, what probability distribution do we expect for the output function? Again, this posterior distribution is also a Gaussian process, and, therefore, it is fully characterized by a mean function $\mu^d(\mathbf{z})$ and a covariance function $\Sigma^d(\mathbf{z})$. The properties of this posterior distribution are determined by the likelihood $\Pr(\mathcal{D} \mid d)$ and can be designed through the kernel function of the prior distribution.

Kernel Functions In Gaussian processes, the similarity between inputs is measured by the covariance function. The covariance function can be lifted into a feature space by a kernel function $k(\mathbf{z}, \mathbf{z}') : \mathbb{R}^{n_z} \times \mathbb{R}^{n_z} \to \mathbb{R}$. Again, these kernel functions play a key role in the inference for the posterior distribution. The covariance function can be used to impose properties on the posterior distribution, such as stationarity and non-degenerateness, and it can be used to tune its variance and characteristic length-scale. For example, increasing the signal variance σ_d of the prior covariance function leads to a larger amplitude of the posterior mean, but also a greater posterior covariance. The length-scale L_s is a measure of the smoothness, or frequency content, of the posterior distribution.

Frequently used covariance functions, particularly in learning-based MPC, are the squared exponential kernel functions [8, 9, 16, 36], which is also known as a radial basis function:

$$k_{\mathbf{z}\mathbf{z}'} = k(\mathbf{z}, \mathbf{z}') = \sigma_d \exp\left(-\frac{1}{2}(\mathbf{z} - \mathbf{z}')^\top L_s^{-2}(\mathbf{z} - \mathbf{z}')\right), \tag{3.6}$$

where σ_d is the signal variance and L_s denotes the length-scale matrix, which is typically chosen to be diagonal, as we do here. The squared exponential kernel is non-degenerate, meaning it has infinite rank. Furthermore, this kernel function is very powerful for nonlinear function regression [36], and it is universal, meaning it can approximate continuous functions in compact spaces with arbitrary accuracy [37, 38]. Despite some experiments with other kernel functions, we limit ourselves to the squared exponential kernel functions. For a detailed discussion on kernel functions and their properties, refer to [35].

3.1.2 Full Gaussian Process

In the previous section, we introduced GPs and briefly discussed how they can be used for inference. In this section, we formalize GP regression. We consider a prior distribution of the GP with a zero mean function and a kernel function as defined in (3.6):

$$d\left(\mathbf{z}\right) \sim \mathcal{N}\left(\mathbf{0}, k_{\mathbf{zz}'}\right). \tag{3.7}$$

By definition, a GP consists of a collection of random variables, any finite number of which have a joint Gaussian distribution. We assume that the training outputs $\mathbf{y} \in \mathbb{R}^M$, and the prior distribution $d\left(\mathbf{z}\right)$ are jointly Gaussian distributed:

$$\begin{bmatrix} \mathbf{y} \\ d\left(\mathbf{z}\right) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K_{\mathbf{ZZ}} & \mathbf{k}_{\mathbf{Zz}} \\ \mathbf{k}_{\mathbf{zZ}} & k_{\mathbf{zz}} \end{bmatrix}\right), \tag{3.8}$$

where $K_{\mathbf{ZZ}} \in \mathbb{R}^{M \times M}$ is a Gramm matrix with kernel evaluations, with $\left[K_{\mathbf{ZZ}'}\right]_{jj'} = k\left(\mathbf{Z}_j, \mathbf{Z}'_{j'}\right)$. Furthermore, the vector $\mathbf{k}_{\mathbf{Zz}} \in \mathbb{R}^{M \times 1}$ is the concatenation of kernel functions evaluated at the test point $\mathbf{z}$ and the training set $\mathbf{Z} = \left[\mathbf{Z}_1, \ldots, \mathbf{Z}_M\right] \in \mathbb{R}^{n_z \times M}$, where $\left[\mathbf{k}_{\mathbf{Zz}}\right]_j = k\left(\mathbf{Z}_j, \mathbf{z}\right)$ and $\mathbf{k}_{\mathbf{Zz}}^\top = \mathbf{k}_{\mathbf{zZ}}$. Given a set of training data $\mathcal{D} = \{(\mathbf{Z}_j, \mathbf{y}_j) \mid j = 1, \ldots, M\}$, we condition the jointly Gaussian prior distribution on the observations $\mathbf{y}$ and obtain the posterior distribution:

$$\Pr\left(d \mid \mathbf{z}, \mathbf{Z}, \mathbf{y}\right) = \mathcal{N}\left(\mu^d\left(\mathbf{z}\right), \Sigma^d\left(\mathbf{z}, \mathbf{z}'\right)\right), \tag{3.9}$$

where the posterior mean and covariance function for a new test point $\mathbf{z}$ read [35]:

$$\mu^d\left(\mathbf{z}\right) = \mathbf{k}_{\mathbf{zZ}} K_{\mathbf{ZZ}}^{-1} \mathbf{y}, \tag{3.10a}$$

$$\Sigma^d\left(\mathbf{z}\right) = k_{\mathbf{zz}} - \mathbf{k}_{\mathbf{zZ}} K_{\mathbf{ZZ}}^{-1} \mathbf{k}_{\mathbf{Zz}}. \tag{3.10b}$$

Now we have a formal expression for the posterior distribution of the process $d\left(\mathbf{z}\right)$, based on some assumed prior (3.7) and training data in the form of $\mathcal{D} = \{\mathbf{Z}, \mathbf{y}\}$.

3.1.3 Sparse Pseudo-Input Gaussian Processes

Sparse Pseudo-Input Gaussian processes (SPGP) [25], also known as the Fully Independent Training Conditional approximation, can be used to approximate full GPs using a set of $\tilde{M}$ inducing points, or pseudo-inputs and pseudo-targets $\mathcal{D}_{\text{ind}} = \{(\mathbf{Z}_{\text{ind},j}, \mathbf{y}_{\text{ind},j}) \mid j = 1, \ldots, \tilde{M}\}$. By exploiting the properties of the GP, the SPGP reduces its computational complexity, with the number of pseudo-inputs being much smaller than the actual training points [25].

Marginalization of the Likelihood Firstly, the likelihood of the target data $\mathbf{Z}$ and the pseudo-set is modeled as a particular Gaussian process [25]:

$$\Pr\left(d \mid \mathbf{Z}, \mathbf{Z}_{\text{ind}}, \mathbf{y}_{\text{ind}},\right) = \mathcal{N}\left(K_{\mathbf{ZZ}_{\text{ind}}} K_{\mathbf{Z}_{\text{ind}}\mathbf{Z}_{\text{ind}}}^{-1} \mathbf{y}_{\text{ind}}, \Lambda\right), \tag{3.11}$$

where:

$$\Lambda = \operatorname{diag}(\boldsymbol{\lambda}), \tag{3.12}$$

$$[\boldsymbol{\lambda}]_j = k_{\mathbf{z}_j\mathbf{z}_j} - \mathbf{k}_{\mathbf{z}_j\mathbf{z}_{\text{ind}}} K^{-1}_{\mathbf{z}_{\text{ind}}\mathbf{z}_{\text{ind}}} \mathbf{k}_{\mathbf{z}_{\text{ind}}\mathbf{z}_j}, \quad \text{for } j = 1, \ldots, M. \tag{3.13}$$

Learning with this model involves finding a suitable set of pseudo-inputs $\mathbf{Z}_{\text{ind}}$ and pseudo-targets $\mathbf{y}_{\text{ind}}$. However, rather than maximizing the likelihood, the pseudo-targets $\mathbf{y}_{\text{ind}}$ can be integrated out by exploiting the marginalization property of Gaussian processes [25]. To this end, one can place a Gaussian prior on the pseudo-set:

$$\Pr(\mathbf{y}_{\text{ind}} \mid \mathbf{Z}_{\text{ind}}) = \mathcal{N}(\mathbf{0}, K_{\mathbf{Z}_{\text{ind}}}), \tag{3.14}$$

and find the posterior distribution over pseudo-targets $\mathbf{y}_{\text{ind}}$ using Bayes theorem (3.4) on the likelihood (3.11) and the prior (3.14):

$$\Pr(\mathbf{y}_{\text{ind}} \mid \mathcal{D}, \mathbf{Z}_{\text{ind}}) = \mathcal{N}\left(K_{\mathbf{Z}_{\text{ind}}\mathbf{Z}_{\text{ind}}} Q^{-1}_{\mathbf{Z}_{\text{ind}}} K_{\mathbf{Z}_{\text{ind}}\mathbf{Z}} \Lambda^{-1} \mathbf{y}, K_{\mathbf{Z}_{\text{ind}}\mathbf{Z}_{\text{ind}}} Q^{-1}_{\mathbf{Z}_{\text{ind}}} K_{\mathbf{Z}_{\text{ind}}\mathbf{Z}_{\text{ind}}}\right), \tag{3.15}$$

where:

$$Q_{\mathbf{Z}_{\text{ind}}} = K_{\mathbf{Z}_{\text{ind}}\mathbf{Z}_{\text{ind}}} + K_{\mathbf{Z}_{\text{ind}}\mathbf{Z}} \Lambda^{-1} K_{\mathbf{Z}\mathbf{Z}_{\text{ind}}}. \tag{3.16}$$

Given a new input $\mathbf{z}$, we obtain the posterior distribution of d by marginalization, through the integration of the likelihood of (3.11) with the posterior of (3.15) [25]:

$$\begin{aligned} \Pr(d \mid \mathbf{z}, \mathcal{D}, \mathbf{Z}_{\text{ind}}) &= \int \mathrm{d}\mathbf{y}_{\text{ind}} \Pr(d \mid \mathbf{z}, \mathbf{Z}_{\text{ind}}, \mathbf{y}_{\text{ind}}) \Pr(\mathbf{y}_{\text{ind}} \mid \mathcal{D}, \mathbf{Z}_{\text{ind}}) \\ &= \mathcal{N}\left(\tilde{\mu}^d(\mathbf{z}), \tilde{\Sigma}^d(\mathbf{z})\right), \end{aligned} \tag{3.17}$$

The expression for the posterior mean function and covariance function for a new test point $\mathbf{z}$ of the SPGP read [25]:

$$\tilde{\mu}^d(\mathbf{z}) = \mathbf{k}_{\mathbf{z}\mathbf{Z}_{\text{ind}}} Q^{-1}_{\mathbf{Z}_{\text{ind}}} K_{\mathbf{Z}_{\text{ind}}\mathbf{Z}} \Lambda^{-1} \mathbf{y} \tag{3.18a}$$

$$\tilde{\Sigma}^d(\mathbf{z}) = k_{\mathbf{z}\mathbf{z}} - \mathbf{k}_{\mathbf{z}\mathbf{Z}_{\text{ind}}} \left(K^{-1}_{\mathbf{Z}_{\text{ind}}\mathbf{Z}_{\text{ind}}} - Q^{-1}_{\mathbf{z}_{\text{ind}}}\right) \mathbf{k}_{\mathbf{Z}_{\text{ind}}\mathbf{z}}. \tag{3.18b}$$

Henceforth, we use a tilde $\left(\tilde{\mu}^d, \tilde{\Sigma}^d\right)$ to refer to the SPGP rather than the full GP. Through the marginalization of the GP on the pseudo-input set, the computational complexity is significantly reduced from $\mathcal{O}(M^3)$ to $\mathcal{O}\left(M\tilde{M}^2\right)$ for training, and from $\mathcal{O}(M^2)$ to $\mathcal{O}\left(\tilde{M}^2\right)$ per test point for inference [25]. For a detailed derivation, refer to [25].

Selecting the Inducing Points Selecting the inducing points can be done based on some information criterion. On the other hand, transductive methods exploit the location of the test point $\mathbf{z}$, rather than only relying on the training set [35], which is of interest for fast methods like MPC for motion planning, in particular in absence of training data. Hewing *et. al* [16] propose a method to dynamically select this set of inducing points as

the previous predicted state-input sequence of the MPC, as predictive control lends itself well to transductive approximations. Through the inducing points, the approximation can be adjusted based on the test points, i.e., the (future) evaluation locations. In MPC, an approximate trajectory through the state-input space is typically available which provides us with an estimate of these test points. As such, we select the inducing points heuristically along the approximate state and input trajectory which is computed at the previous MPC step [16].

3.2 Learned Dynamics

In this section, we exploit Gaussian processes to construct a data-driven prediction model for the Follower's future states over the prediction horizon. Although, GPs can account for Gaussian measurement noise on the observations, as mentioned in section 2.1, we assume that any measurements, and hence the training data, is free of any noise. The Follower is controlled by the Interactive Merge-Reactive IDM model, detailed in section 2.2.2. The resulting closed-loop dynamics are composed of nominal dynamics $\boldsymbol{f}^1$ which are assumed to be known, and of residual dynamics $\boldsymbol{g}^1$ which are assumed to be unknown:

$$\mathbf{x}_{k+1}^1 = \boldsymbol{f}^1\left(\mathbf{x}_k^1\right) + \boldsymbol{g}^1\left(\mathbf{x}_k^0, \mathbf{x}_k^1, \mathbf{u}_k^0, \mathbf{x}_k^2, \mathbf{u}_k^1, \mathbf{u}_k^2\right). \tag{3.19}$$

However, when the Ego vehicle has not observed any data, the Ego initially only has access to the Follower's current state. Accordingly, we have to make a prediction of the Follower's state using only its current state. To this end, the nominal dynamics $\boldsymbol{f}^1 : \mathbb{R}^n \to \mathbb{R}^n$ are a constant velocity model which is a homogeneous and non-autonomous system, as the current velocity is updated at every time step. Furthermore, it is independent of the states of the other vehicles. Conversely, the residual dynamics $\boldsymbol{g}^1 : \mathbb{R}^n \to \mathbb{R}^n$ may describe a policy that depends on the states as well as the inputs from the other agents. For example, the IDM uses the current acceleration of the reference vehicle as an input for its policy. The initial prediction, based on $\boldsymbol{f}^1$, is that the Follower maintains its current velocity at time step k over the entire prediction horizon.

The residual dynamics $\boldsymbol{g}^1$ are approximated using GP regression, which is introduced in section 3.1. More specifically, we use a GP to model the learned dynamics $d^1 : \mathbb{R}^{n_z} \to \mathbb{R}$:

$$\mathbf{x}_{i+1|k}^1 = \boldsymbol{f}^1\left(\mathbf{x}_{i|k}^1\right) + Bd^1\left(\mathbf{x}_{i|k}^0, \mathbf{x}_{i|k}^1, \mathbf{x}_{i|k}^2, \mathbf{u}_{i|k}^0\right), \tag{3.20a}$$

$$d^1 \sim \mathcal{N}\left(\mu^{d^1}, \Sigma^{d^1}\right). \tag{3.20b}$$

The learned dynamics are assumed to lie in the subspace spanned by a vector B, which we define later in (3.30). This prediction model is assumed to have access only to the current states of all vehicles and the Ego's input. Note that the learned dynamics are dependent on $\mathbf{x}^0$, $\mathbf{x}^1$, $\mathbf{x}^2$ and $\mathbf{u}$, making the Follower's prediction model *interaction-aware*, as a control input $\mathbf{u}$ of the Ego vehicle will influence the state $\mathbf{x}^1$, and the resulting state $\mathbf{x}^1$ may in turn influence the selection of control input $\mathbf{u}$. This coupling enables us to jointly optimize the

motion of the Ego and the Follower. Subsequently, we will define the nominal dynamics $\boldsymbol{f}^1$ and learned dynamics d^1 formally.

3.2.1 Deterministic Prediction Model

Our initial assumption is that the Follower maintains a constant velocity over the prediction horizon. As such, the GP prediction model employs a constant velocity as a deterministic mean function:

$$\bar{v}^1_{i|k} = v^1(k), \text{ for } i = 0, \dots, N. \tag{3.21}$$

Due to the kinematics of the kinematic bicycle model, we have:

$$\bar{X}^1_{i+1|k} = \bar{X}^1_{i|k} + T_s v^1(k), \text{ for } i = 0, \dots, N-1, \tag{3.22}$$

where $\bar{X}^1_{0|k} = X^1(k)$. Hence, the deterministic mean function of the prediction model is equal to the nominal dynamics in (2.48):

$$\bar{\mathbf{x}}^1_{i+1} = \boldsymbol{f}^1\left(\bar{\mathbf{x}}^1_i\right) = A\bar{\mathbf{x}}^1_i = \begin{bmatrix} 1 & 0 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \bar{\mathbf{x}}^1_i, \text{ for } i = 0, \dots, N-1, \tag{3.23}$$

where $\bar{\mathbf{x}}^1_{0|k} = \mathbf{x}^1(k)$. The differences from this deterministic mean, i.e. the residual dynamics, are approximated by the learned dynamics d^1 which is a Gaussian process. We employ a zero mean Gaussian process and learn the difference with respect to the constant velocity assumption. When we do not have any training data the Gaussian process has a zero mean function, and the prediction model assumes a constant velocity over the prediction horizon. Note that we do get an associated covariance which is equal to the covariance of the assumed prior distribution. Now, we use previous observations to predict deviations from this velocity. We confine ourselves to predicting the residual velocity.

Residual Dynamics Let us recall the state of the Follower:

$$\mathbf{x}^1_k = \begin{bmatrix} X^1_k \\ Y^1_k \\ v^1_k \\ \psi^1_k \\ \delta^1_k \end{bmatrix}. \tag{3.24}$$

We infer these predictions using the velocities of the different vehicles, as well as their relative positions:

$$\mathbf{z} = \begin{bmatrix} v^0 & v^1 & v^2 & \left(X^1 - X^0\right) & \left(X^1 - X^2\right) & \left(Y^1 - Y^0\right) \end{bmatrix}^\top, \tag{3.25}$$

here we omit the relative lateral position between the Follower and Leader for we assume this to be zero. To this end, GP regression is used to predict the velocity of the Follower at the next time step $k+1$. The output that we observe and aim to predict is the difference between the Follower's actual velocity and the deterministic constant velocity prediction:

$$y_k = v^1_{k+1} - v^1_k = \Delta v^1_k. \tag{3.26}$$

As such, we can store previous observations in a row vector $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M]^\top$ and use them for learning using GP regression. The deviation from the deterministic mean, or the velocity increment, is approximated by the posterior distribution of the Gaussian process:

$$\Delta v^1_{i|k} = v^1_{i+1|k} - v^1_{i|k} \approx d^1\left(\mathbf{z}_{i|k}\right) \mid \mathcal{D}, \mathbf{Z}_{\text{ind}}, \tag{3.27}$$

where the posterior GP reads:

$$d^1\left(\mathbf{z}\right) \sim \mathcal{GP}\left(\mu^{d^1}\left(\mathbf{z}_{i|k}\right), \Sigma^{d^1}\left(\mathbf{z}_{i|k}, \mathbf{z}'_{i|k}\right)\right), \tag{3.28}$$

with the posterior mean and covariance functions as defined in (3.10) and (3.18), for the full GP and the SPGP, respectively. The predicted velocity of the Follower equals:

$$v^1_{i+1|k} = v^1_{i|k} + \Delta v^1_{i|k}, \tag{3.29}$$

with $v_{0|k} = v(k)$. The residual dynamics d^1 are assumed to only affect the velocity of the Follower, as the relationship between the longitudinal position and velocity is kinematic. The input matrix B is used to project the learned dynamics d^1 onto the state $\mathbf{x}^1$:

$$B = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}^\top. \tag{3.30}$$

Hence, we obtain the following posterior of the prediction for the Follower:

$$\mathbf{x}^1_{i+1|k} = A\mathbf{x}^1_{i|k} + Bd^1\left(\mathbf{z}_{i|k}\right), \text{ for } i = 0, \ldots, N-1. \tag{3.31}$$

It should be noted, however, that d and the predicted test points $\mathbf{z}_{i|k}$ are random variables. In order to make multi-step predictions over the horizon, we will formulate a tractable deterministic approximation for the MPC in the subsequent sections.

Posterior Mean Function Multi-step predictions can be made by consecutively evaluating the GP at the posterior mean of the preceding prediction:

$$\mu^{\mathbf{x}^1}_{i+1|k} = A\mu^{\mathbf{x}^1}_{i|k} + B\mu^{d^1}\left(\mu^{\mathbf{z}}_{i|k}\right), \text{ for } i = 0, \ldots, N-1. \tag{3.32}$$

The posterior mean of the Gaussian process $\mu^d(\mathbf{z})$ is a function of the test point $\mathbf{z}$ and is evaluated at the mean of the test point $\mu^{\mathbf{z}}$:

$$\mu^{\mathbf{z}}_{i|k} = \begin{bmatrix} v^0_{i|k} & \mu^{v^1}_{i|k} & v^2_{i|k} & \left(\mu^{X^1}_{i|k} - X^0_{i|k}\right) & \left(\mu^{X^1}_{i|k} - X^2_{i|k}\right) & \left(\mu^{Y^1}_{i|k} - Y^0_{i|k}\right) \end{bmatrix}^\top, \tag{3.33}$$

where $\mu^{\mathbf{x}^1}_{0|k} = \mathbf{x}^1(k)$. The posterior mean function $\mu^{d^1}(\mathbf{z}_{i|k})$ of the GP reads as follows:

$$\mu^{d^1}\left(\mathbf{z}_{i|k}\right) = \mathbf{k}_{\mathbf{z}_{i|k}\mathbf{Z}} K^{-1}_{\mathbf{Z}\mathbf{Z}} \mathbf{y}, \tag{3.34}$$

for the full GP, as defined in (3.10). In case of the SPGP, the posterior mean function $\tilde{\mu}^{d^1}$ is defined in (3.18) and reads:

$$\tilde{\mu}^{d^1}(\mathbf{z}) = \mathbf{k}_{\mathbf{z}\mathbf{Z}_{\text{ind}}} Q^{-1}_{\mathbf{Z}_{\text{ind}}} K_{\mathbf{Z}_{\text{ind}}\mathbf{Z}} \Lambda^{-1} \mathbf{y}. \tag{3.35}$$

In the next section, we derive an expression for the covariance of the state $\Sigma^{\mathbf{x}^1}$.

3.3 Uncertainty Propagation

Since the predictions of the GP are random variables that are Gaussian distributed, the predicted state trajectories are characterized by a mean trajectory as well as a covariance at each prediction step. This covariance provides an approximate but qualitative measure of uncertainty that can be used in stochastic MPC to provide a degree of probabilistic constraint satisfaction. Consecutive GP evaluations result in a distribution over distributions which is generally intractable. In this section, we derive a tractable approximation of the distribution that enables uncertainty propagation over consecutive prediction steps. We employ a first-order Taylor approximation of the GP and propagate the uncertainty over the prediction horizon through a linear filter. We derive these notions for the full GP, as well as the extensions for the Sparse Pseudo-Input GP, in parallel.

3.3.1 Independence Assumption

Closed-form solutions for prediction models are typically limited to one-step-ahead predictions in the case of Gaussian distributions [15, 39]. When using GPs for a sequence of predictions, the output of the GP is an input for the inference at the next prediction step. As mentioned before, consecutive evaluations are generally intractable. Therefore, predicting the state distributions of a dynamical system over a horizon of length N using GPs, typically requires numerical approximation. While successive evaluations of the true system are highly correlated [39], established methods use successive approximate evaluations which are assumed to be independent at each time step k and neglect the fact that GPs describe a distribution over functions. The inferred posterior distribution on the trajectory given data $\mathcal{D} = \{(\mathbf{Z}_j, \mathbf{y}_j) \mid j = 1, \ldots, M\}$ can be expressed as:

$$\begin{bmatrix} d^1_1 \\ \vdots \\ d^1_N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu^{d^1}(\mathbf{z}_1) \\ \vdots \\ \mu^{d^1}(\mathbf{z}_N) \end{bmatrix}, \begin{bmatrix} \Sigma^{d^1}(\mathbf{z}_1, \mathbf{z}_1) & \ldots & \Sigma^{d^1}(\mathbf{z}_1, \mathbf{z}_N) \\ \vdots & \ddots & \vdots \\ \Sigma^{d^1}(\mathbf{z}_N, \mathbf{z}_1) & \ldots & \Sigma^{d}(\mathbf{z}_N, \mathbf{z}_N) \end{bmatrix} \right). \tag{3.36}$$

The common independence assumption can be understood as approximating the distribution of the predicted state trajectory over N time steps as [39]:

$$\begin{bmatrix} d_1^1 \\ \vdots \\ d_N^1 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu^{d^1}(\mathbf{z}_1) \\ \vdots \\ \mu^{d^1}(\mathbf{z}_N) \end{bmatrix}, \begin{bmatrix} \Sigma^{d^1}(\mathbf{z}_1, \mathbf{z}_1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \Sigma^{d^1}(\mathbf{z}_N, \mathbf{z}_N) \end{bmatrix} \right). \tag{3.37}$$

3.3.2 Approximation of Covariance

In order to derive an expression for the covariance of the predicted state $\Sigma_i^{\mathbf{x}^1}$, we make some assumptions. Firstly, the predicted Follower's state $\mathbf{x}_i^1$ and the posterior of the GP d^1 of the prediction model are approximated as jointly Gaussian distributed at every time step:

$$\begin{bmatrix} \mathbf{x}_i^1 \\ d_i^1 \end{bmatrix} \sim \mathcal{N}(\mu_i, \Sigma_i) = \mathcal{N}\left(\begin{bmatrix} \mu_i^{\mathbf{x}^1} \\ \mu_i^{d^1} \end{bmatrix}, \begin{bmatrix} \Sigma_i^{\mathbf{x}^1} & \Sigma_i^{\mathbf{x}^1 d^1} \\ \Sigma_i^{d^1 \mathbf{x}^1} & \Sigma_i^{d^1} \end{bmatrix} \right), \tag{3.38}$$

such that the covariance can be approximated and propagated over the prediction horizon. Secondly, we assume that $\Sigma^{\mathbf{x}^0} = 0$, $\Sigma^{\mathbf{x}^2} = 0$ and $\Sigma^{\mathbf{u}} = 0$. Finally, for the uncertainty propagation of the Gaussian process, we need to find an approximation of $\mu_i^d, \Sigma_i^d, \Sigma_i^{\mathbf{x}^1 d^1}$. To this end, we make a first-order Taylor approximation of the Gaussian process. For details of this derivation please refer to [40]. This method provides a good trade-off between approximation accuracy and computational complexity [16]:

$$\mu_i^{d^1} = \mu^{d^1}(\mu_i^{\mathbf{z}}) \tag{3.39a}$$

$$\begin{bmatrix} \Sigma_i^{\mathbf{x}^1 d^1} \\ \Sigma_i^{d^1} \end{bmatrix} = \begin{bmatrix} \Sigma_i^{\mathbf{x}^1} \left(\nabla_{\mathbf{x}^1} \mu^{d^1}(\mu_i^{\mathbf{z}}) \right)^\top \\ \Sigma^{d^1}(\mu_i^{\mathbf{z}}) + \nabla_{\mathbf{x}^1} \mu^{d^1}(\mu_i^{\mathbf{z}}) \Sigma_i^{\mathbf{x}^1} \left(\nabla_{\mathbf{x}^1} \mu^{d^1}(\mu_i^{\mathbf{z}}) \right)^\top \end{bmatrix}, \tag{3.39b}$$

where the posterior mean function $\mu^{d^1}(\mathbf{z})$ and covariance $\Sigma^{d^1}(\mathbf{z})$ function of the GP are defined per (3.10) and (3.18) for the full and sparse GP, respectively. For the first-order Taylor approximation of the Gaussian process, we need an expression of the gradient of the posterior mean, with respect to the state $\mathbf{x}^1$. This expression is specific to our prediction model and is derived in the next section.

3.3.3 Gradient of Posterior Mean

Full Gaussian Process Firstly, we derive an expression for the gradient of the posterior mean of the full Gaussian process with respect to the Follower's state:

$$\begin{aligned} \nabla_{\mathbf{x}^1} \mu^{d^1}(\mathbf{z}) &= \nabla_{\mathbf{x}^1} \mu^{d^1}(\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2) \in \mathbb{R}^{1 \times n_{\mathbf{x}}} \\ &\overset{(3.10)}{=} \nabla_{\mathbf{x}^1} \left(\mathbf{k}_{\mathbf{z}\mathbf{Z}} K_{\mathbf{Z}\mathbf{Z}}^{-1} \mathbf{y} \right) \\ &= \begin{bmatrix} \frac{\partial}{\partial \mathbf{x}_1^1} \mathbf{k}_{\mathbf{z}\mathbf{Z}} K_{\mathbf{Z}\mathbf{Z}}^{-1} \mathbf{y} & \frac{\partial}{\partial \mathbf{x}_2^1} \mathbf{k}_{\mathbf{z}\mathbf{Z}} K_{\mathbf{Z}\mathbf{Z}}^{-1} \mathbf{y} & \dots & \frac{\partial}{\partial \mathbf{x}_{n_{\mathbf{x}}}^1} \mathbf{k}_{\mathbf{z}\mathbf{Z}} K_{\mathbf{Z}\mathbf{Z}}^{-1} \mathbf{y} \end{bmatrix}. \end{aligned} \tag{3.40}$$

Since the Gramm matrix with kernel evaluations $K_{\mathbf{ZZ}} \in \mathbb{R}^{M\times M}$ and the vector of observed targets $\mathbf{y} \in \mathbb{R}^{M\times 1}$ are not a function of $\mathbf{x}^1$, we separate them as follows:

$$\nabla_{\mathbf{x}^1}\mu^{d^1}(\mathbf{z}) = \mathbf{y}^\top \left(K_{\mathbf{ZZ}}^{-1}\right)^\top \frac{\partial}{\partial \mathbf{x}^1}\mathbf{k}_{\mathbf{zZ}}^\top = \mathbf{y}^\top K_{\mathbf{ZZ}}^{-1}\frac{\partial}{\partial \mathbf{x}^1}\mathbf{k}_{\mathbf{Zz}}. \tag{3.41}$$

We compute the gradient of the squared exponential kernel function evaluated at one training point $\mathbf{Z}_j$:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{x}^1}k(\mathbf{Z}_j,\mathbf{z}) &= \frac{\partial}{\partial \mathbf{x}^1}\exp\left(-\frac{1}{2}(\mathbf{Z}_j-\mathbf{z})^\top L_s^{-2}(\mathbf{Z}_j-\mathbf{z})\right)\\
&= k(\mathbf{Z}_j,\mathbf{z})\frac{\partial}{\partial \mathbf{x}^1}\left(-\frac{1}{2}(\mathbf{Z}_j-\mathbf{z})^\top L_s^{-2}(\mathbf{Z}_j-\mathbf{z})\right)\\
&= k(\mathbf{Z}_j,\mathbf{z})\left[\frac{\mathbf{Z}_{j,4}-X^1+X^0}{L_{s,4}^2}+\frac{\mathbf{Z}_{j,5}-X^1+X^2}{L_{s,5}^2} \quad \frac{\mathbf{Z}_{j,6}-Y^1+Y^0}{L_{s,6}^2} \quad \frac{\mathbf{Z}_{j,2}-v^1}{L_{s,2}^2} \quad 0 \quad 0\right]\\
&= k(\mathbf{Z}_j,\mathbf{z})\left[\frac{\mathbf{Z}_{j,4}-\mathbf{z}_4}{L_{s,4}^2}+\frac{\mathbf{Z}_{j,5}-\mathbf{z}_5}{L_{s,5}^2} \quad \frac{\mathbf{Z}_{j,6}-\mathbf{z}_6}{L_{s,6}^2} \quad \frac{\mathbf{Z}_{j,2}-\mathbf{z}_2}{L_{s,2}^2} \quad 0 \quad 0\right],
\end{aligned} \tag{3.42}$$

where $\mathbf{Z}_{j,i}$ is the i^{th} element of data point $\mathbf{Z}_j$, and L_s is a positive, diagonal length-scale matrix and $L_{s,i}$ is the i^{th} diagonal element of L_s. This can simply be extended to compute the gradient of the vector of the kernel evaluated at the set of training points:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{x}^1}\mathbf{k}_{\mathbf{Zz}} &= \begin{bmatrix}\frac{\partial}{\partial \mathbf{x}_1^1}\mathbf{k}_{\mathbf{Zz}} & \frac{\partial}{\partial \mathbf{x}_2^1}\mathbf{k}_{\mathbf{Zz}} & \dots & \frac{\partial}{\partial \mathbf{x}_{n_\mathbf{x}}^1}\mathbf{k}_{\mathbf{Zz}}\end{bmatrix}\\
&= \begin{bmatrix}
\frac{\partial}{\partial \mathbf{x}_1^1}k(\mathbf{Z}_1,\mathbf{z}) & \frac{\partial}{\partial \mathbf{x}_2^1}k(\mathbf{Z}_1,\mathbf{z}) & \dots & \frac{\partial}{\partial \mathbf{x}_{n_\mathbf{x}}^1}k(\mathbf{Z}_1,\mathbf{z})\\
\frac{\partial}{\partial \mathbf{x}_1^1}k(\mathbf{Z}_2,\mathbf{z}) & \frac{\partial}{\partial \mathbf{x}_2^1}k(\mathbf{Z}_2,\mathbf{z}) & \dots & \frac{\partial}{\partial \mathbf{x}_{n_\mathbf{x}}^1}k(\mathbf{Z}_2,\mathbf{z})\\
\vdots & & \ddots & \vdots\\
\frac{\partial}{\partial \mathbf{x}_1^1}k(\mathbf{Z}_M,\mathbf{z}) & \frac{\partial}{\partial \mathbf{x}_2^1}k(\mathbf{Z}_M,\mathbf{z}) & \dots & \frac{\partial}{\partial \mathbf{x}_{n_\mathbf{x}}^1}k(\mathbf{Z}_M,\mathbf{z})
\end{bmatrix} \in \mathbb{R}^{M\times n_\mathbf{x}}\\
&= \mathbf{k}_{\mathbf{Zz}}\begin{bmatrix}1 & \dots & 1\end{bmatrix}\begin{bmatrix}
\frac{\mathbf{Z}_{1,4}-\mathbf{z}_4}{L_{s,4}^2}+\frac{\mathbf{Z}_{1,5}-\mathbf{z}_5}{L_{s,5}^2} & \frac{\mathbf{Z}_{1,6}-\mathbf{z}_6}{L_{s,6}^2} & \frac{\mathbf{Z}_{1,2}-\mathbf{z}_2}{L_{s,2}^2} & 0 & 0\\
\frac{\mathbf{Z}_{2,4}-\mathbf{z}_4}{L_{s,4}^2}+\frac{\mathbf{Z}_{2,5}-\mathbf{z}_5}{L_{s,5}^2} & \frac{\mathbf{Z}_{2,6}-\mathbf{z}_6}{L_{s,6}^2} & \frac{\mathbf{Z}_{2,2}-\mathbf{z}_2}{L_{s,2}^2} & 0 & 0\\
\vdots & \vdots & \vdots & \vdots & \vdots\\
\frac{\mathbf{Z}_{M,4}-\mathbf{z}_4}{L_{s,4}^2}+\frac{\mathbf{Z}_{M,5}-\mathbf{z}_5}{L_{s,5}^2} & \frac{\mathbf{Z}_{M,6}-\mathbf{z}_6}{L_{s,6}^2} & \frac{\mathbf{Z}_{M,2}-\mathbf{z}_2}{L_{s,2}^2} & 0 & 0
\end{bmatrix},
\end{aligned} \tag{3.43}$$

which is equal to the vertical concatenation of the gradient of the kernel function evaluated at the individual training points in (3.42). Note that $\frac{\partial}{\partial \mathbf{x}_i^1}$ denotes the partial derivative with respect to the i-th element of the state $\mathbf{x}^1$ and should not be confused with the i-th prediction, and $n_\mathbf{x} = 5$ is the size of the state vector. Finally, we obtain an expression for

the gradient of the posterior mean:

$$\nabla_{\mathbf{x}^1}\mu^{d^1}(\mathbf{z}) = \mathbf{y}^\top K_{\mathbf{ZZ}}^{-1}\mathbf{k}_{\mathbf{Z}\mathbf{z}}\begin{bmatrix}1 & \dots & 1\end{bmatrix}\begin{bmatrix} \frac{\mathbf{Z}_{1,4}-\mathbf{z}_4}{L_{s,4}^2}+\frac{\mathbf{Z}_{1,5}-\mathbf{z}_5}{L_{s,5}^2} & \frac{\mathbf{Z}_{1,6}-\mathbf{z}_6}{L_{s,6}^2} & \frac{\mathbf{Z}_{1,2}-\mathbf{z}_2}{L_{s,2}^2} & 0 & 0 \\ \frac{\mathbf{Z}_{2,4}-\mathbf{z}_4}{L_{s,4}^2}+\frac{\mathbf{Z}_{2,5}-\mathbf{z}_5}{L_{s,5}^2} & \frac{\mathbf{Z}_{2,6}-\mathbf{z}_6}{L_{s,6}^2} & \frac{\mathbf{Z}_{2,2}-\mathbf{z}_2}{L_{s,2}^2} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\mathbf{Z}_{M,4}-\mathbf{z}_4}{L_{s,4}^2}+\frac{\mathbf{Z}_{M,5}-\mathbf{z}_5}{L_{s,5}^2} & \frac{\mathbf{Z}_{M,6}-\mathbf{z}_6}{L_{s,6}^2} & \frac{\mathbf{Z}_{M,2}-\mathbf{z}_2}{L_{s,2}^2} & 0 & 0 \end{bmatrix}. \tag{3.44}$$

Sparse Pseudo-Input Gaussian Process Note that the gradient of the posterior mean with respect to the Follower's state $\mathbf{x}^1$ of the SPGP, is similar to that of the full GP as $\mathbf{k}_{\mathbf{z}\mathbf{Z}_{\text{ind}}}$ is the only function that is dependent on $\mathbf{x}^1$. Adaptations to the expression of the posterior mean and covariance functions (3.10) are independent of $\mathbf{x}^1$ and, hence, do not affect the gradient:

$$\begin{aligned}\nabla_{\mathbf{x}^1}\tilde{\mu}^{d^1}(z) &= \nabla_{\mathbf{x}^1}\tilde{\mu}^{d^1}\left(\mathbf{x}^0,\mathbf{x}^1,\mathbf{x}^2\right) \\ &\overset{(3.18)}{=} \nabla_{\mathbf{x}^1}\left(\mathbf{k}_{\mathbf{z}\mathbf{Z}_{\text{ind}}}Q_{\mathbf{Z}_{\text{ind}}}^{-1}K_{\mathbf{Z}_{\text{ind}}\mathbf{Z}}\Lambda^{-1}\mathbf{y}\right) \\ &= \mathbf{y}^\top\left(Q_{\mathbf{Z}_{\text{ind}}}^{-1}K_{\mathbf{Z}_{\text{ind}}\mathbf{Z}}\Lambda^{-1}\right)^\top\frac{\partial}{\partial\mathbf{x}^1}\mathbf{k}_{\mathbf{Z}_{\text{ind}}\mathbf{z}} \\ &= \mathbf{y}^\top\Lambda^{-1}K_{\mathbf{Z}\mathbf{Z}_{\text{ind}}}\left(Q_{\mathbf{Z}_{\text{ind}}}^{-1}\right)^\top\frac{\partial}{\partial\mathbf{x}^1}\mathbf{k}_{\mathbf{Z}_{\text{ind}}\mathbf{z}}.\end{aligned} \tag{3.45}$$

We merely have to change the argument for the vector of kernel functions $\mathbf{k}_{\mathbf{Z}_{\text{ind}}\mathbf{z}}$ as well as its dimensions from M to $\tilde{M}$:

$$\frac{\partial}{\partial\mathbf{x}^1}\mathbf{k}_{\mathbf{Z}_{\text{ind}}\mathbf{z}} = \mathbf{k}_{\mathbf{Z}_{\text{ind}}\mathbf{z}}\begin{bmatrix}1 & \dots & 1\end{bmatrix}\begin{bmatrix} \frac{\mathbf{Z}_{\text{ind}1,4}-\mathbf{z}_4}{L_{s,4}^2}+\frac{\mathbf{Z}_{\text{ind}1,5}-\mathbf{z}_5}{L_{s,5}^2} & \frac{\mathbf{Z}_{\text{ind}1,6}-\mathbf{z}_6}{L_{s,6}^2} & \frac{\mathbf{Z}_{\text{ind}1,2}-\mathbf{z}_2}{L_{s,2}^2} & 0 & 0 \\ \frac{\mathbf{Z}_{\text{ind}2,4}-\mathbf{z}_4}{L_{s,4}^2}+\frac{\mathbf{Z}_{\text{ind}2,5}-\mathbf{z}_5}{L_{s,5}^2} & \frac{\mathbf{Z}_{\text{ind}2,6}-\mathbf{z}_6}{L_{s,6}^2} & \frac{\mathbf{Z}_{\text{ind}2,2}-\mathbf{z}_2}{L_{s,2}^2} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\mathbf{Z}_{\text{ind}\tilde{M},4}-\mathbf{z}_4}{L_{s,4}^2}+\frac{\mathbf{Z}_{\text{ind}\tilde{M},5}-\mathbf{z}_5}{L_{s,5}^2} & \frac{\mathbf{Z}_{\text{ind}\tilde{M},6}-\mathbf{z}_6}{L_{s,6}^2} & \frac{\mathbf{Z}_{\text{ind}\tilde{M},2}-\mathbf{z}_2}{L_{s,2}^2} & 0 & 0 \end{bmatrix}. \tag{3.46}$$

Now that we have an expression for the gradient of the posterior mean, we can compute the covariance of the joint distribution of the state $\mathbf{x}^1$ and the learned dynamics d^1:

$$\Sigma_i = \begin{bmatrix}\Sigma_i^{\mathbf{x}^1} & \Sigma_i^{\mathbf{x}^1 d^1} \\ \Sigma_i^{d^1\mathbf{x}^1} & \Sigma_i^{d^1}\end{bmatrix}. \tag{3.47}$$

This allows us to make multi-step predictions with a tractable approximation of the GP that allows for the propagation of the uncertainty over the prediction horizon. Subsequently, we formalize the uncertainty propagation with multi-step predictions and construct the stochastic GP-based prediction model.

3.3.4 Stochastic Prediction Model

Full Gaussian Process The full GP is used to formulate a stochastic prediction model for the Follower, which is characterized by the expected value of the Follower's state $\mu_{i|k}^{\mathbf{x}^1}$ and the covariance of its state $\Sigma_{i|k}^{\mathbf{x}^1}$ at time step $k+i$. Similar to Kalman filtering, we propagate the uncertainty through the prediction model (3.31), including the nominal dynamics:

$$\mu_{i+1|k}^{\mathbf{x}^1} = A\mu_{i|k}^{\mathbf{x}^1} + B\mu_{i|k}^{d^1}, \tag{3.48a}$$

$$\Sigma_{i+1|k}^{\mathbf{x}} = \begin{bmatrix} A & B \end{bmatrix} \Sigma_{i|k} \begin{bmatrix} A & B \end{bmatrix}^\top, \quad \text{for } i = 0, \dots, N-1, \tag{3.48b}$$

where the initial prediction equal the current state $\mu_{0|k}^{\mathbf{x}^1} = \mathbf{x}^1(k)$ with full certainty $\Sigma^{\mathbf{x}^1} = 0$. Furthermore, the covariance matrix Σ_i propagates through the variance of the predicted state of the following vehicle $\Sigma_i^{\mathbf{x}^1}$ at prediction step i over a horizon of $i = 0, \dots, N$. The consecutive prediction steps are assumed to be independent and are evaluated according to (3.37). Hence, we have a tractable approximation of the stochastic prediction model (3.31) of the Follower that can be used in the MPC problem.

Sparse Pseudo-Input Gaussian Process The SPGP prediction model is similar to that of (3.48), where the expressions for the posterior mean and covariance function are replaced by those in (3.18):

$$\tilde{\mu}_{i+1}^{\mathbf{x}^1} = A\tilde{\mu}_i^{\mathbf{x}^1} + B\tilde{\mu}_i^{d^1} \tag{3.49a}$$

$$\tilde{\Sigma}_{i+1}^{\mathbf{x}} = \begin{bmatrix} A & B \end{bmatrix} \tilde{\Sigma}_i \begin{bmatrix} A & B \end{bmatrix}^\top, \quad \text{for } i = 0, \dots, N-1. \tag{3.49b}$$

Chance Constraints As mentioned before, one of the main advantages of GPs is their measure of uncertainty that is quantified by the covariance. Recall that the collision avoidance constraints, defined in (2.41), can be expanded to account for this uncertainty:

$$\begin{aligned} h_e\left(\mathbf{x}_{i|k}^0, \mathbf{x}_{i|k}^1, \Sigma_{i|k}^{X^1}, \sigma\right) &= -\frac{\left(c_{x,i|k}^1 - c_{x,i|k}^0\right)^2}{\left(\mathcal{E}_{c,A} + \sigma\sqrt{\Sigma_{i|k}^{X^1}}\right)^2} - \frac{\left(c_{y,i|k}^1 - c_{y,i|k}^0\right)^2}{\mathcal{E}_{c,B}^2} + 1 \\ h_e\left(\mathbf{x}_{i|k}^0, \mathbf{x}_{i|k}^1, \Sigma_{i|k}^{X^1}, \sigma\right) &\leq 0, \quad \text{for } i = 0, \dots, N, \end{aligned} \tag{3.50}$$

where the posterior covariance from the GP $\Sigma_{i|k}^d$ can be propagated to the covariance of the longitudinal position of the Follower $\Sigma_{i|k}^{X^1}$. The covariance of the longitudinal position is the first element of the covariance matrix of the joint distribution in (3.38):

$$\Sigma^{X^1} = [\Sigma^{\mathbf{x}^1}]_{1,1} = [\Sigma]_{1,1}. \tag{3.51}$$

Hence, we can explicitly incorporate the predicted uncertainty from the GP-based prediction model in the collision avoidance constraints of the MPC. By tuning the parameter $\sigma \geq 0$

in (2.41) we can adjust the number of standard deviations that we account for in our probabilistic constraints (2.40). Although this covariance is approximate, for the prediction model is merely an approximation, we can still exploit it as a tunable degree of conservatism.

3.3.5 Validation of the Sparse Pseudo-Input Gaussian Process

The Sparse-Pseudo Input GP was initially designed as an approximation of the full GP [25]. Hence, one could argue that the SPGP should be compared against the full GP to validate if the SPGP is capable of approximating the full GP with sufficient accuracy for its intended application. However, one does not simply validate the SPGP by comparing it with the full GP.

Firstly, it is difficult to validate the sparse GP simply by comparing the SPGP with the full GP, as the full GP is only an approximation of the Follower's policy. Secondly, its computational complexity limits the full GP-based prediction model in its prediction horizon and the size of the training set. This limits its capabilities to learn from new data and improve its predictions, and hence, the relevance of the full GP model. For this reason, most works on GP-MPC employ sparse GPs. Similarly, our SPGP prediction model is designed with the intention of constructing a GP-based prediction model that lends itself to a larger-scale implementation. In conclusion, in this study, we consider both methods as a means to approximate the residual dynamics of the Following vehicle.

3.4 Gaussian Process MPC

In this chapter, two Gaussian process-based prediction models are presented to predict the future states of the Follower. On the one hand, we have a full GP-based prediction model that is purely conditioned on the training data $\mathcal{D}$. On the other hand, we have a Sparse Pseudo-Input GP-based prediction model that is additionally conditioned on a set of pseudo-inputs, which can be determined using the previous state-input trajectory from the MPC [16]. The SPGP reduces the complexity of the prediction model, making it more tractable for implementation in an online MPC algorithm.

With these GP-based prediction models, we have expressions for the mean and covariance of the state of the Follower over the prediction horizon which can be employed in our interaction-aware learning MPC problem. Firstly, by expanding the chance constraints of (2.41). Secondly, by exploiting the covariance for active learning. Chapter 4 will expand on the baseline MPC, as well as different MPC algorithms that employ the GP-based prediction models, presented in this chapter, including an MPC-based active learning algorithm.

Chapter 4

Learning-based GP-MPC

In this study, we consider three MPC methods to plan the motion of the Ego vehicle in a lane merging scenario. Chapter 2 casts this motion planning problem into an optimal control problem. In this problem, the Ego vehicle has to consider the Leading target vehicle, as well as the Following target vehicle. The Leader maintains a constant velocity throughout the scenario. The Follower interacts with the Leader as well as the Ego vehicle according to a novel interactive, merge-reactive IDM, which is introduced in Chapter 2. As a baseline, we first use a constant velocity MPC (CV-MPC) which assumes that the Follower maintains its current velocity, presented in section 4.1. Secondly, section 4.2 proposes a Gaussian process-based MPC (GP-MPC) algorithm that uses the interactive GP-based prediction model, detailed in Chapter 3, to predict the velocity of the Follower. Finally, we expand this GP-MPC algorithm by employing an active learning framework by [13], in section 4.3.

4.1 MPC with a Constant Velocity Model

Firstly, we discuss the baseline MPC that assumes a constant velocity over the prediction horizon for the Leader and the Follower. We employ the objective function defined in (2.29):

$$\begin{aligned} J\left(\mathbf{x}(k), \mathbf{u}(k-1), \mathbf{U}_k\right) = & \left\|\mathbf{x}_{N|k} - \mathbf{x}_k^r\right\|_P^2 + \left(Y_{N|k} - W_l\right)^2 P_Y \left(Y_{N|k} - m\left(X_{N|k}\right)\right)^2 \\ & + \sum_{i=0}^{N-1} \left\|\mathbf{x}_{i|k} - \mathbf{x}_k^r\right\|_Q^2 + \left(Y_{i|k} - W_l\right)^2 Q_Y \left(Y_{i|k} - m\left(X_{i|k}\right)\right)^2 \\ & + \left\|\mathbf{u}_{i|k}\right\|_R^2 + \left\|\boldsymbol{\Delta}\mathbf{u}_{i|k}\right\|_S^2, \end{aligned} \tag{4.1}$$

where $\mathbf{x}^r$ denotes the reference of the state (2.27), W_l is the lane width, and $m(X)$ is a function that describes the center of the merge lane, defined in (2.28). Moreover, we employ a set of constraints to delimit the solution of the MPC to an attainable and safe trajectory. The construction of the objective function and constraints are detailed in section 2.3. The weights of the objective function are selected as follows: $Q = \text{diag}\,(0, 0, 10, 200, 100)$, $Q_Y = 100$, $R = \text{diag}\,(10, 500)$, $S = \text{diag}\,(100, 10000)$, $P = Q$, $P_Y = Q_Y$. Note that the angles have large costs for they are relatively small quantities.

Considering the complexity of the problem, the recursive feasibility of the MPC is not considered in this thesis. The collision avoidance constraints (4.2g)-(4.2j) are softly constrained such that active constraints can be relaxed to restore the feasibility of the MPC problem, in the case of infeasibility. To this end, we use an l_1 penalty function [41], which is a non-smooth regularization method. Each of the constraints (4.2g)-(4.2j) can be relaxed by their individual non-negative slack variable $\boldsymbol{\epsilon}_{k,j} \in \mathbb{R}^{N+1}_{\geq 0}$ for $j = 1, \dots 4$ with a linear penalty. Subsequently, we formulate the constant velocity MPC (CV-MPC) problem as follows:

$$\min_{\mathbf{U}_k, \boldsymbol{\epsilon}_k} J\left(\mathbf{x}^0\left(k\right), \mathbf{u}\left(k-1\right), \mathbf{U}_k\right) + \mathbf{1}^\top \boldsymbol{\epsilon}_k \boldsymbol{\rho} \tag{4.2a}$$

$$\text{s.t. } \mathbf{x}^0_{i+1|k} = f^0\left(\mathbf{x}^0_{i|k}, \mathbf{u}_{i|k}\right), \qquad i = 0, \dots, N-1 \tag{4.2b}$$

$$\mathbf{x}^1_{i+1|k} = A\mathbf{x}^1_{i|k}, \qquad i = 0, \dots, N-1 \tag{4.2c}$$

$$\mathbf{x}^2_{i+1|k} = A\mathbf{x}^2_{i|k}, \qquad i = 0, \dots, N-1 \tag{4.2d}$$

$$\mathbf{x}^0_{min} \leq \mathbf{x}^0_{i|k} \leq \mathbf{x}^0_{max}, \qquad i = 0, \dots, N \tag{4.2e}$$

$$\mathbf{u}_{min} \leq \mathbf{u}_{i|k} \leq \mathbf{u}_{max}, \qquad i = 0, \dots, N-1 \tag{4.2f}$$

$$h_c\left(\mathbf{x}^0_{i|k}, \mathbf{x}^1_{i|k}\right) \leq \boldsymbol{\epsilon}_{i|k,1}, \qquad i = 0, \dots, N \tag{4.2g}$$

$$h_c\left(\mathbf{x}^0_{i|k}, \mathbf{x}^2_{i|k}\right) \leq \boldsymbol{\epsilon}_{i|k,2}, \qquad i = 0, \dots, N \tag{4.2h}$$

$$h_s\left(\mathbf{x}^0_{i|k}, \mathbf{x}^1_{i|k}\right) \leq \boldsymbol{\epsilon}_{i|k,3}, \qquad i = 0, \dots, N \tag{4.2i}$$

$$h_s\left(\mathbf{x}^0_{i|k}, \mathbf{x}^2_{i|k}\right) \leq \boldsymbol{\epsilon}_{i|k,4}, \qquad i = 0, \dots, N \tag{4.2j}$$

$$h_r\left(\mathbf{x}^0_{i|k}\right) \leq 0, \qquad i = 0, \dots, N \tag{4.2k}$$

$$\boldsymbol{\epsilon}_{i|k} \geq 0, \qquad i = 0, \dots, N \tag{4.2l}$$

$$\mathbf{x}^j_{0|k} = \mathbf{x}^j\left(k\right), \qquad j = 1, 2, 3. \tag{4.2m}$$

where $\boldsymbol{\epsilon}_k \in \mathbb{R}^{(N+1)\times 4}_{\geq 0}$ denotes the concatenation the slack variable. To limit the complexity of the MPC, each of the collision avoidance constraints can be relaxed by a slack variable over the entire prediction horizon. The safety collision avoidance constraints (4.2g) and (4.2h) should only be relaxed if the problem is infeasible. Conversely, the social constraints (4.2i) and (4.2j) ideally would be satisfied, but are not critical for safety. The l_1 penalty method is exact, provided that the penalty weights are sufficiently large. However, the problem can become ill-conditioned for too large values of ρ [41].

The performance of the MPC strongly depends on the penalty of these slack variables. Selecting a too low penalty can negatively affect the solution by allowing the solver to converge to a relaxed local optimum. Selecting a too high penalty can lead to an ill-conditioned problem that has poor convergence. Hence, to have a fair comparison of all methods, we select the penalty of the slack variables that retains feasibility in all methods, such that all methods can be evaluated with the same parameterization. In addition, in

section 5.5, we perform some alternative studies to assess the effect of the penalty of the slack variables. The penalty weights of the slack variables are tuned *ad hoc*. To this end, the nominal penalty weights are selected as follows:

$$\rho = \begin{bmatrix} 10^5 & 10^5 & 10^3 & 10^3 \end{bmatrix}^\top . \tag{4.3}$$

Constant Velocity MPC We apply the first element of the resulting locally optimal control sequence to (4.2) to the system (2.26), provided that the problem is feasible. Subsequently, we compute the state transition of all agents for one time step. In case the problem is infeasible, the shifted last feasible solution is employed. We assume that the problem is feasible at time step $k = 0$. This process is summarized in Algorithm 1 and serves as a baseline for the succeeding, more advanced MPC algorithms.

Algorithm 1 Constant Velocity MPC

1: **for** each time step k = 0, 1, 2, . . . , **do**
2: Solve CV-MPC problem (4.2) for $\mathbf{U}_k$
3: **if** Problem (4.2) is feasible **then**
4: Apply $\mathbf{u}(k) \leftarrow \mathbf{u}_{0|k}$ of the solution to (4.2) to the system
5: **else**
6: Apply shifted input $\mathbf{u}(k) \leftarrow \mathbf{u}_{1|k-1}$ of the last feasible solution to (4.2) to the system
7: **end if**
8: **end for**

4.2 Passive Learning with GP-MPC

We extend the prediction model for the Following vehicle to improve its predictions using online and offline training. To this end, we can use observations from previous similar driving scenarios to train a GP-based prediction model, detailed in Chapter 3. As motivated in Chapter 3, we confine ourselves to Sparse Pseudo-Input GPs. In this section, we limit ourselves to passive learning, i.e., we do not change the objective function. Subsequently, we explore active learning methods that relax the primary performance objective and actively look for control inputs that explore the state space and seek to improve the MPC its prediction model.

4.2.1 Gaussian Process MPC

The GP-MPC algorithms employ the GP-based prediction model, presented in Chapter 3, to learn the residual dynamics of the Follower in order to better anticipate the behavior of the Follower. The expected values of the consecutive state predictions of the Follower are determined by the mean function of the posterior of the GP. Furthermore, the covariance of the GP-based prediction model is exploited in the collision avoidance constraints. To this

end, we evaluate the collision avoidance constraints at the expected value of the state of the Follower. Additionally, we expand the collision avoidance ellipses with the covariance of the longitudinal position of the Follower.

In order to exploit the GP-based prediction model, we change the constraint that governs the system dynamics (4.2c). Here, we add the posterior mean of the learned dynamics d^1 to the nominal prediction model. As mentioned before, we confine ourselves to the SPGP prediction model. The posterior mean of the SPGP prediction model reads (3.18):

$$\tilde{\mu}^{d^1}\left(\mathbf{z}_{i|k}\right) = \mathbf{k}_{\mathbf{z}_{i|k}\mathbf{Z}_{\text{ind}}} Q_{\mathbf{Z}_{\text{ind}}}^{-1} K_{\mathbf{Z}_{\text{ind}}\mathbf{Z}} \Lambda^{-1} \mathbf{y}. \tag{4.4}$$

Furthermore, we can exploit the covariance of the predicted state of the Follower $\tilde{\Sigma}_{i|k}^{\mathbf{x}^1}$ to expand the collision avoidance constraint for the Follower (2.41). The joint covariance of the state of the Follower and the Gaussian process is propagated as follows:

$$\tilde{\Sigma}_{i+1|k}^{\mathbf{x}^1} = \begin{bmatrix} A & B \end{bmatrix} \tilde{\Sigma}_{i|k} \begin{bmatrix} A & B \end{bmatrix}^\top, \quad \text{for } i = 0, \ldots, N-1, \tag{4.5}$$

where the joint covariance matrix of the state of the Follower and the Gaussian process is:

$$\tilde{\Sigma}_i = \begin{bmatrix} \tilde{\Sigma}_i^{\mathbf{x}^1} & \tilde{\Sigma}_i^{\mathbf{x}^1 d^1} \\ \tilde{\Sigma}_i^{d^1 \mathbf{x}^1} & \tilde{\Sigma}_i^{d^1} \end{bmatrix}. \tag{4.6}$$

For a detailed derivation, please refer to Chapter 3.

Stochastic GP-MPC Although the prediction model and its associated probability distribution are approximate, the stochastic GP-MPC can exploit the uncertainty of the prediction model by expanding the ellipse of collision avoidance constraint formulated in (2.41). The major axis of the collision avoidance ellipse is expanded by σ standard deviations of the longitudinal position of the Follower:

$$\mathcal{E}_{c,A} + \sigma \sqrt{\tilde{\Sigma}^{X^1}}, \tag{4.7}$$

where the longitudinal covariance is equal to the first element of the covariance matrix of the state:

$$\tilde{\Sigma}^{X^1} = [\tilde{\Sigma}^{\mathbf{x}^1}]_{1,1} = [\tilde{\Sigma}]_{1,1}. \tag{4.8}$$

In order to limit excessive conservatism, the softly constrained *social ellipse* (4.9j), (4.9k) is not expanded. As argued in section 2.3.2, the expanded safety ellipse already introduces extra conservatism due to the eccentricity of the ellipse, hence, accounting for uncertainty in the social ellipse would introduce even more conservatism. Moreover, one could argue that human drivers simply keep a social distance to account for uncertainty. Recent trends in cooperative driving show potential to drive with minimal headway, actually, as a result of reduced uncertainty. Perhaps, such stochastic formulations replace the need for explicit social driving as it could follow implicitly from the problem formulation.

Apart from these adaptations, we solve the same finite horizon OCP as with the CV-MPC (4.2), employing the same l_1 relaxation. The adaptations of the prediction model yield the Gaussian process MPC (GP-MPC) problem:

$$\min_{\mathbf{U}_k, \boldsymbol{\epsilon}_k} J\left(\mathbf{x}^0(k), \mathbf{u}(k-1), \mathbf{U}_k\right) + \mathbf{1}^\top \boldsymbol{\epsilon}_k \boldsymbol{\rho} \tag{4.9a}$$

$$\text{s.t. } \mathbf{x}^0_{i+1|k} = f^0\left(\mathbf{x}^0_{i|k}, \mathbf{u}_{i|k}\right), \qquad i = 0, \ldots, N-1 \tag{4.9b}$$

$$\tilde{\mu}^{\mathbf{x}^1}_{i+1|k} = A\tilde{\mu}^{\mathbf{x}^1}_{i|k} + B\tilde{\mu}^{d^1}\left(\tilde{\mu}^{\mathbf{z}}_{i|k}\right), \qquad i = 0, \ldots, N-1 \tag{4.9c}$$

$$\tilde{\Sigma}^{\mathbf{x}}_{i+1|k} = \begin{bmatrix} A & B \end{bmatrix} \tilde{\Sigma}_{i|k} \begin{bmatrix} A & B \end{bmatrix}^\top, \qquad i = 0, \ldots, N-1 \tag{4.9d}$$

$$\mathbf{x}^2_{i+1|k} = A\mathbf{x}^2_{i|k}, \qquad i = 0, \ldots, N-1 \tag{4.9e}$$

$$\mathbf{x}^0_{min} \leq \mathbf{x}^0_{i|k} \leq \mathbf{x}^0_{max}, \qquad i = 0, \ldots, N \tag{4.9f}$$

$$\mathbf{u}_{min} \leq \mathbf{u}_{i|k} \leq \mathbf{u}_{max}, \qquad i = 0, \ldots, N-1 \tag{4.9g}$$

$$h_e\left(\mathbf{x}^0_{i|k}, \mu^{\mathbf{x}^1}_{i|k}, \tilde{\Sigma}^{X^1}_{i|k}, \sigma\right) \leq \boldsymbol{\epsilon}_{i|k,1}, \qquad i = 0, \ldots, N \tag{4.9h}$$

$$h_c\left(\mathbf{x}^0_{i|k}, \mathbf{x}^2_{i|k}\right) \leq \boldsymbol{\epsilon}_{i|k,2}, \qquad i = 0, \ldots, N \tag{4.9i}$$

$$h_s\left(\mathbf{x}^0_{i|k}, \tilde{\mu}^{\mathbf{x}^1}_{i|k}\right) \leq \boldsymbol{\epsilon}_{i|k,3}, \qquad i = 0, \ldots, N \tag{4.9j}$$

$$h_s\left(\mathbf{x}^0_{i|k}, \mathbf{x}^2_{i|k}\right) \leq \boldsymbol{\epsilon}_{i|k,4}, \qquad i = 0, \ldots, N \tag{4.9k}$$

$$h_r\left(\mathbf{x}^0_{i|k}\right) \leq 0, \qquad i = 0, \ldots, N \tag{4.9l}$$

$$\boldsymbol{\epsilon}_{i|k} \geq 0, \qquad i = 0, \ldots, N \tag{4.9m}$$

$$\mathbf{x}^j_{0|k} = \mathbf{x}^j(k), \qquad j = 1, 2, 3, \tag{4.9n}$$

$$\tilde{\Sigma}^{\mathbf{x}^1}_{0|k} = \mathbf{0}. \tag{4.9o}$$

We employ the same parameterization of the l_1 penalty function for the slack variables $\boldsymbol{\epsilon}_k \in \mathbb{R}^{(N+1)\times 4}_{\geq 0}$ as in (4.2):

$$\rho = \begin{bmatrix} 10^5 & 10^5 & 10^3 & 10^3 \end{bmatrix}^\top. \tag{4.10}$$

The stochastic collision avoidance constraints (4.9h) are enforced with $\sigma = 2$ standard deviations of the approximative predicted covariance of the position of the Follower. Again, the objective function is defined in (2.29). The weights of the objective function are selected as follows: $Q = \text{diag}(0, 0, 10, 200, 100)$, $Q_Y = 100$, $R = \text{diag}(10, 500)$, $S = \text{diag}(100, 10000)$, $P = Q$, $P_Y = Q_Y$. Like the constant velocity MPC, this problem is solved in a receding horizon fashion. Subsequently, we construct an algorithm for the SPGP-MPC problem (4.9) to accommodate passive online learning.

4.2.2 Passive Learning-based GP-MPC

The prediction quality and the covariance can be improved online as the training set can be extended online with current observations. As mentioned before, the GP can exploit

Algorithm 2 Passive Learning GP-based MPC

for each time step k = 0, 1, 2, ..., **do**
  Solve MPC problem (4.9) for $\mathbf{U}_k$
  if Problem (4.9) is feasible **then**
    Apply $\mathbf{u}(k) \leftarrow \mathbf{u}_{0|k}$ of the solution to (4.9) to the system
  else
    Apply shifted input $\mathbf{u}(k) \leftarrow \mathbf{u}_{1|k-1}$ of the last feasible solution to (4.9) to the system
  end if
  Compute training point $\mathbf{z}_k$ and training target $\mathbf{y}_k$
  Update training set $D_{k+1} \leftarrow \mathcal{D}_k \cup (\mathbf{z}_k, \mathbf{y})$
  if Sparse Pseudo-Input GP prediction model is used **then**
    Update the set of inducing points to $\tilde{M}$ uniformly sampled points from the state trajectory of the solution to (4.9)
  end if
end for

observations from previous scenarios through an initial training set $\mathcal{D}_0$. Moreover, the GP can exploit observations from the previous time step $k-1$ by extending the training set to $\mathcal{D}_k$. As such, we use these observations to improve our predictions and *learn* online. The predictions are updated through the mean function, but also by adapting the covariance.

Firstly, the primary MPC problem (4.9) is solved using the sparse pseudo-input GP-based prediction model, detailed in (3.49). The hyperparameters of the GP are tuned *ad hoc*, as the commonly employed log-likelihood optimization [35] was unsuccessful. To this end, the length-scale matrix is selected as $L_s = \text{diag}(10, 10, 10, 10, 10, 5)$ and the variance of the prior of the GP is selected as $\sigma_d = 0.3$. Again, we apply the first element of the resulting locally optimal control sequence to the system, provided that the problem is feasible. We assume also here that the problem is feasible at the first time step. Then, we compute the state transition of all agents for one time step. Subsequently, this state transition is used to calculate the residual of the prediction model:

$$\mathbf{y}_k = v_{k+1}^1 - v_k^1. \tag{4.11}$$

The inputs to the GP are simply a combination of the states:

$$\mathbf{z}_k = \begin{bmatrix} v_k^0 & v_k^1 & v_k^2 & \left(X_k^1 - X_k^0\right) & \left(X_k^1 - X_k^2\right) & \left(Y_k^1 - Y_k^0\right) \end{bmatrix}^\top. \tag{4.12}$$

These inputs and the residual are added to the training set $\mathcal{D}$ as a new training pair $(\mathbf{z}_k, \mathbf{y}_k)$. Conclusively, the passive learning GP-based MPC algorithm is detailed in Algorithm 2.

4.3 Active Learning with GP-MPC

The dual control paradigm actively seeks control inputs that improve our understanding of the system and, as a result, improve the overall performance. Rather than focusing on

maximizing the control performance with a nominal (prediction) model of the system, we allow a sacrifice in performance to improve the nominal (prediction) model that allows us to improve the overall performance [12]. Furthermore, the dual control paradigm enables exploration of the state space that could improve safety and feasibility. To this end, we integrate our GP-MPC in an active learning framework for MPC proposed by [13].

4.3.1 Active Learning Framework

Soloperto *et al.* [13] present a general method to augment existing MPC algorithms with active learning through a generic user-defined learning objective function. This framework uses a generic learning objective function $H(\cdot)$ while the primary objective function $J(\cdot)$, defined in (4.1) appears in additional constraints:

$$\min_{\mathbf{U}_k, \Delta J_k} \; H\left(\mathbf{x}^0(k), \mathbf{x}^1(k), \mathbf{x}^2(k), \mathbf{U}_k\right) \tag{4.13a}$$

$$\text{s.t. } J\left(\mathbf{x}^0(k), \mathbf{u}(k-1), \mathbf{U}_k\right) = J_k^B + \Delta_k^J, \tag{4.13b}$$

$$\Delta_k^J \leq \bar{\beta} \max\{J_k^+, 0\} + \bar{\gamma} + \Gamma_{k-1}, \tag{4.13c}$$

$$\Delta_k^J \leq \beta^{\max} \max\{J_k^+, 0\} + \gamma^{\max}, \tag{4.13d}$$

$$\mathbf{x}_{i|k}^0 \in \mathbb{X}_{i|k}^0, \qquad i = 0, \ldots, N, \tag{4.13e}$$

$$\mathbf{u}_{i|k} \in \mathbb{U}_{i|k}, \qquad i = 0, \ldots, N-1, \tag{4.13f}$$

$$\mathbf{x}_{0|k}^j = \mathbf{x}^j(k), \qquad j = 1, 2, 3. \tag{4.13g}$$

For the sake of simplicity, the nominal state and input constraints are written in a shorthand notation, where $\mathbb{X}$ and $\mathbb{U}$ denote the subspace of feasible states and inputs, respectively. The components of the active learning framework [13] are detailed, below.

Learning Objective With GP-MPC, the active learning framework [13] can be used to seek state-input trajectories which maximize uncertainty/covariance [13]. To this end, we take the learning objective as the negative sum of the covariance function of the GP:

$$H\left(\mathbf{x}^0(k), \mathbf{x}^1(k), \mathbf{x}^2(k), \mathbf{U}_k\right) = -\sum_{i=0}^{N} \tilde{\Sigma}^{d^1}\left(\tilde{\mu}_i^{\mathbf{z}}\right). \tag{4.14}$$

This quantity is closely related to the covariance of the state of the Follower $\tilde{\Sigma}^{\mathbf{x}^1}$, however, it is more tractable as an objective function than the covariance of the state.

Relaxation of Primary Cost The primary objective function is composed of the desired primary objective J_k^B and the relaxation variable ΔJ_k, which is an additional optimization variable:

$$J\left(\mathbf{x}^0(k), \mathbf{u}(k-1), \mathbf{U}_k\right) = J_k^B + \Delta_k^J. \tag{4.15}$$

This relaxation variable represents how much the primary objective function $J(\cdot)$ is allowed to deviate from its desired value J_k^B. The desired value J_k^B is the solution to the primary

MPC problem (4.9). This relaxation can be bounded by two constraints. Firstly, by an *average* constraint that uses a storage Γ_k, using notions from economic MPC [13]:

$$\Delta_k^J \leq \bar{\beta} \max\{J_k^+, 0\} + \bar{\gamma} + \Gamma_{k-1}. \tag{4.16}$$

The variable J_k^+ is defined as follows:

$$J_k^+ := \hat{J}_{k-1} - J_k^B, \tag{4.17}$$

where $\hat{J}_{k-1}$ is the primary cost of the solution of the previous MPC. The max operation is used to ensure that performance relaxation is allowed only if it is possible to guarantee a potential decrease of the primary objective function, i.e., $J_k^+ \geq 0$. The storage Γ_k is defined as follows:

$$\Gamma_k := \Gamma_{k-1} + \bar{\beta} \max\{J_k^+, 0\} + \bar{\gamma} - \Delta_k^J, \tag{4.18}$$

with $\Gamma_0 \geq 0$. The parameter $\bar{\gamma} \geq 0$ represents the average absolute relaxation of the primary objective, whereas $\bar{\beta} \in [0, 1)$ represents the relative average relaxation of the primary objective that we allow for learning or exploration. In addition, the relaxation can be bounded by a hard (non-averaged) bound:

$$\Delta_k^J \leq \beta^{\max} \max\{J_k^+, 0\} + \gamma^{\max}, \tag{4.19}$$

where $\beta^{\max} \geq 0$ and $\gamma^{\max} \geq 0$ have an analogous meaning to $\bar{\beta}$ and $\bar{\gamma}$, respectively [13].

Feasibility and Performance The construction of our primary objective function poses some limitations. The value of the MPC its objective function is very low until the Ego has to deviate from its reference, that is, as the Ego vehicle approaches the merge point. As such, it is very difficult to enforce an average, let alone absolute, decrease of the objective function over the horizon. It is important to note that even though the primary problem (4.9) remains feasible, the learning problem that is to be solved subsequently is infeasible as it cannot satisfy the (average) cost reduction.

Furthermore, [13] provides performance bounds based on the assumption that the primary MPC problem is recursively feasible and has a suitable performance bound. However, as the focus of this study is on the comparison between the constant velocity prediction model and the GP-based prediction model, no attempt is made to satisfy this performance bound. Hence, the performance bounds and recursive feasibility from [13] do not transfer and we do not exploit this active learning framework to its maximum capabilities. Yet, by only relying on $\gamma_{\max}$ or $\bar{\gamma}$ we can still explore the state space, an initial attempt is made to investigate the potential of GP-based active learning using this framework. Then it is trivial that (4.20) is feasible with $\Delta J_k^* = 0$, provided that (4.9) is feasible.

4.3.2 Active Learning with GP-MPC

The proposed active learning GP-MPC will allow for a fixed amount of performance degradation/relaxation of $J(\cdot)$ to minimize the learning objective $H(\cdot)$. Casting the primary

MPC problem (4.9) into this framework yields the active learning MPC problem:

$$\begin{aligned}
\min_{\mathbf{U}_k, \boldsymbol{\epsilon}_k, \Delta J_k} \quad & H\left(\mathbf{x}^0(k), \mathbf{x}^1(k), \mathbf{x}^2(k), \mathbf{U}_k\right) + \mathbf{1}^\top \boldsymbol{\epsilon}_k \boldsymbol{\rho} && && \text{(4.20a)}\\
\text{s.t.} \quad & J\left(\mathbf{x}^0(k), \mathbf{u}(k-1), \mathbf{U}_k\right) = J_k^B + \Delta_k^J, && && \text{(4.20b)}\\
& \Delta_k^J \leq \bar{\beta} \max\{J_k^+, 0\} + \bar{\gamma} + \Gamma_{k-1}, && && \text{(4.20c)}\\
& \Delta_k^J \leq \beta^{\max} \max\{J_k^+, 0\} + \gamma^{\max}, && && \text{(4.20d)}\\
& \mathbf{x}^0_{i+1|k} = f^0\left(\mathbf{x}^0_{i|k}, \mathbf{u}_{i|k}\right), && i = 0, \ldots, N-1 && \text{(4.20e)}\\
& \mu^{\mathbf{x}^1}_{i+1|k} = A\mu^{\mathbf{x}^1}_{i|k} + B\mu^{d^1}\left(\mu^{\mathbf{z}}_{i|k}\right), && i = 0, \ldots, N-1 && \text{(4.20f)}\\
& \mathbf{x}^2_{i+1|k} = A\mathbf{x}^2_{i|k}, && i = 0, \ldots, N-1 && \text{(4.20g)}\\
& \mathbf{x}^0_{min} \leq \mathbf{x}^0_{i|k} \leq \mathbf{x}^0_{max}, && i = 0, \ldots, N, && \text{(4.20h)}\\
& \mathbf{u}_{min} \leq \mathbf{u}_{i|k} \leq \mathbf{u}_{max}, && i = 0, \ldots, N-1, && \text{(4.20i)}\\
& h_e\left(\mathbf{x}^0_{i|k}, \mu^{\mathbf{x}^1}_{i|k}, \Sigma^{\mathbf{x}^1}_{i|k}, \sigma\right) \leq \boldsymbol{\epsilon}_{i|k,1}, && i = 0, \ldots, N, && \text{(4.20j)}\\
& h_c\left(\mathbf{x}^0_{i|k}, \mathbf{x}^2_{i|k}\right) \leq \boldsymbol{\epsilon}_{i|k,2}, && i = 0, \ldots, N, && \text{(4.20k)}\\
& h_s\left(\mathbf{x}^0_{i|k}, \mu^{\mathbf{x}^1}_{i|k}\right) \leq \boldsymbol{\epsilon}_{i|k,3}, && i = 0, \ldots, N, && \text{(4.20l)}\\
& h_s\left(\mathbf{x}^0_{i|k}, \mathbf{x}^2_{i|k}\right) \leq \boldsymbol{\epsilon}_{i|k,4}, && i = 0, \ldots, N, && \text{(4.20m)}\\
& h_r\left(\mathbf{x}^0_{i|k}\right) \leq 0, && i = 0, \ldots, N, && \text{(4.20n)}\\
& \boldsymbol{\epsilon}_{i|k} \geq 0, && i = 0, \ldots, N, && \text{(4.20o)}\\
& \mathbf{x}^j_{0|k} = \mathbf{x}^j(k), && j = 1, 2, 3, && \text{(4.20p)}\\
& \Sigma^{\mathbf{x}^1}_{0|k} = \mathbf{0}, && && \text{(4.20q)}
\end{aligned}$$

where $H(\cdot)$ is the learning cost function as defined in (4.14), and $J(\cdot)$ is the primary cost, defined in (4.1). The weights of the primary objective function are selected as follows: $Q = \text{diag}(0, 0, 10, 200, 100)$, $Q_Y = 100$, $R = \text{diag}(10, 500)$, $S = \text{diag}(100, 10000)$, $P = Q$, $P_Y = Q_Y$. Also here, we employ the same l_1 penalty function on the slack variables as in (4.2). As the learning objective is much smaller than the primary objective, we scale the weight of the slack variables ρ of the l_1 penalty function accordingly:

$$\rho = \begin{bmatrix} 10 & 10 & 0.1 & 0.1 \end{bmatrix}^\top. \tag{4.21}$$

Active Learning Parameterization The hyperparameters $\bar{\beta}$, $\beta_{\max}$, $\bar{\gamma}$ and $\gamma_{\max}$ of the active learning framework can be used to adjust its behavior. The active learning framework [13] can be used to enforce the (average) decrease of the objective function over time. As such, by tuning $\bar{\beta}$ and $\beta_{\max}$ we can enforce a certain (average) cost reduction over time, through (4.16) and (4.19), respectively.

During simulation studies, we find that the learning problem becomes infeasible when we such an (average) cost reduction over time. As mentioned before, we cannot satisfy these learning constraints using $\bar{\beta}$ or $\beta_{\max}$, as it is very difficult to enforce an absolute or average decrease of the objective function over the horizon due to the current problem formulation. Note that the primary problem remains feasible, however, the learning problem that is to be solved subsequently becomes infeasible as it cannot satisfy the (average) cost reduction. In conclusion, due to the current construction of the MPC's objective function, the active learning framework by [13] could not be exploited to its full potential.

We can still exploit the framework to actively perturb the Follower and explore the state space. However, this exploration is parameterized by the fixed relaxation γ_{max}, and the exploration is not guaranteed to improve the performance. The hyperparameters $\bar{\beta}$, $\beta_{\max}$, $\bar{\gamma}$ and $\gamma_{\max}$ of the active learning framework are tuned *ad hoc* and are selected as follows: $\bar{\beta} = 0$, $\beta_{\max} = 0$, $\bar{\gamma} = \infty$ and $\gamma_{\max} = 100$.

The Active Learning GP-MPC Algorithm Note that the active learning GP-MPC relies on a desired primary cost J_k^B. Soloperto *et. al* [13] propose three expressions for J_k^B that satisfy these assumptions. The desired performance bound can be defined by additionally solving the primary MPC scheme [13], formulated in (4.9). However, this requires solving two MPC problems. Considering we do not satisfy their assumptions, this option is the simplest and sufficient for a proof of concept. We assume that both problems are feasible at the first time step. The active learning GP-MPC is formalized in Algorithm 3.

4.3.3 Active Learning with SPGP-MPC

Oscillatory Input Behavior During the simulation studies, discussed in section 5.4, we find that the active learning MPC from Algorithm 3 with the SPGP prediction model causes an oscillatory input trajectory. In an online learning scheme with a receding horizon, new training data is obtained at every time step. We expect that this constant updating of the training data caused these oscillations, however, updating the training set every K steps does not rectify this oscillatory behavior.

Although these oscillations could be reduced by penalizing changes in the input, this merely limits the exploration but does not resolve the underlying problem. It is important to note that the inducing points of the SPGP are updated every time step. Consequently, the posterior distribution will be conditioned on the previous state trajectory. Furthermore, the active learning GP-MPC will tend toward a state trajectory with maximum covariance, namely one that deviates from the previous state trajectory. As a result, the active learning GP-MPC keeps switching between input sequences, and, due to the receding horizon of the MPC, leads to this oscillatory input behavior. Such inputs are very undesired in a human-interfacing application, like that of autonomous driving. Moreover, the MPC is unable to adequately explore the state space as it lacks a form of *dedication*.

Buisson-Fenet *et al.* [19] propose an alternative method for active learning GP-dynamics to MPC, namely a plan-and-apply algorithm, where an input sequence to an N-step finite

Algorithm 3 Active Learning with GP-MPC

for Each time step k = 0, 1, 2, ..., **do**
  Solve MPC problem (4.9) for $\mathbf{U}_k$
  if Problem (4.9) is feasible **then**
    Update the desired primary cost $J_k^B \leftarrow J(\mathbf{U}_k)$ from the solution to (4.9)
  else
    Update the desired primary cost $J_k^B \leftarrow J(\mathbf{U}_{k-1})$ from the last feasible solution to (4.9)
  end if
  Solve active learning MPC problem (4.20) for $\mathbf{U}_k$
  if Problem (4.20) is feasible **then**
    Apply $\mathbf{u}(k) \leftarrow \mathbf{u}_{0|k}$ of the solution to (4.20) to the system
  else
    Apply shifted input $\mathbf{u}(k) \leftarrow \mathbf{u}_{1|k-1}$ of the last feasible solution to (4.20) to the system
  end if
  Compute training point $\mathbf{z}_k$ and training target $\mathbf{y}_k$
  Update training set $D_{k+1} \leftarrow \mathcal{D}_k \cup (\mathbf{z}_k, \mathbf{y_k})$
  if Sparse Pseudo-Input GP prediction model is used **then**
    Update the set of inducing points to $\tilde{M}$ uniformly sampled points from the state trajectory of the solution to (4.9)
  end if
end for

OCP is computed and rolled out for N time steps. This generally leads to better coverage of the state space than a greedy version where $N = 1$ [19]. However, a drawback of this method is that one loses the receding horizon properties. This could affect performance and safety, in particular, in multi-agent systems with large uncertainty and varying conditions. Subsequently, we propose an alternative algorithm to rectify the oscillatory input behavior that arises in active learning with Sparse Pseudo-Input GP-based MPC.

Learning Period We propose an alternative active learning algorithm that mitigates the switching behavior of the Active Learning with GP-MPC (Algorithm 3) while maintaining the powerful receding horizon from MPC, which allows it to cope with changing conditions. In addition to a prediction horizon of length N, we propose a *learning period* of length K.

The Active Learning with SPGP-MPC algorithm, detailed in Algorithm 4, solves the OCP with a prediction horizon of N with a receding horizon, however, it only updates the pseudo-inputs every K time steps. This leads to less frequent switching of input sequences and rectifies the oscillations in the input trajectories. Consequently, the MPC has more *dedication* and sticks to one learning input sequence. When this becomes too costly, it is controlled to another area of the state space.

Algorithm 4 Active Learning with SPGP

for Each time step k = 0, 1, 2, ..., **do**
  Solve MPC problem (4.9) for $\mathbf{U}_k$
  if Problem (4.9) is feasible **then**
    Update the desired primary cost $J_k^B \leftarrow J(\mathbf{U}_k)$ from the solution to (4.9)
  else
    Update the desired primary cost $J_k^B \leftarrow J(\mathbf{U}_{k-1})$ from the last feasible solution to (4.9)
  end if
  Solve active learning MPC problem (4.20) for $\mathbf{U}_k$
  if Problem (4.20) is feasible **then**
    Apply $\mathbf{u}(k) \leftarrow \mathbf{u}_{0|k}$ of the solution to (4.20) to the system
  else
    Apply shifted input $\mathbf{u}(k) \leftarrow \mathbf{u}_{1|k-1}$ of the last feasible solution to (4.20) to the system
  end if
  Compute training point $\mathbf{Z}_k$ and training target $\mathbf{y}_k$
  Update training set $D_{k+1} \leftarrow \mathcal{D}_k \cup (\mathbf{z}_k, \mathbf{y_k})$
  if k mod K = 0 **then**
    Update the set of inducing points to $\tilde{M}$ uniformly sampled points from the state trajectory of the solution to (4.9)
  end if
end for

The MPC problems discussed so far, are posed as non-convex nonlinear programs that can be computationally hard to solve. The next section expands on symbolic and numerical methods and considerations that are used to solve these nonlinear programming problems.

4.4 Numerical Optimization

The MPC algorithms are solved numerically using the automatic differentiation toolbox CasADi [42] in MATLAB. This toolbox offers a lot flexibility in the formulation of the problem, e.g. with the GP-based prediction model. Furthermore, it supports state-of-the-art solvers used for nonlinear optimization, such as the Sequential Quadratic Programming method and Interior Point Optimization (IPOPT) [29]. Specifically, CasADi's [42] Opti stack is used to formulate and solve the nonlinear program with the solver IPOPT [29]. The internal line search of IPOPT is performed by the linear solver MA57 by HSL [43].

4.4.1 Parametrization Method

The motion planning problem is solved through a direct optimal control parametrization, as discussed in section 2.2.3. Direct single-shooting and multiple-shooting are investigated.

These problems are equivalent as they rely on the same integration method [7], namely Runge-Kutta 4 (2.26). The multiple-shooting method is found to be considerably faster, in particular when the optimization is warmstarted with the shift initialization.

Shift Initialization After the first iteration, at time step $k = 1$, the nonlinear program is warmstarted using a shifted predicted state and input trajectory to initialize the nonlinear program. Firstly, the input sequence is shifted to correct for the advancement of time, the last input is held constant such that:

$$\mathbf{u}_{N-1|k_{\text{init}}} = \mathbf{u}^*_{N-1|k-1}, \tag{4.22}$$

and the input sequence is initialized as:

$$\mathbf{U}_{k_{\text{init}}} = \left(\mathbf{u}^*_{1|k-1}, \mathbf{u}^*_{2|k-1}, ..., \mathbf{u}^*_{N-1|k-1}, \mathbf{u}^*_{N-1|k-1}\right), \tag{4.23}$$

where $\mathbf{u}^*_{i|k}$ denotes the locally optimal input at prediction step i at time step k. Furthermore, the shifted predicted state trajectory can be computed as:

$$\mathbf{X}^0_{k_{\text{init}}} = \left(\mathbf{x}^{0*}_0, \mathbf{x}^{0*}_{2|k-1}, ..., \mathbf{x}^{0*}_{N|k-1}, \boldsymbol{f}^0\left(\mathbf{x}^{0*}_{N|k-1}, \mathbf{u}^*_{N-1|k-1}\right)\right), \tag{4.24}$$

where $\mathbf{x}^*_{i|k}$ denotes the locally optimal state of the Ego at prediction step i when the current time step is k. Note that the initial condition $\mathbf{x}^0(k)$ is known and the final predicted state is computed using the prediction model (2.26) and the final input defined in (4.22). The predicted state of the Follower is also initialized using the shift initialization, however, since the inputs are unknown, we simply rely on the nominal prediction model (2.26) with zero input. As the Leader is assumed to drive at a constant velocity, its predicted states are no optimization variables and, therefore, can be computed exactly. The slack $\boldsymbol{\epsilon}_k$ is initialized with zero vectors to promote solutions that are feasible to the unrelaxed MPC problem.

4.4.2 Implementation of Gaussian Process Dynamics

As mentioned before, CasADi provides great flexibility in the construction of the problem. We exploit its symbolic framework to construct the GP-based prediction models as well as the multiple-shooting constraints in which they are embedded. The closed-form expressions of (3.10) and (3.18) are modeled as CasADi functions. Consequently, CasADi can use these expressions to compute their gradients symbolically for optimization using IPOPT.

Downsampling of Training Data In order to reduce the computational burden, the initial training data $\mathcal{D}_0$ is uniformly downsampled with a factor 2. For example, during an episode of $T = 20$ seconds at $f_s = 4Hz$, we obtain 80 samples, of which only $|D_0| = 40$ are used for inference during the next episode. Moreover, considering the nominal length-scale of the kernel function is set to $l = 10$ we do not see major improvements in the quality of the predictions when using a finer sampling grid. However, for shorter length-scales the

correlation with the training points will reduce and finer sampling of the state space could further improve predictions. With online learning, we do use every observation for inference to maximize information collection and the adaptability of the prediction model.

4.4.3 Simulation

The IDM model, detailed in section 2.2.2, uses the acceleration signal of the reference vehicle to determine the acceleration output of the IDM. To overcome this, the MPC algorithm computes the current control input for the Ego based on the current states of all vehicles. Subsequently, this input is fed to the IDM model which computes the current control input for the Follower. Finally, the state transitions can be computed for all vehicles after which the simulation proceeds to the next iteration.

4.4.4 Hardware

The simulations are performed on an HP ZBook Studio G5, with an Intel i7-9750H CPU with a base speed of 2.6 [GHz] and 32 GB of RAM. As MATLAB has limited GPU support, we currently do not utilize the GPU. GPU utilization is considered part of future work.

4.4.5 Concluding Remarks

Conclusively, in this chapter, three MPC methods are proposed to solve the interactive motion planning problem. Firstly, an MPC which predicts the state of the Follower using a constant velocity model. Secondly, we present an interactive Gaussian process-based MPC which utilizes Gaussian process regression to predict the velocity of the Follower based on training data. This model uses online observations to passively learn the unknown residual dynamics of the Follower. Thirdly, we extend this passive GP-MPC by composing it into an active learning MPC framework from [13]. The implementation of the active learning framework using Sparse Pseudo-Input GP-MPC leads to switching input sequences and, hence, oscillatory input trajectories. A fourth algorithm is proposed that mitigates this effect by introducing a learning period of length K. At the end of this period, we update the inducing points of the GP to increase the dedication of the learning plan. Finally, we conclude the chapter with the numerical optimization methods to solve these problems.

Chapter 5

Results

This chapter presents the results of the three MPC methods, proposed in Chapter 4, with the aim to highlight the capabilities of online learning GP-MPC. As motivated in section 1.5, we confine ourselves to simulation studies. In section 5.1, we motivate and define the primary test case of this study. To this end, we focus on a challenging scenario where the Following vehicle attempts to close the gap with its Leader, and tries to prevent the Ego vehicle from merging in between. Furthermore, we provide some preliminary remarks on the simulation study and the assessment of the results. As a baseline, section 5.2 studies the behavior of the deterministic constant-velocity MPC. Subsequently, we show in section 5.3 that Gaussian process-based MPC (GP-MPC) is capable of successfully learning the behavior of other road users online. Furthermore, section 5.4 details the results of the active learning-based GP-MPC. We also consider the generalizability of the proposed motion planners by studying various initial conditions, other driving behaviors through different parameter settings of the IDM, as well as the parameterization of the slack penalty weights, in section 5.5.

5.1 Experiment Design

It should be noted that a simple prediction model would suffice in many scenarios, and provides safe and satisfactory results. However, safety is about the tail of the distribution, and considering the practically infinite operational domain of vehicles, it is essential to have a prediction model that can cope with challenging scenarios, and uncertain and unseen behavior. To investigate the learning and prediction capabilities of the various prediction models, the primary test case is designed to show strong interactions between the vehicles.

5.1.1 The Primary Test Case

The primary test case of this study is a lane merging scenario where the Ego and Follower are initialized at the same longitudinal position, i.e., side by side. The initial conditions are detailed in Table 5.1. Note that all vehicles start with zero heading angle (ψ) and zero steering angle (δ). It is important to note that the parameterization, including the slack variables, is identical for all simulation studies, apart from the horizon length N.

Table 5.1: Initial conditions of the primary test case.

State	Symbol	Value
Longitudinal pos. Ego	X^0	-75 [m]
Longitudinal pos. Follower	X^1	-75 [m]
Longitudinal pos. Leader	X^2	0 [m]
Lateral pos. Ego	Y^0	0 [m]
Lateral pos. Follower	Y^1	3.5 [m]
Lateral pos. Leader	Y^2	3.5 [m]
Velocity Ego	v^0	110 [km/h]
Velocity Follower	v^1	110 [km/h]
Velocity Leader	v^2	90 [km/h]

The Ego and Follower approach the Leader at a higher velocity than the Leader. Meanwhile, due to the parameterization of the novel Interactive MR-IDM (I-MR-IDM), detailed in section 2.2.2, the egoistic Follower does not yield to the Ego and tries to close the gap with its Leader. The nominal parameters of the I-MR-IDM are listed in Table 2.2. The reference velocities and headway time parameters for this test case are listed in Table 5.2.

Table 5.2: Parameters of the Interactive MR-IDM of the adversarial Follower.

IDM Parameter	Symbol	Value
Nominal reference velocity	v_{nom}	110 [km/h]
Nominal headway time	T_{nom}	1 [s]
Interactive reference velocity	v_{act}	140 [km/h]
Interactive headway time	T_{act}	0.25 [s]

As mentioned in Chapter 1, we perform a quantitative and qualitative analysis of the methods to get a better understanding of their capabilities and shortcomings. We study the effects of the horizon length, the learning period, and the effects of both passive and active online learning. Before we discuss these results, some preliminary remarks are in order.

5.1.2 Sparse Pseudo-Input Gaussian Process

As discussed in section 3.3.5, the computational complexity limits the capabilities of the full GP and its relevance for real-time control. During simulation studies, it was found that the full GP is very memory intensive and prohibited extensive simulations. Although we have considered both a full and a sparse GP model, we focus on the SPGP-MPC implementation.

5.1.3 Problem Formulation

Simplifications For the sake of simplicity, the study is limited to a scenario with only three traffic agents. While traffic is rather sparse, this allows for a better qualitative analysis of the prediction methods, and provides more insight into the behavior of the control algorithms. As such, this work should be considered as a proof of concept. Coping with more dense traffic and scalability are considerations for future research. Also, we do not consider any uncertainty or noise, apart from the uncertainty in the Follower's policy.

Objective Function When traffic is kept sparse for the sake of analysis, as we do here, incentivizing the Ego to merge would bias the solution to merge as quickly as possible. In section 2.3.1, we discussed this paradoxical challenge of motion planning. To this end, the problem formulation should not contain the solution, rather, the motion plan should follow from the problem. If traffic would be more dense, we could incentivize the Ego to merge to the target lane by adding a penalty in the objective function for driving in the merge lane, without introducing significant bias. In an attempt to present an unprejudiced analysis and fair comparison between the presented methods, we design the objective function to be unbiased toward any prediction model. As such, we set the objective to maintain its initial velocity and let the closing of the merge ramp be an incentive to merge.

5.1.4 Performance Evaluation

Quantitative Evaluation With motion planning, quantitative key performance indicators (KPIs) can be difficult to select. While we focus on a qualitative analysis of the motion planner, we consider some quantitative key performance indicators (KPIs). We primarily focus on four KPIs: (i) the maximum instantaneous slack variable on the collision avoidance constraints — excluding the social constraints — for both the Follower and the Leader over the entire simulation, denoted by $\epsilon_{\max}$, (ii) the highest and lowest instantaneous speed amongst all vehicles, denoted by $v_{\max}$ and $v_{\min}$, respectively, provides a measure of the traffic flow and the intervention caused by the merging Ego vehicle, (iii) the minimal instantaneous gap, that is, the smallest distance from any bumper to another bumper, between all the vehicles, denoted by $s_{\min}$, and (iv) the maximum instantaneous acceleration ($a_{\max}$) and deceleration ($a_{\min}$) that is attained during a scenario. However, these metrics do not provide a complete qualitative assessment of the prediction models. To this end, video footage of the simulations can be requested from the author and can support the reader in the interpretation and qualitative assessment of the results. An overview of all results can be found in Appendix B.

Qualitative Evaluation As mentioned before, refer to Appendix B for video footage on these simulations. Throughout this chapter, we use time-lapses and figures to visualize the qualitative results. The details of these figures are discussed next. For example, Figure 5.1 in section 5.2 shows a time-lapse of a birds-eye view of the merging maneuver. Here, the instantaneous position of the vehicles is indicated by a solid rectangle. The crosses and circles are used to indicate the predicted location of the rear axle of the Ego and the Follower, respectively. The predictions of the Leader are omitted for the sake of clarity. Furthermore, the Leader maintains a constant velocity and is assumed to be known. The predicted final position of all vehicles is indicated by a transparent rectangle. Figure 5.2 in section 5.2 shows the velocity and acceleration trajectories of all vehicles. Moreover, the predictions of the MPC at $t = 11.75$ [s] are projected onto these trajectories. In addition, Figure 5.2 also includes the function value of the activation function of the I-MR-IDM, note that this function value is unitless, as opposed to the acceleration trajectories and the input constraints.

5.2 Baseline MPC

As a baseline, we first consider the constant velocity MPC (CV-MPC) algorithm, presented in section 4.1. The constant velocity model is agnostic to the interaction between the Follower, the Ego, and the Leader, it does not anticipate any (state-dependent) acceleration or braking and merely relies on current measurements of the velocity of the Follower. Consequently, the CV-MPC predicts the future motion of the Follower by assuming that it maintains its current velocity. In this study, we investigate the quality of the predictions and study the effect of the horizon length on the performance and safety of the motion plan.

5.2.1 Constant Velocity MPC

Experiments In this section, we study the CV-MPC, detailed in Algorithm 1, in the primary test case defined in section 5.1. The initial conditions for this study are defined in Table 5.1, and the I-MR-IDM is parameterized according to Table 2.2 and Table 5.2.

Simulation Results Firstly, let us consider the case of $N = 12$ in more detail. As the Ego approaches the merge point, it identifies a gap and slows down to merge between the Leader and the Follower at $t = 9.75$ [s] as seen in Figure 5.1 and Figure 5.2. The CV-MPC assumes that the Follower maintains its current speed, which in this test case is overconservative as the Follower is slowing down. It maintains a high velocity to account for the small gap, at $t = 11.25$ [s] (Figure 5.1). The CV-MPC decelerates during the merge to maintain a safe distance from the Leader, seen at $t = 12.25$ [s] and $t = 13$ [s] in Figure 5.1. The predictions of the Follower are agnostic to any interactions, and although the response of the Follower is uncertain, its predictions remain unchanged, as seen in Figure 5.2.

The simulation results of CV-MPC in the primary test case are summarized in Table 5.3. Note that the metrics provided in Table 5.3 are described in section 5.1.4. For more detailed results, refer to Table B.2 in Appendix B. The motion plan of the CV-MPC is generally safe and either successfully merges between the Follower and Leader, or behind the two target vehicles (Table 5.3). For short horizons ($N = 6, \dots 12$), the CV-MPC is able to identify a gap between the Follower and Leader and merges into the target lane while keeping distance to both vehicles. It is important to note that in this test case, the constant velocity prediction is always conservative with respect to the Follower that is slowing down. Consequently, the Ego vehicle will try to stay in front of this conservative prediction. For long predictions horizon ($N = 14, \dots 24$), the CV-MPC cannot identify this gap as it does not anticipate the Follower braking. It merely relies on its constant velocity predictions.

Computation Time For the CV-MPC, excluding the first cold-started iteration, the average solve time over all simulations is 0.020 [s]. The maximum warm-started solve time, using the shifted MPC solution of the previous iteration, of all simulations is 0.099 [s]. The CV-MPC is well capable to run in real-time. Refer to Table B.2 for detailed solve times.

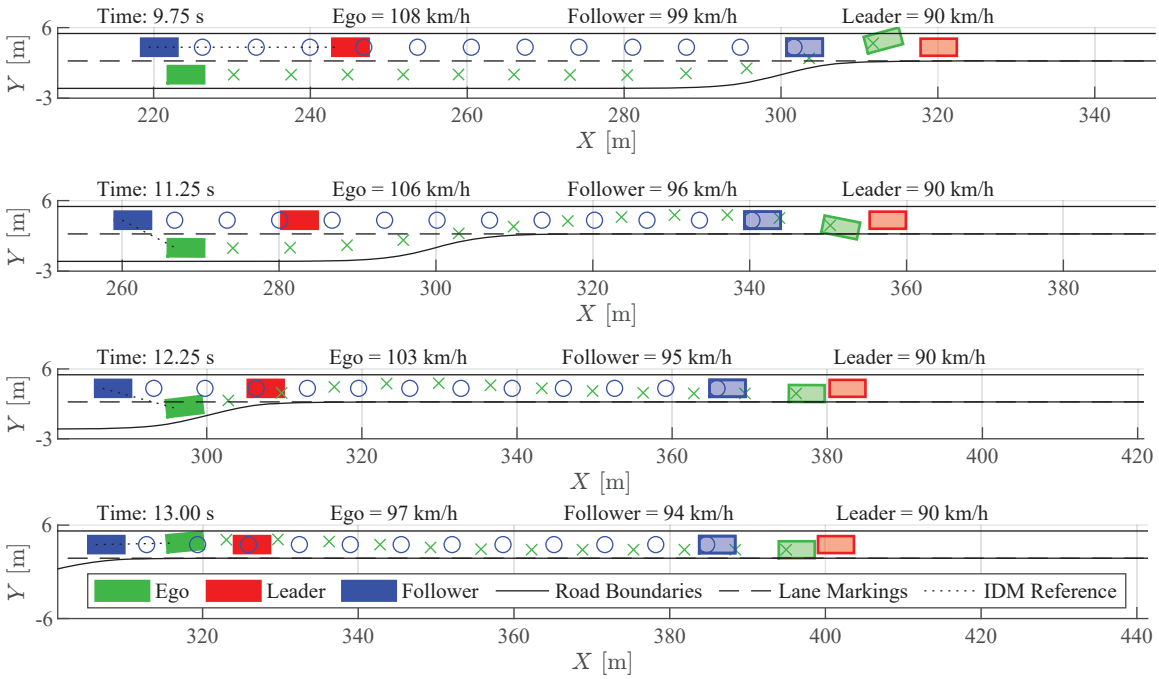

Figure 5.1: Time-lapse of CV-MPC on the primary lane merging test case with a prediction horizon of $N = 12$.

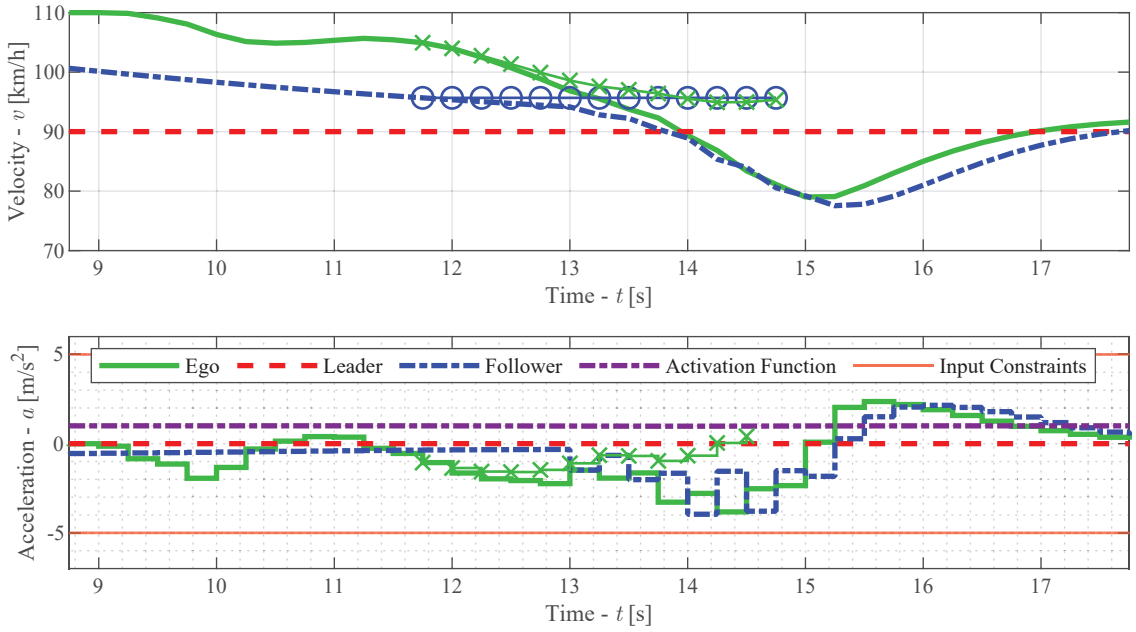

Figure 5.2: Velocity and acceleration trajectories with the predictions of the CV-MPC at $t = 11.75$ [s] on the primary lane merging test case with a prediction horizon of $N = 12$.

Table 5.3: Results of Constant Velocity MPC

N	Result	$\epsilon_{\max}$ [−]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	Note
6	Merged in Between	0.26	115	86	1.5	4.2	4.4	-
8	Merged in Between	0.29	115	86	1.5	4.8	4.2	-
10	Merged in Between	0.46	115	78	2.6	3.3	3.1	-
12	Merged in Between	0.47	115	78	2.4	4.0	3.0	-
14	Merged Behind	0.04	115	72	1.6	4.0	5.6	-
16	Merged Behind	0	115	75	1.5	3.8	6.7	-
18	Merged Behind	0	115	71	1.5	3.3	8.4	-
20	Merged Behind	0	115	72	1.5	3.3	9.8	-
22	Merged Behind	0	115	72	1.5	2.6	9.4	-
24	Merged Behind	0	115	73	1.5	2.3	10.2	-

Conclusion In many cases, a simple prediction model suffices. This also applies here, the CV-MPC is able to identify a gap safely merge in between, despite the inferior prediction quality of the constant velocity model. From the figures and video footage, it can be concluded that the CV predictions are unable to capture the relevant dynamics. In this test case, the Follower slows down so as not to collide with its Leader. Consequently, the constant velocity predictions are always conservative. As such, if it is initially able to identify a gap, this gap will only grow and the Ego can safely merge. Currently, the CV-MPC relies on the softly constrained collision avoidance constraints to identify this gap. When the penalty weights of the slack variables are increased, i.e. the constraints are *hardened*, the motion planner will become more dependent on the quality of the prediction model. This dependency is further studied in section 5.5.3.

It should be noted that the solution of the CV-MPC strongly depends on the parameterization of the problem. The CV prediction model does not anticipate the braking of the Following vehicles. However, due to the parametrization, the CV-MPC uses the slack variables to violate the constraints in order to make it easier to merge. However, these slack variables are only intended to recover from potential infeasibility. During the study, it was found that the penalty on the slack variables was set too low to achieve this. In section 5.7, we reflect on the simulation studies and their results in detail.

5.3 Passive Learning with GP-MPC

In the primary test case, the Ego vehicle must merge into the merge lane, while the adversarial Follower tries to close the gap between the Follower and the Leader. In order to anticipate the future motion of the Follower in the lane merging scenario, we use previous observations to improve the predictions of the Follower. To this end, the GP-MPC uses a Sparse Pseudo-Input GP-based prediction model, detailed in Chapter 3.

Firstly, in section 5.3.1, we focus on the case of solely online learning, in which the predictions of the GP are only based on data observed during the current episode. Secondly, in section 5.3.2, we consider iterative learning in which we *pre-train* the GP using data from previous episodes of the same scenario. As seen in the previous section on the CV-MPC, the prediction horizon length N is an important hyperparameter in these predictive methods, as the accuracy of the predictions strongly depends on the prediction horizon length. Hence, we study the effect of the horizon length on the overall performance of the motion planner.

5.3.1 Online Passive Learning

Experiments In this section, we focus on online passive learning for the aforementioned primary test case defined in section 5.1. The initial conditions for this study are defined in Table 5.1, and the I-MR-IDM is parameterized according to Table 2.2 and Table 5.2. The passive learning SPGP-MPC is detailed in Algorithm 2 (section 4.2). In this section, we focus on purely online learning, such that the GP starts without any training data ($\mathcal{D}_0 = \emptyset$).

Simulation Results Again, we first discuss the results of SPGP-MPC for a prediction horizon of $N = 12$ in more detail, to compare its performance with the CV-MPC. In this case, the Ego approaches at approximately the same speed as the CV-MPC, as seen at $t = 9.75$ [s] in Figure 5.3. Note that the SPGP-MPC considers the uncertainty of the predictions for collision avoidance and the 2σ-bounds on the final predicted position of the Follower are resembled by the two transparent blue rectangles. As such, low and high covariance are resembled by overlapping or separated rectangles, respectively. While both the CV-MPC and SPGP-MPC slow down to merge between the two vehicles, the SPGP-MPC slows down further as it anticipates the Follower braking, which can be seen at $t = 11.25$ [s] and $t = 12.25$ [s] in Figure 5.3. In Figure 5.4, we can observe the predictions of the velocity profile of the Follower. Based on online observations, the SPGP-MPC expects the Follower to initially slow down. However, the interactive predictions are uncertain about how the Follower might respond to the Ego changing its speed, and the SPGP-MPC accounts for this uncertainty through the covariance, indicated by the approximated 2σ-bounds on the predicted velocity. Consequently, the SPGP-MPC takes extra caution with the moderate input sequence at $t = 11.75$ [s], seen in Figure 5.4. However, as the GP gathers more data and learns about the Follower's behavior, it better predicts the velocity and can therefore increase its braking effort (Figure 5.4), to maintain sufficient distance from both the Follower as well as the Leader, as seen in Figure 5.3 at $t = 13$ [s] and Table 5.4.

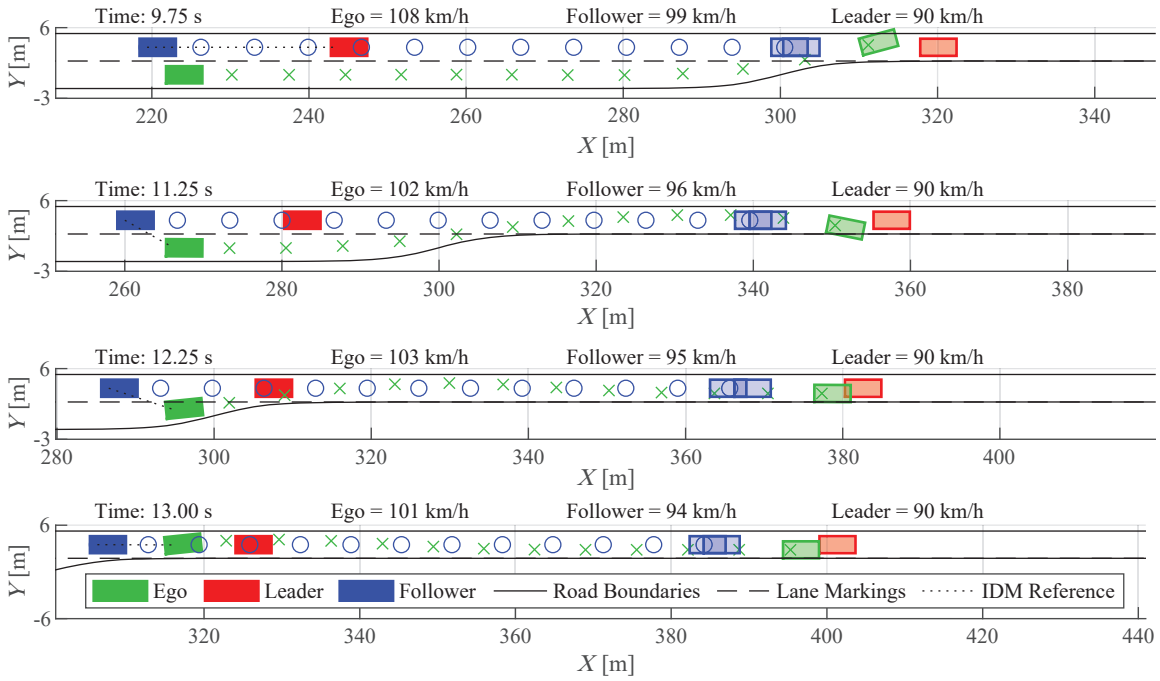

Figure 5.3: Time-lapse of passive learning SPGP-MPC without pre-training on the primary lane merging test case with a prediction horizon of $N = 12$.

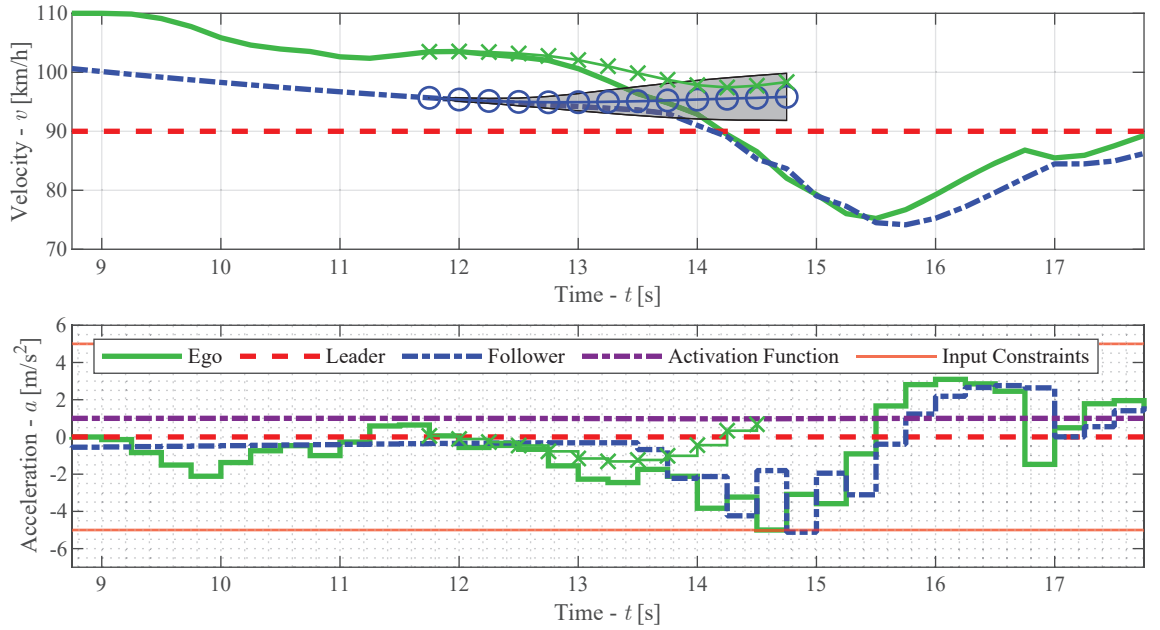

Figure 5.4: Velocity and acceleration trajectories with the predictions of passive learning SPGP-MPC without pre-training at $t = 11.75$ [s] on the primary lane merging test case with a prediction horizon of $N = 12$.

Table 5.4: Results of passive learning SPGP-MPC without pre-training

N	Result	$\epsilon_{\max}$ [−]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	Note
6	Merged in Between	0.25	115	85	1.5	4.3	4.4	-
8	Merged in Between	0.41	115	78	2.7	5.0	3.2	-
10	Merged in Between	0.48	115	77	2.4	5.3	2.9	-
12	Merged in Between	0.52	115	74	3.1	5.2	2.6	-
14	Merged Behind	0	115	71	1.5	4.4	5.7	-
16	Merged Behind	0	115	68	2.0	3.9	8.1	-
18	Merged Behind	0	115	66	2.4	3.7	10.5	-
20	Merged Behind	0	115	65	2.6	3.3	14.6	-
22	Merged Behind	0	115	68	2.2	3.2	15.2	-
24	Merged Behind	0	115	71	1.7	2.9	15.5	-

The simulation results of passive learning with the SPGP-MPC without pre-training in the primary test case are summarized in Table 5.4. For more detailed results, refer to Table B.3 in Appendix B. For shorter prediction horizons ($N = 6, \ldots, 12$), the passive SPGP-MPC can safely merge in between the two target vehicles by leveraging the online observations to predict the motion of the Follower. For longer prediction horizons ($N = 14, \ldots, 24$), the predictions of the GP have too much uncertainty to identify a safe gap, causing the Ego to merge behind the Follower. Although the Ego cannot merge in between, learned predictions can still be leveraged in the motion plan. The SPGP-MPC takes caution when the Follower starts braking after it did not yield, and maintains a safe distance when merging behind the Follower.

As mentioned in section 5.2, the predictions of the CV-MPC are overconservative with respect to the Follower that is slowing down. Consequently, the performance of the cautious SPGP-MPC is similar to that of the 'conservative' CV-MPC. Although quantitatively the performance of the CV-MPC and the passive learning SPGP-MPC are comparable, the prediction quality of the SPGP-MPC is qualitatively superior. The SPGP-MPC shows potential to leverage these predictions, however, the performance of the motion plan is very sensitive to parameterization, as discussed in section 5.5.3 and section 5.7.

The SPGP-MPC successfully utilizes online observations to learn the interactive motion of the Follower, identifies a gap, and safely merges between the two vehicles. Not only could the predictions be leveraged to improve the performance of the motion plan, for example, by successfully merging between the two target vehicles, they also account for the uncertainty associated with the interactive predictions that inherently provide extra caution when we are uncertain of the predictions, which should improve the overall safety of the motion plan.

Computation Time Again, excluding the first cold-started iteration, the average solve time of SPGP-MPC without pre-training over all simulations is 0.323 [s]. The maximum warm-started solve time of all simulations is 3.574 [s]. For $N \leq 16$ the average and maximal solve time are below 0.273 [s] and 0.455 [s], respectively, showing the potential to run this algorithm in real-time. Refer to Table B.2 for details on each simulation.

Conclusion In the case of purely online, passive learning, the SPGP-MPC starts completely agnostic to the Follower and assumes it will maintain a constant velocity. However, as data is gathered, the predictions of the Follower improve, allowing the MPC to leverage these predictions in planning the motion of the Ego vehicle. Whilst the Ego starts off without any knowledge of the behavior of the Follower, the GP-based prediction model can accurately predict the behavior of the Follower after several seconds of observing. Although the interaction-aware motion planner accounts for the covariance provided by the GP by keeping more distance from the Follower, for longer prediction horizons, the predictions have too much uncertainty to safely merge in between the target vehicles, and the Ego safely merges behind the two target vehicles. Similar to the CV-MPC, the SPGP-MPC has a clear cross-over point in its decision-making: merge in between for short horizons, and merge behind for longer horizons. As opposed to CV-MPC, this is not due to poor predictions, but due to the anticipated uncertainty of the predictions. In conclusion, the SPGP-MPC can successfully learn the previously unseen behavior of the Follower, relying only on online observations. Although the SPGP-MPC shows potential for interaction-aware motion planning, further validation of the motion planner is required to determine if it is outperform the CV-MPC in terms of safety and performance.

5.3.2 Iterative Passive Learning

Experiments In this section, rather than starting without any training data, the GP is pre-trained with the observations $\mathcal{D}_0$ from the same scenario with identical parameterization, including the prediction horizon length N. For example, the first iteration uses training data from the online passive SPGP-MPC, studied in the previous section. The second iteration extends this data with the observations of the first iteration, *etc.* We employ the SPGP-MPC algorithm, detailed in Algorithm 2 for iterative learning. We confine ourselves to prediction horizons of length $N = 12,\ 14,\ 16$, for these horizons just succeed or fail to merge in between the target vehicles and iterative learning could change this outcome.

Simulation Results The results of passive learning with the SPGP-MPC using with pre-training in the primary test case are summarized in Table 5.5. For more detailed results, refer to Table B.4. Figure 5.5 shows the time-lapse for the passive learning SPGP-MPC that is pre-trained using the observations from the simulation in section 5.3.1 with $N = 12$. The motion plan of the pre-trained SPGP-MPC is almost identical to that of the SPGP-MPC untrained, up to $t = 11.25$ [s]. The predictions of the pre-trained SPGP-MPC and the untrained SPGP-MPC at $t = 11.75$ [s] are comparable up to $t = 13$ [s], as seen in Figure 5.4

and Figure 5.6. The pre-trained SPGP-MPC predicts that the Follower is going to slow down based on its training data, whereas the predictions of the untrained SPGP-MPC tend to a constant velocity with large uncertainty. While the GP does not capture all the high frequent dynamics, the low frequent dynamics are accurately predicted and enables the SPGP-MPC to improve its performance. As such, the pre-trained SPGP-MPC slows down during the merge, as seen at $t = 12.25$ [s] and $t = 13$ [s] in Figure 5.5. Consequently, the Ego does not need to decelerate like in the previous case, and it smoothly merges in between the target vehicles while adjusting its speed to that of the Leader, as seen in Figure 5.5. The Follower briefly brakes accordingly and both vehicles maintain a higher velocity, thereby improving traffic flow. In addition, the Ego vehicle can maintain a greater gap with the other vehicles with the pre-trained SPGP-MPC, as seen in Table 5.5.

In turn, we leverage the observations from the first iteration to pre-train the second iteration. Again, the motion of both the Ego and the Follower is comparable up to $t = 12.25$ (Figure 5.7). At $t = 11.75$ [s] the SPGP-MPC plans to smoothly brake into the merge as in the previous scenario, however, at $t = 12.25$ the Ego accelerates (Figure 5.8). Considering the fact that we are solving a complex receding horizon nonlinear optimization problem with varying conditions, it is possible that this brief acceleration is a different local minimum to the optimization problem, and the SPGP-MPC briefly deviates from its intended course. This acceleration is followed by a strong deceleration of the Ego, and consequently, a strong deceleration of the Follower. Moreover, the low simulation frequency could have delayed the braking effort of the Follower which could have contributed to the high deceleration value.

The untrained SPGP-MPC with $N = 14$ was unable to merge in between. However, using the observations of this scenario, the pre-trained SPGP-MPC is able to identify a gap and merge in between the target vehicles. As this training data contains data of merging behind, the predictions have larger covariance when the SPGP-MPC decides to merge in between, for this behavior is not resembled in the training data, yet, the motion planner accounts for this uncertainty and safely merges in between the vehicles. However, this does result in smaller gaps and induces larger decelerations by the Follower. During the second iteration, the previous lane merge is resembled in the training data, and the SPGP-MPC can leverage these improved predictions. Consequently, the Ego merges in between the vehicles with increased distance to the other vehicles and lower induced decelerations. The third iteration extends this trend, leading to an even minimal gap and lower decelerations.

For a longer prediction horizon of $N = 16$, pre-training the SPGP-MPC does not help it to merge in between the target vehicles. However, the improved predictions enable the Ego to merge at a higher velocity and with a smaller gap than with the untrained SPGP-MPC with $N = 16$. Also here, we can observe improved performance during the second iteration, with an even higher minimum velocity and a smaller (but safe) gap to the Follower. An attempt to reduce the conservatism by setting $\sigma = 1$ resulted in a collision, demonstrating that long predictions with little experience require adequate caution. By pre-training the SPGP-MPC with a horizon of $N = 16$ on a successful merge (using data from $N = 12$), the SPGP-MPC is able to identify a gap and merge between the two target vehicles.

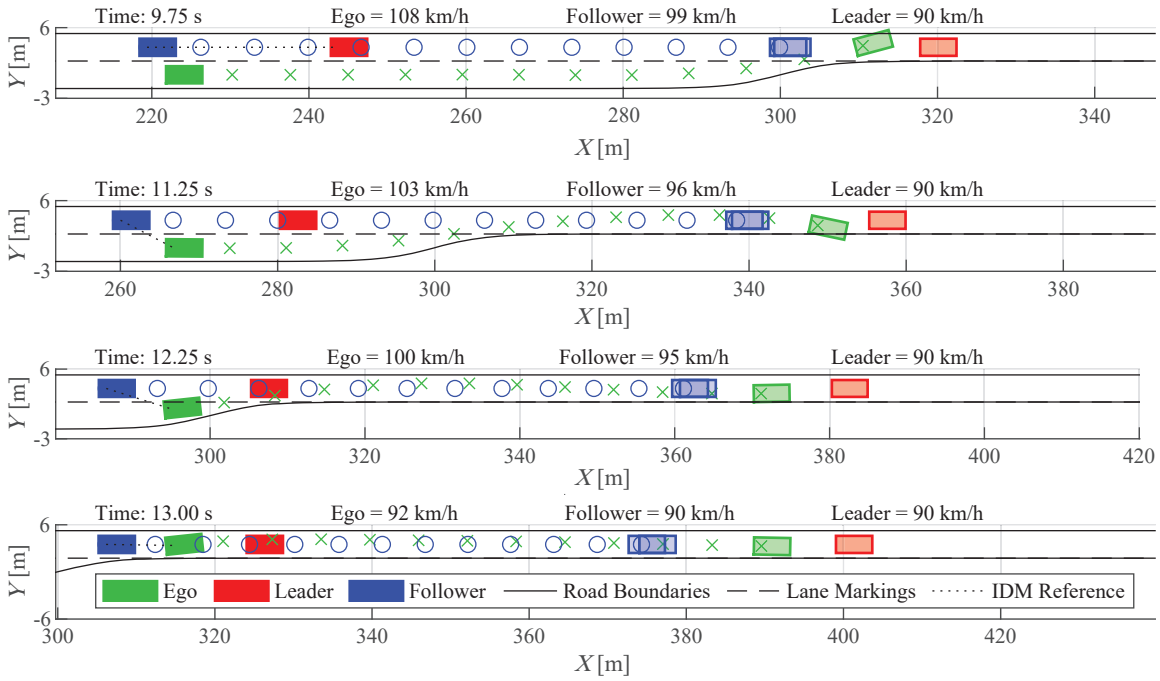

Figure 5.5: Time-lapse of passive learning SPGP-MPC with one iteration of pre-training on the primary lane merging test case with a prediction horizon of $N = 12$.

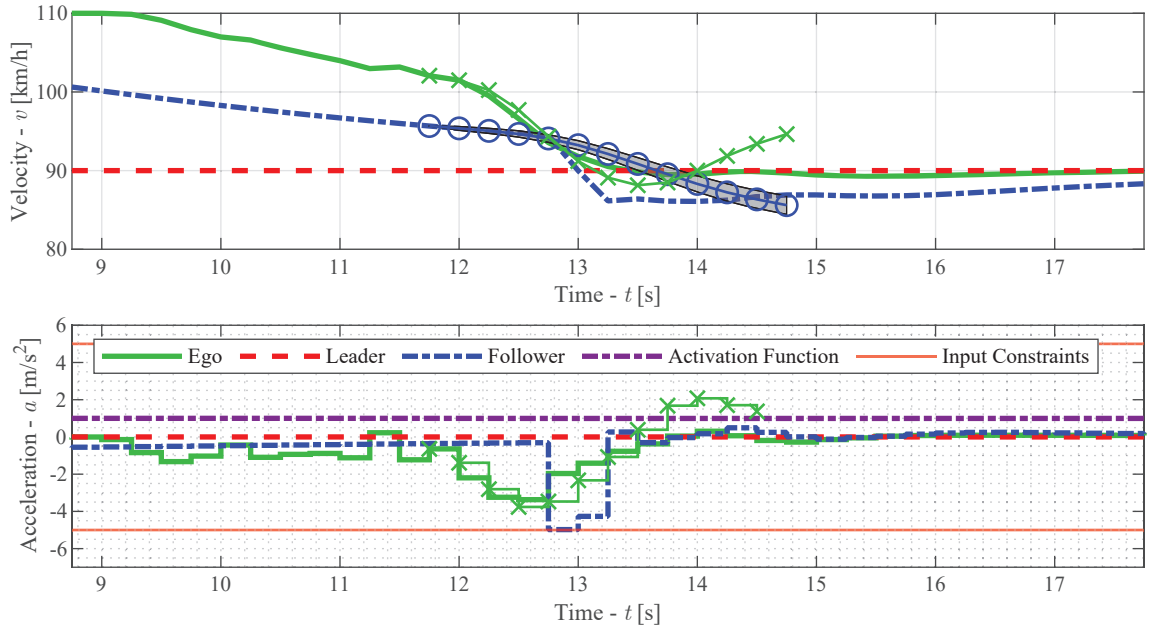

Figure 5.6: Velocity and acceleration trajectories with the predictions of passive learning SPGP-MPC with one iteration of pre-training at $t = 11.75$ [s] on the primary lane merging test case with a prediction horizon of $N = 12$.

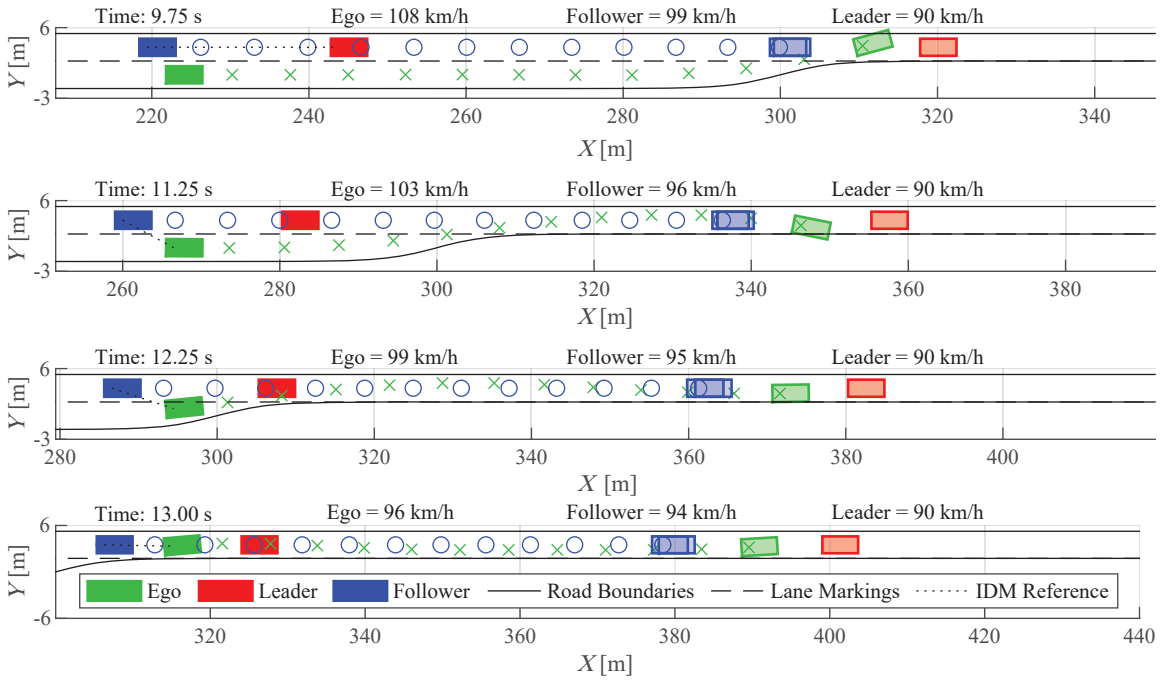

Figure 5.7: Time-lapse of passive learning SPGP-MPC with two iterations of pre-training on the primary lane merging test case with a prediction horizon of $N = 12$.

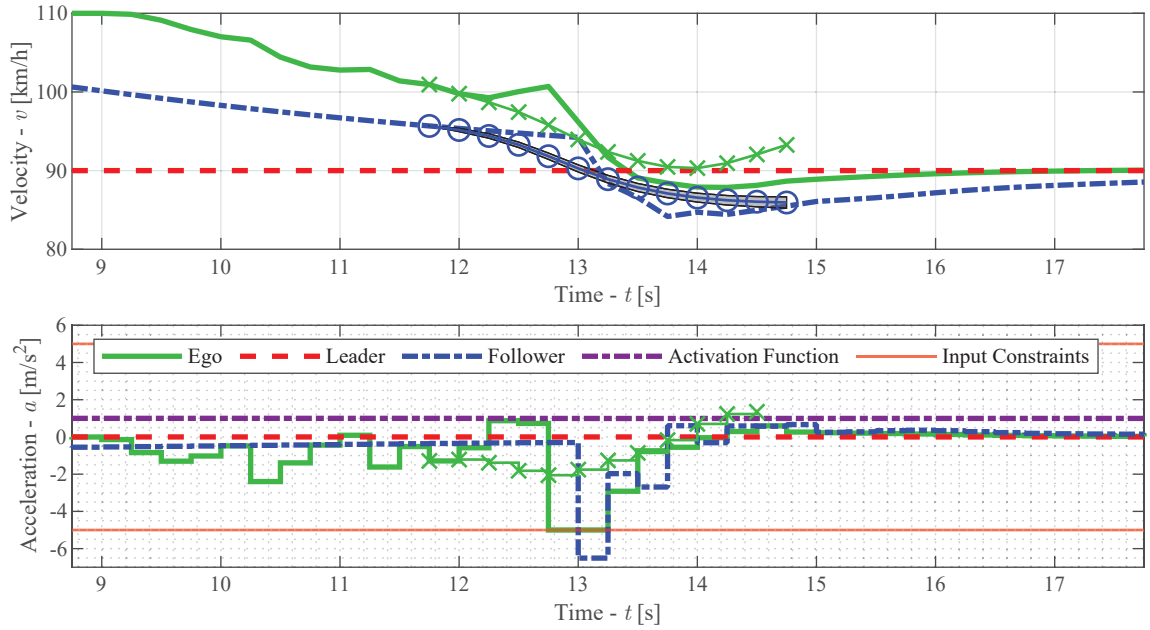

Figure 5.8: Velocity and acceleration trajectories with the predictions of passive learning SPGP-MPC with two iterations of pre-training at $t = 11.75$ [s] on the primary lane merging test case with a prediction horizon of $N = 12$.

Table 5.5: Results of passive learning SPGP-MPC with pre-training

N	Training Iteration	Result	$\epsilon_{\max}$ [−]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	Note
12	1	Merged in Between	0.33	115	86	1.5	5.0	3.7	-
12	2	Merged in Between	0.30	115	84	1.5	6.5	3.5	Harsh brake Follower
14	1	Merged in Between	0.52	115	72	3.7	6.4	2.6	Harsh brake Follower
14	2	Merged in Between	0.34	115	86	1.5	5.4	3.5	Moderate brake Follower
14	3	Merged in Between	0.32	115	80	1.6	4.0	3.7	-
16	1	Merged Behind	0	115	74	1.5	3.8	6.7	-
16	2	Merged Behind	0	115	75	1.5	3.8	6.6	-
16	2	Collision with Follower	0.97	115	90	5.0	5.0	0.0	Higher risk: $\sigma = 1$
16	1 with $N = 12$	Merged Behind	0.40	115	86	1.5	5.9	3.3	Moderate brake Follower

Computation Time The average solve time of the warm-started SPGP-MPC with pre-training over all simulations is 0.323 [s], which is close to the sampling time of 0.25 [s]. The maximal solve time of all simulations is 1.595 [s]. Future research has to determine how this scales with large amounts of pre-training, however, these solve times demonstrate the real-time potential of this algorithm. Refer to Table B.4 for details on each simulation.

Conclusion Rather than merely relying on data that is gathered online, the GP-based prediction model can also exploit *offline* training data that is obtained from past scenarios. Using these past observations on a similar scenario, enables the SPGP-MPC to better anticipate the motion of the Follower, with reduced variance. Consequently, the motion planner is able to identify a gap to safely merge in between the vehicles more in advance than without pre-training. With little pre-training this can lead to close encounters with the Follower, however, as we obtain more data, the performance increases significantly. For longer horizons, the MPC decides to safely merge behind as the predictions are still subject to too much uncertainty to safely merge in between. However, iterative learning can still improve the performance in such cases, as it enables the Ego to merge more smoothly behind the traffic than in the untrained case. This uncertainty is vital for safety, as reducing the conservatism compromises the safety of the motion plan. Moreover, training the GP on merging in between, also enables longer horizons to merge in between. In conclusion, pre-training can improve the performance of the SPGP-MPC, provided that the motion planner considers the uncertainty of its predictions based on the training data.

5.4 Active Learning with GP-MPC

In section 5.3, we considered the SPGP-MPC motion planner with passive learning. In this section, we investigate SPGP-MPC using active learning. To this end, we use our SPGP-MPC with the active learning MPC framework of Soloperto *et al.* [13]. By using active learning, we aim to further advance the performance of the motion planner through incentivized data gathering and exploration of the state space. As with the CV-MPC and the passive learning SPGP-MPC, we study the primary test case, which is introduced in section 5.1, to provide a comparison between the three motion planners.

Firstly, in section 5.4.1, we focus on the case of solely online learning, in which the predictions of the GP are only based on data observed during the current scenario. In section 5.4.2, we study the behavior that emerges when using a pre-trained sparse GP model with the active learning-based MPC. Finally, in section 5.4.3, we consider iterative learning in which we *pre-train* the GP using data from previous episodes of the same scenario.

5.4.1 Online Active Learning

Experiments In this section, we focus on purely online active learning for the primary test case defined in section 5.1, whose initial conditions are defined in Table 5.1. The I-MR-IDM is parameterized according to Table 2.2 and Table 5.2. The hyperparameters of the active learning framework are defined in section 4.3.2. Due to the emergent behavior discussed in section 5.4.2, there are significant shortcomings with Algorithm 3. Hence, in this section, we use our novel active learning SPGP-MPC, Algorithm 4, with a nominal learning period of $K = 5$, to ensure that the results are comparable with those of iterative active learning and follow from the same algorithm. In this section, the motion planner starts without any training data $\mathcal{D}_0 = \emptyset$. Also here, we investigate the effect of the prediction horizon length N on the overall performance and safety of the active learning-based motion planner.

Simulation Results As with the other studies so far, we first discuss the active learning SPGP-MPC for $N = 12$ in detail. While the CV-MPC and passive learning SPGP-MPC maintain a constant velocity until they see the closing of the merge lane, the active learning SPGP-MPC actively explores the state space, as seen in Figure 5.9. So far, the I-MR-IDM has always been *active* since the Ego and Follower were very close to one another. In Figure 5.9 we see a clear demonstration of the Interactive MR-IDM. When the Ego vehicle falls behind, the value of the activation function drops, the Follower smoothly changes to its nominal driving behavior and increases the gap to its Leader. As the Ego accelerates again, the activation function rises and the Follower continues to close the gap.

Figure 5.10 shows the solution to the primary MPC problem which tries to minimize the primary objective function, as well as the solution to the learning MPC problem which actively explores the state space by maximizing the covariance of the GP. Due to its exploration, the Ego approaches the merge point at a much higher speed than the other MPCs, at $t = 9.75$ [s] in Figure 5.11. As a result, it is in a better position to merge in between. Ego decreases its speed while considering the uncertainty of the predictions ($t = 11.25$ [s]).

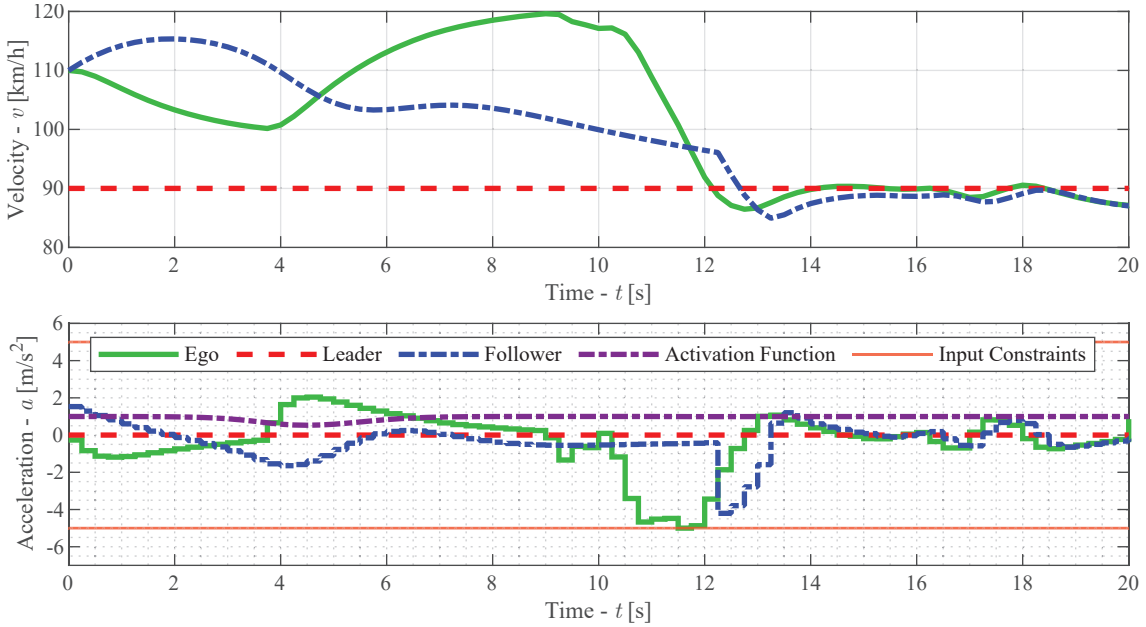

Figure 5.9: Velocity and acceleration trajectories of active learning SPGP-MPC without pre-training on the primary test case with a prediction horizon of $N = 12$.

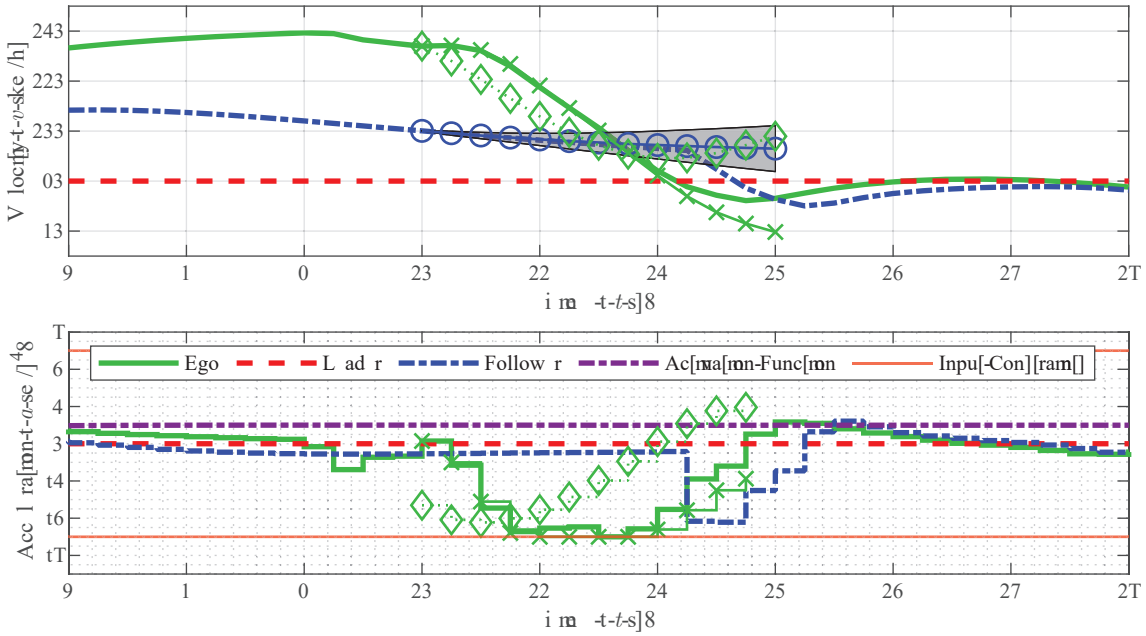

Figure 5.10: Velocity and acceleration trajectories with the predictions of primary (diamonds) and learning (crosses) MPC without pre-training at $t = 10$ [s] on the primary lane merging test case with a prediction horizon of $N = 12$.

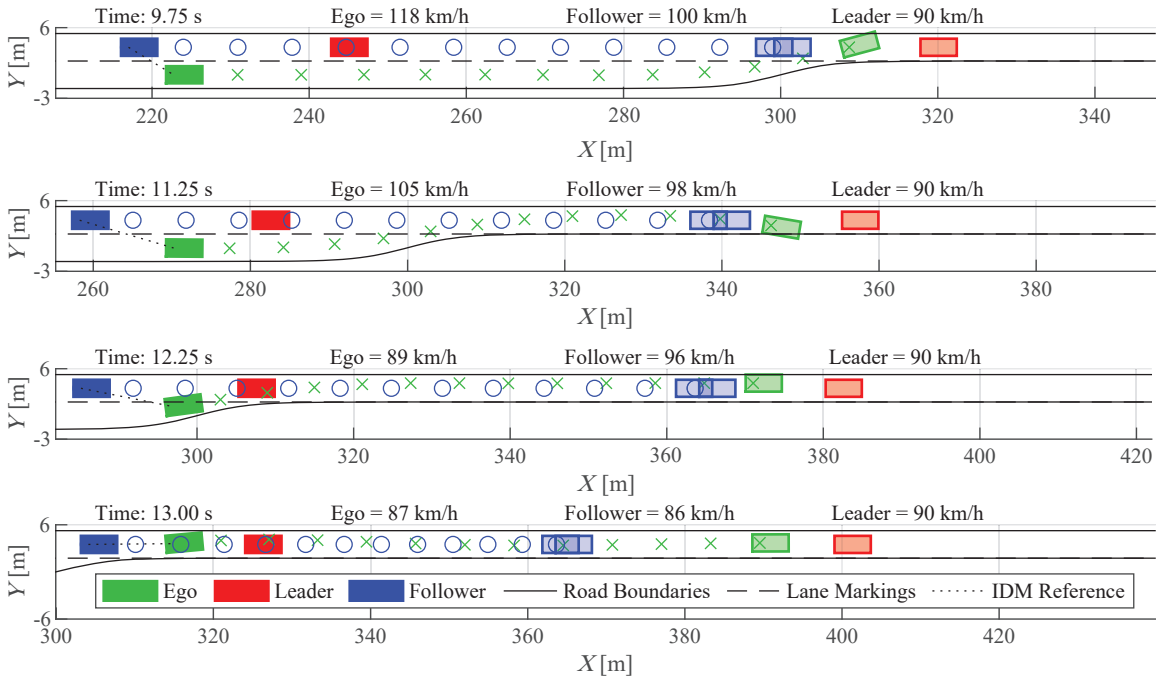

Figure 5.11: Time-lapse of active learning SPGP-MPC without pre-training on the primary lane merging test case with a prediction horizon of $N = 12$.

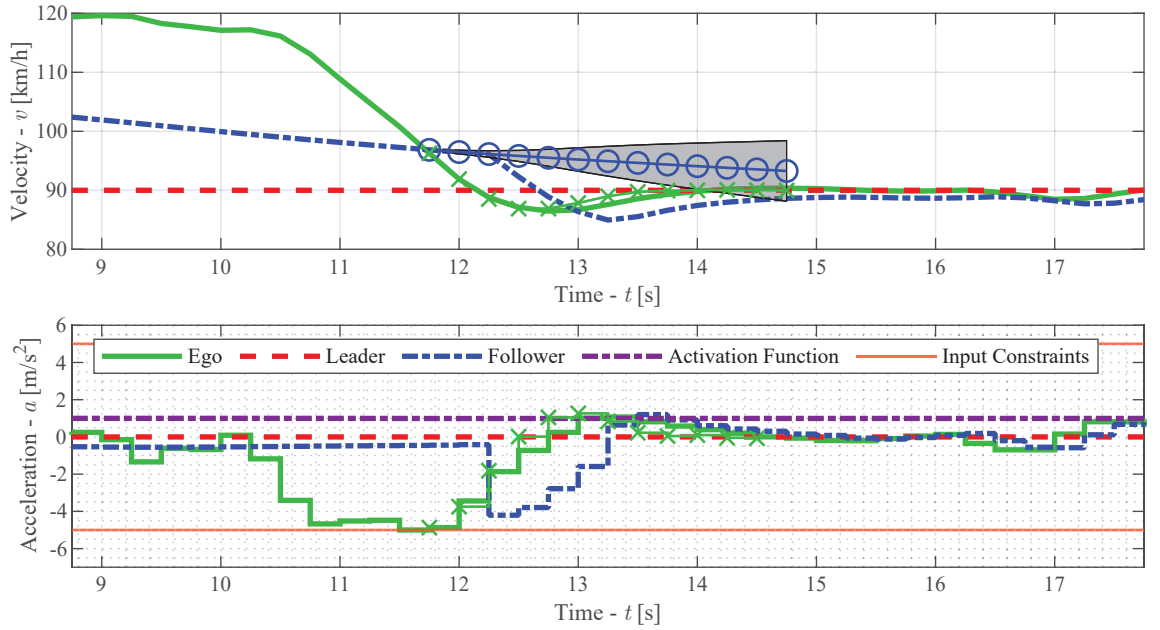

Figure 5.12: Velocity and acceleration trajectories with the predictions of active learning SPGP-MPC without pre-training at $t = 11.75$ [s] on the primary lane merging test case with a prediction horizon of $N = 12$.

Table 5.6: Results of active learning SPGP-MPC without pre-training

N	Result	$\epsilon_{\max}$ [−]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	Note
6	Collision with Leader	0.87	128	66	4.8	5.0	0.0	-
8	Merged in Between	0.15	124	78	3.7	7.1	4.3	Harsh brake Follower
10	Merged in Between	0.10	121	85	2.2	5.0	4.6	-
12	Merged in Between	0.11	120	85	2.0	5.0	4.7	-
14	Merged in Between	0.11	123	84	2.3	5.0	4.5	-
16	Merged in Between	0.04	126	84	3.8	5.0	4.3	-
18	Merged in Between	0.06	128	81	5.0	5.0	3.8	-
20	Merged Behind	0	116	67	3.6	5.0	17.7	-
22	Merged Behind	0	116	66	3.6	4.5	19.3	-
24	Merged Behind	0	115	70	2.9	3.8	21.2	-

Although the predictions of the velocity of the Follower are comparable to that of passive learning SPGP-MPC, seen in Figure 5.12, the Ego's state is more advantageous to complete the merging maneuver. The Ego approaches at a much higher velocity and has to brake strongly. However, as the Ego keeps an adequate distance from the Follower, the induced deceleration of the Follower is lower than for the untrained passive learning SPGP-MPC. At $t = 13$ [s], both the Ego and the Follower have almost attained their final speed and do not need to reduce their momentum, as opposed to the untrained passive learning SPGP-MPC. Furthermore, both the Ego and Follower can maintain a significantly higher minimal velocity, as seen in Table 5.6, which indicates that traffic flow is improved.

The results of active learning with the SPGP-MPC without pre-training in the primary test case for various horizon lengths N are summarized in Table 5.6. For more detailed results, refer to Table B.5 in Appendix B. For short horizons, the active learning SPGP-MPC still explores the state space, however, due to its short horizon it lacks sufficient anticipation to safely merge. This leads to a collision with the Leader ($N = 6$) or a harsh brake by the Follower ($N = 8$). However, for the horizons $N = 10, \ldots, 18$ the active learning SPGP-MPC can leverage these explorations to safely merge as seen for $N = 12$, above. Furthermore, the minimum velocity of all vehicles $v_{\min}$ in the scenario is on average much higher than for the untrained passive learning SPGP-MPC (Table 5.6). In addition, the active learning SPGP-MPC is able to maintain a greater gap from the other vehicles, and the maximum instantaneous slack is significantly lower, compared to the CV-MPC and passive learning SPGP-MPC. For long horizons ($N = 20, 22, 24$), active learning does not contribute to any increase in performance compared to their passive learning counterpart.

Computation Time The average solve time of the warm-started, untrained active learning SPGP-MPC over all simulations is 1.361 [s] and the maximal solve time of all simulations is 8.014 [s]. While there is potential to run this algorithm real-time, further improvement of the algorithm is required to realize this. Refer to Table B.5 for detailed solve times.

Conclusion The active learning framework from Soloperto *et al.* [13] is exploited by the active SPGP-MPC to actively learn the interaction dynamics and explore the state space. To this end, the SPGP-MPC actively seeks states that are associated with large uncertainty. Consequently, the Ego vehicle deviates from its nominal solution, starts to vary its speed in order to perturb the Follower and moves to a position that enables it to improve its performance. As a result, the overall traffic flow is improved on average and the Ego maintains greater distance from the Follower, when compared with the untrained passive SPGP-MPC. In conclusion, the active learning framework Soloperto *et al.* [13] enables constrained exploration and can improve the performance of the untrained SPGP-MPC.

5.4.2 Oscillatory Behavior

Experiments In this section, we investigate emergent behavior that arises when the active learning SPGP-MPC is pre-trained. Initially, Algorithm 3 has been designed to use the SPGP-MPC in an active learning framework with its nominal receding horizon. Typically, the training data as well as the pseudo-inputs are updated every time step. Without pre-training Algorithm 3 does not provide any issues, however, it is found that updating the pseudo-inputs of the SPGP at every time step leads to oscillatory behavior when the SPGP is pre-trained with previous observations. To mitigate this oscillatory behavior, we propose a novel active learning SPGP-MPC algorithm that uses a learning period of length K. This algorithm, detailed in Algorithm 4, updates the pseudo-inputs every K time steps, while maintaining the receding horizon of the MPC and expanding the training set every single time step. Note that Algorithm 4 with $K = 1$ is equivalent to Algorithm 3.

Subsequently, we study the effect of the learning period length K on the overall performance and the oscillatory behavior of the pre-trained SPGP-MPC using active learning. The hyperparameters of the active learning framework are selected in section 4.3.2. We confine ourselves to a single prediction horizon $N = 12$ as we only seek to identify and rectify this oscillatory behavior, which is independent of the prediction horizon length. The active learning SPGP-MPC is pre-trained with the observations from one scenario.

Simulation Results In Figure 5.13, we can observe the oscillatory behavior that occurs due to the phenomenon described above. The switching input sequence prohibits the pre-trained active learning SPGP-MPC from active exploration. In Figure 5.14 the oscillatory behavior is rectified by using our novel active learning SPGP-MPC algorithm with a learning period of $K = 3$, which has sufficient dedication to explore the state space and successfully merge between the two vehicles. By increasing the learning period to $K = 5$ and $K = 10$ the explorations become more low frequent, as seen in Figure 5.15 and Figure 5.16, respectively.

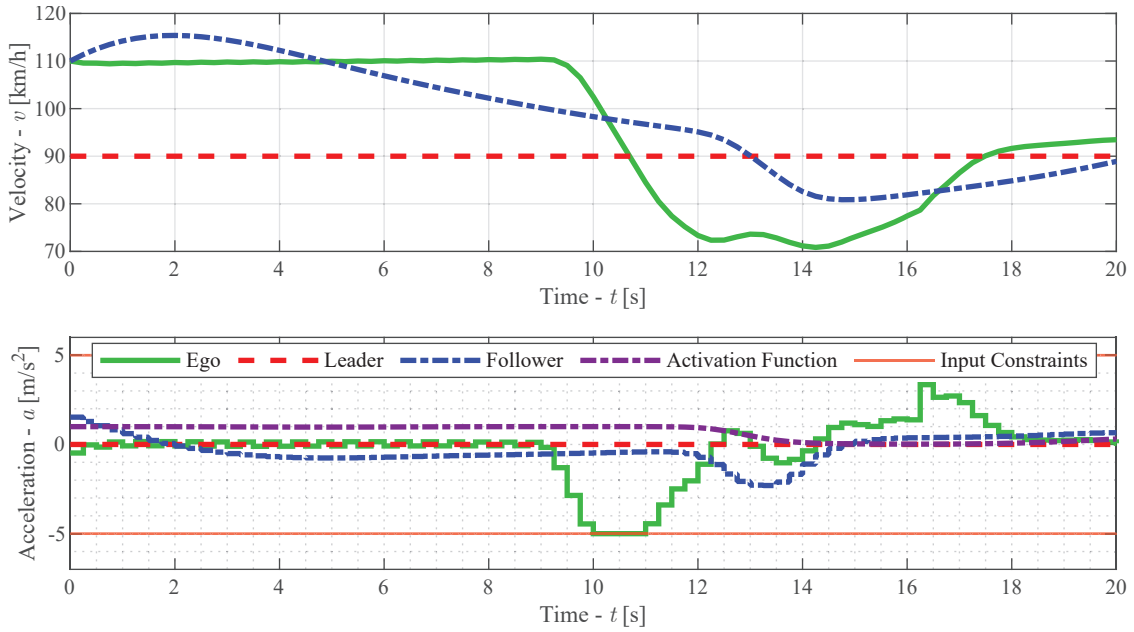

Figure 5.13: Oscillatory velocity and acceleration trajectories of the pre-trained active learning SPGP-MPC on the primary lane merging test case with a prediction horizon of $N = 12$ and a learning period of length $K = 1$.

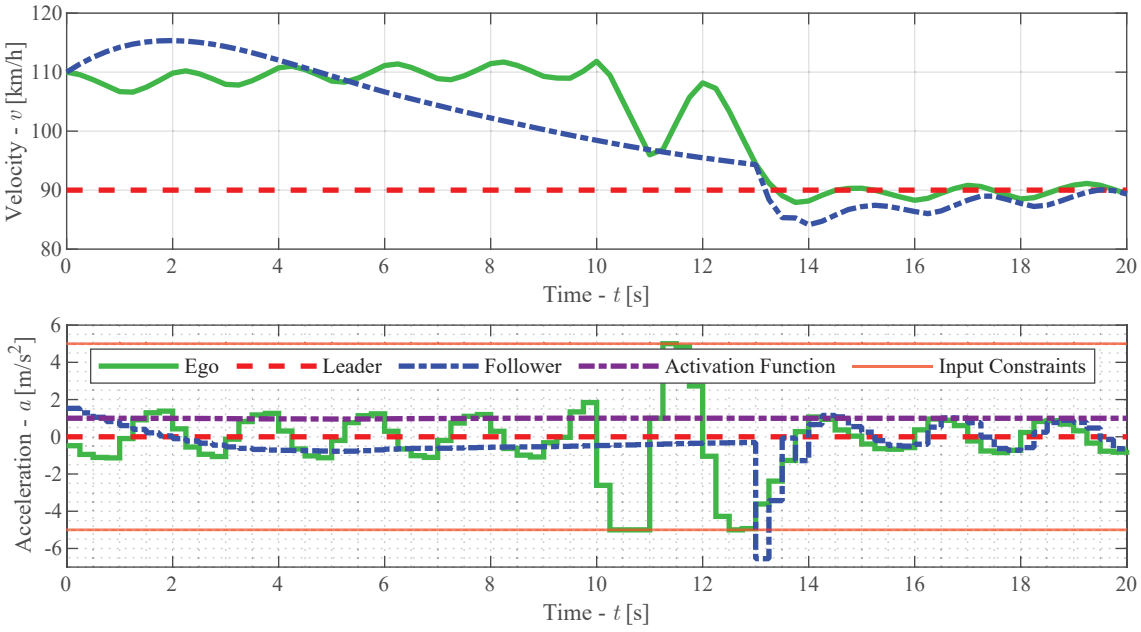

Figure 5.14: Oscillatory velocity and acceleration trajectories of the pre-trained active learning SPGP-MPC on the primary lane merging test case with a prediction horizon of $N = 12$ and a learning period of length $K = 3$.

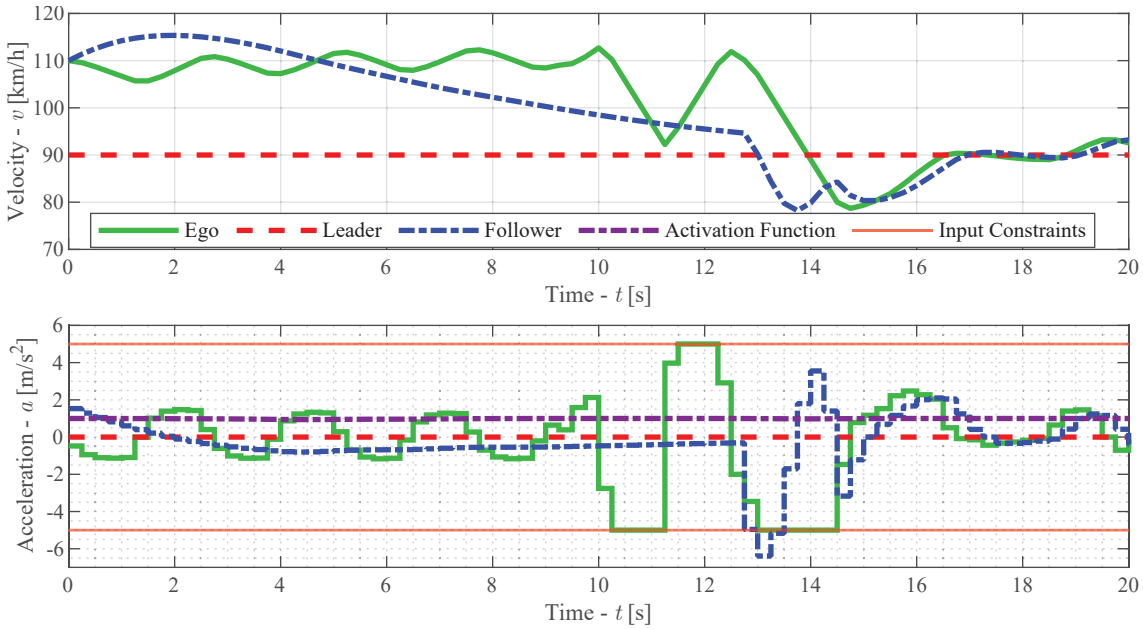

Figure 5.15: Oscillatory velocity and acceleration trajectories of the pre-trained active learning SPGP-MPC on the primary lane merging test case with a prediction horizon of $N = 12$ and a learning period of length $K = 5$.

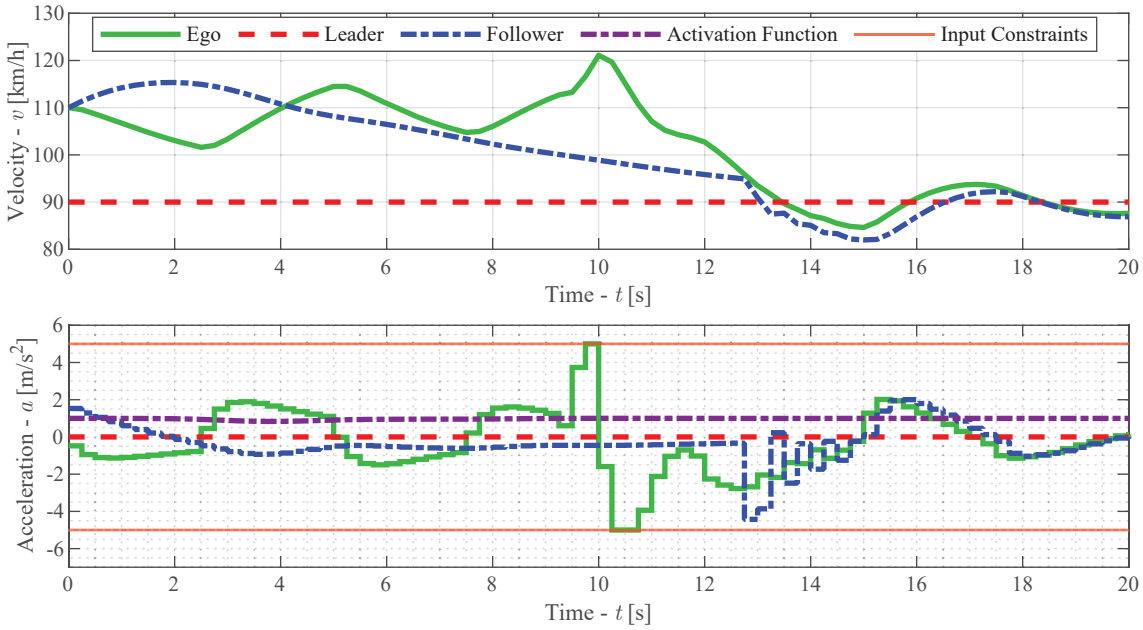

Figure 5.16: Oscillatory velocity and acceleration trajectories of the pre-trained active learning SPGP-MPC on the primary lane merging test case with a prediction horizon of $N = 12$ and a learning period of length $K = 10$.

Table 5.7: Pre-trained active learning SPGP-MPC for various learning period lengths

N	K	Result	$\epsilon_{\max}$ [−]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	Note
12	1	Merged Behind	0	115	71	3.4	5.0	5.8	Oscillatory Behavior
12	3	Merged in Between	0.34	115	84	5.0	6.6	3.8	Moderate brake Follower
12	5	Merged in Between	0.46	115	78	5.0	6.4	3.4	Moderate brake Follower
12	10	Merged in Between	0.29	121	82	5.0	5.0	4.2	-

Table 5.7 summarizes the results of the active learning SPGP-MPC for various learning periods. For more detailed results, refer to Table B.6. The standard implementation of the pre-trained SPGP-MPC ($K = 1$) causes oscillatory behavior that prevents the motion planner from active exploration. In addition, the oscillatory input causes the Ego to merge behind the vehicles, where the other algorithms where successful in merging in between. The novel active learning SPGP-MPC algorithm enables active exploration and manages to successfully merge in between the vehicles. The longest learning period ($K = 10$) leads to the smoothest merge and the least intervention from the Follower, as seen in Figure 5.16.

Computation Time The average solve time of the warm-started, pre-trained active learning SPGP-MPC over all simulations is 0.604 [s]. The maximal solve time over all simulations is 1.030 [s]. The build time and the solve time is not affected by the learning period length K. Refer to Table B.6 for details on each simulation.

Conclusion In section 5.3.2, it has been shown that iterative learning can improve the performance of the motion planner by using offline training data to pre-train the GP model. However, the active learning SPGP-MPC with the standard SPGP implementation causes oscillatory behavior of the motion planner when it is pre-trained on offline data. The set of pseudo-inputs is used to condition the prediction on the previous MPC predictions in an attempt to reduce the complexity associated with full GPs, discussed in section 3.3.4. As the pseudo-inputs change every MPC step, so does the distribution of the GP. Consequently, when the active SPGP-MPC tries to maximize the covariance, it switches to another input sequence, whenever the pseudo-inputs change. The resulting oscillatory behavior of the active learning SPGP-MPC can be rectified by updating the pseudo-inputs every K time steps and enables active learning with a pre-trained SPGP. In the next subsection, we study iterative learning using this novel active learning-based SPGP-MPC algorithm.

5.4.3 Iterative Active Learning

Experiments As with passive learning, we can use observations from past scenarios to pre-train the GP. Subsequently, we study how this pre-training affects the overall performance of the motion planner. Also here, we consider the same primary test case that is

detailed in Table 5.1, with the I-MR-IDM according to Table 2.2 and Table 5.2. The hyperparameters of the active learning framework are selected in section 4.3.2. Similar to section 5.3.2, we consider iterative learning, such that the GP model is pre-trained with the observations $\mathcal{D}_0$ from the same scenario with identical parameterization, including the prediction horizon length N and learning period length K. We confine ourselves to a single prediction horizon of length $N = 12,\ 14,\ 16$.

Simulation Results Firstly, we discuss the pre-trained active learning SPGP-MPC for a prediction horizon length of $N = 12$. This scenario was briefly discussed in section 5.4.2, however, there the focus is on the oscillatory behavior. In this section, we compare it against the other MPCs. The SPGP-MPC actively explores, as seen in Figure 5.15. In comparison to the untrained active learning SPGP-MPC, the pre-trained SPGP-MPC approaches at a lower velocity, as seen in Figure 5.17 and Figure 5.18 at $t = 9.75$ [s]. Consequently, the solution to the learning MPC problem decides to decelerate and subsequently accelerate, whereas the primary MPC plans to gradually reduce the Ego's speed (Figure 5.17). Due to its lower approaching speed, Ego has to accelerate during the merge. Note that the predictions are less accurate (Figure 5.19) than for the passive learning counterpart, found in Figure 5.6. Subsequently, the Ego decelerates strongly after the merge ($t = 13$ [s], Figure 5.18) and induces a strong deceleration on the Follower, as seen in Figure 5.12.

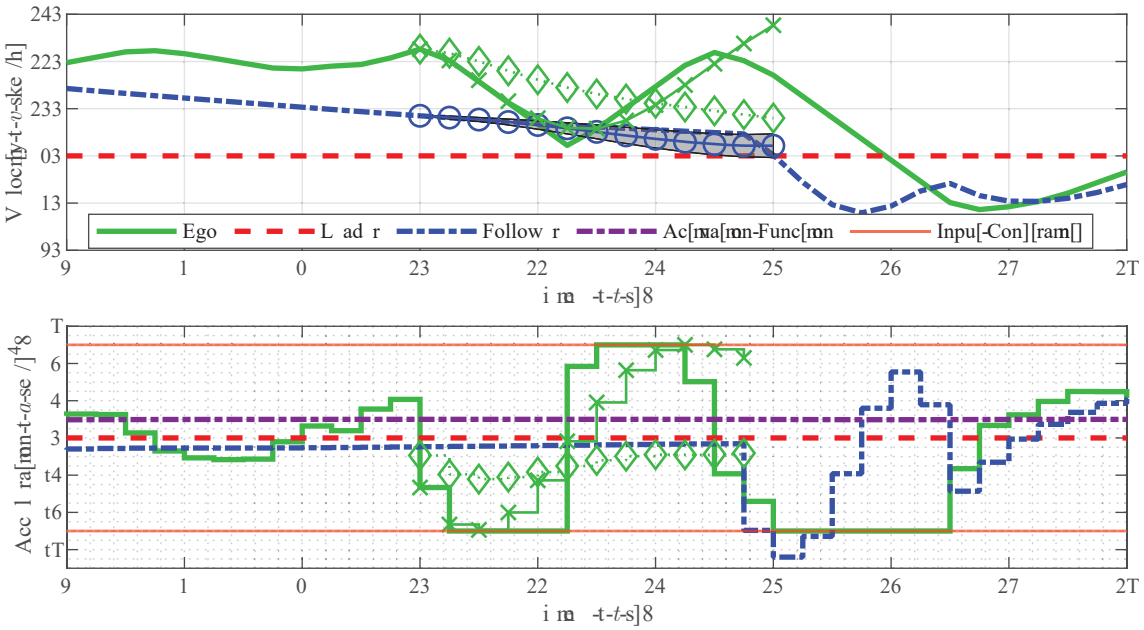

Figure 5.17: Velocity and acceleration trajectories with the predictions of primary (diamonds) and learning (crosses) MPC with one iteration of pre-training at $t = 10$ [s] on the primary lane merging test case with a prediction horizon of $N = 12$.

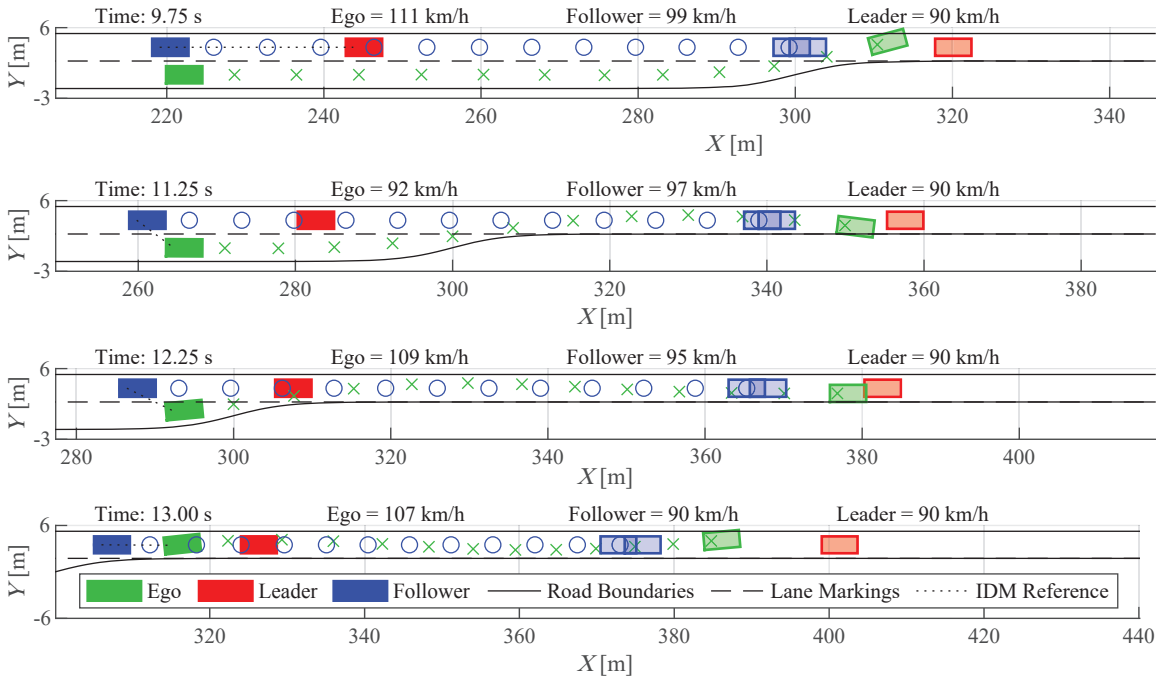

Figure 5.18: Time-lapse of active learning SPGP-MPC with one iteration of pre-training on the primary lane merging test case with a prediction horizon of $N = 12$.

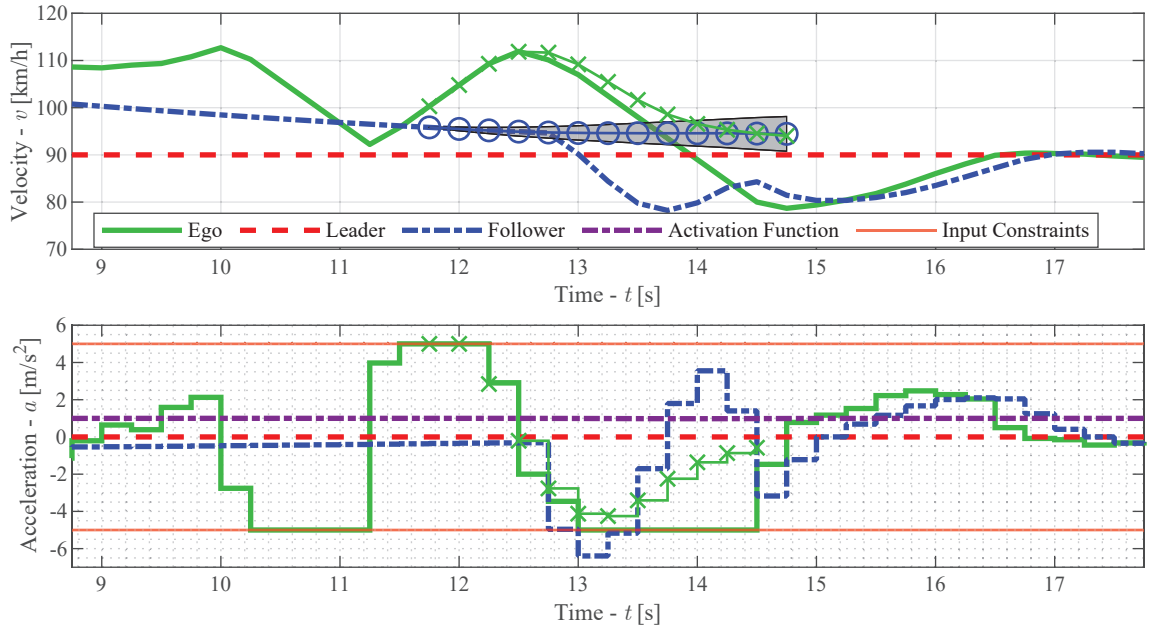

Figure 5.19: Velocity and acceleration trajectories with the predictions of active learning SPGP-MPC with one iteration of pre-training at $t = 11.75$ [s] on the primary lane merging test case with a prediction horizon of $N = 12$.

Table 5.8: Results of active learning SPGP-MPC with pre-training

N	Training Iteration	Result	$\epsilon_{\max}$ [−]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	Note
12	1	Merged in Between	0.46	115	78	5.0	6.4	3.4	Moderate brake Follower
12	2	Merged in Between	0.44	115	78	2.6	5.0	3.0	-
14	1	Merged Behind	0	115	70	4.0	4.6	10.7	-
14	2	Merged in Between	0.35	118	85	5.0	5.3	3.7	-
16	1	Merged Behind	0	115	66	4.8	4.3	6.4	-
16	2	Merged Behind	0	115	77	3.2	4.3	8.1	-

The results of active learning with the SPGP-MPC with pre-training in the primary test case are summarized in Table 5.8. For more detailed results, refer to Table B.7. Whereas pre-training with the passive learning SPGP-MPC resulted in a significant overall performance increase, the effect of pre-training the active learning SPGP-MPC is inconclusive. For $N = 12$, the pre-trained active exploration does not result in significant improvements in comparison to the pre-trained passive learning SPGP-MPC. During the first iteration with $N = 14$, the pre-trained active learning SPGP-MPC had to merge behind as the exploration put the Ego at a disadvantage and caused him to merge behind the two vehicles. However, during the second iteration, the reduced covariance enables Ego to merge behind. For $N = 16$ the active exploration does not lead to any increase in performance. Overall, the pre-trained active learning SPGP-MPC explores less than the untrained SPGP-MPC.

During the second iteration, the Ego approaches at the same speed as in the first iteration ($t = 9.75$ [s] in Figure 5.20). However, in this case, the GP's predictions have improved and the GP accurately captures the interaction dynamics, as shown in Figure 5.21, and Ego gradually reduces its speed before and during the merge ($t = 11.25$ [s] and $t = 12.25$ [s], Figure 5.20). Consequently, the induced deceleration of the Follower is much lower than during the first iteration. Also, in this case, the pre-trained active learning SPGP-MPC explores less than the untrained counterpart

A possible explanation for the reduced exploration when using a pre-trained GP, is that without any data, the covariance is more defined by the prior distribution, whereas the covariance of the pre-trained GP is more defined by the training data rather than the (bounded) prior, such that it requires a lot of relaxation to maximize the uncertainty. As the exploration of the active SPGP-MPC is bounded by a hyperparameter, the SPGP-MPC does not explore as it would with the untrained active SPGP-MPC. Furthermore, since we are solving a non-convex optimization problem, it is likely that we only find a locally optimal solution to the active learning problem. Consequently, it is possible that due to the pre-training, the solution converges to a local minimum that is close to the training data.

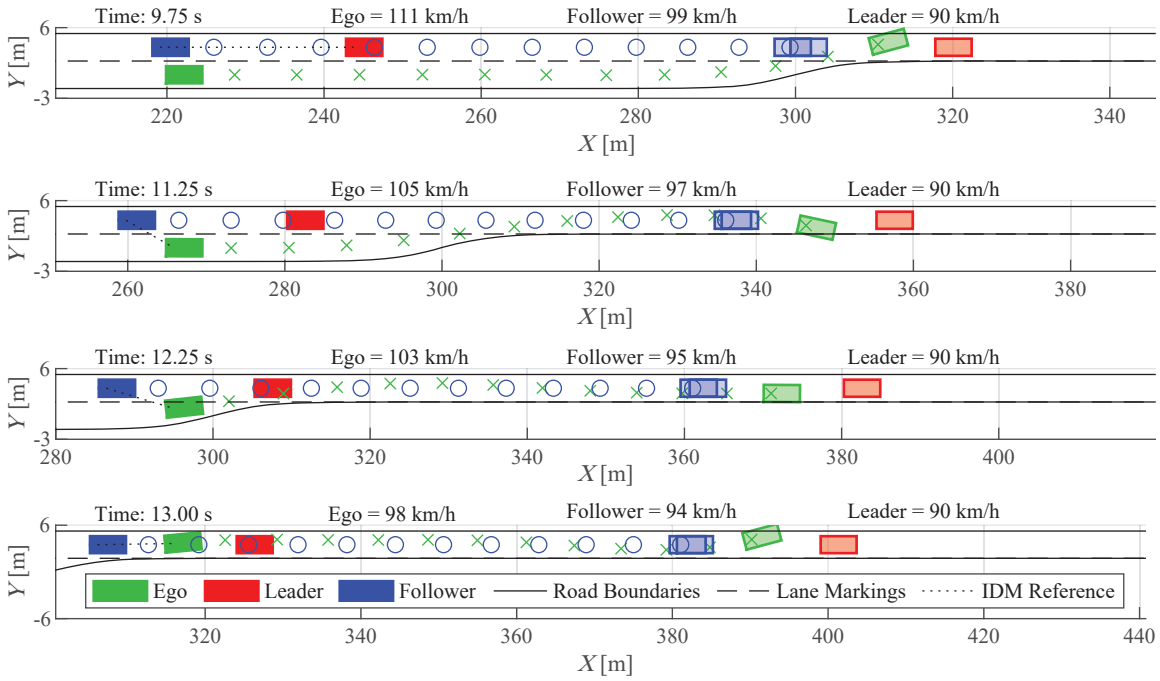

Figure 5.20: Time-lapse of active learning SPGP-MPC with two iterations of pre-training on the primary lane merging test case with a prediction horizon of $N = 12$.

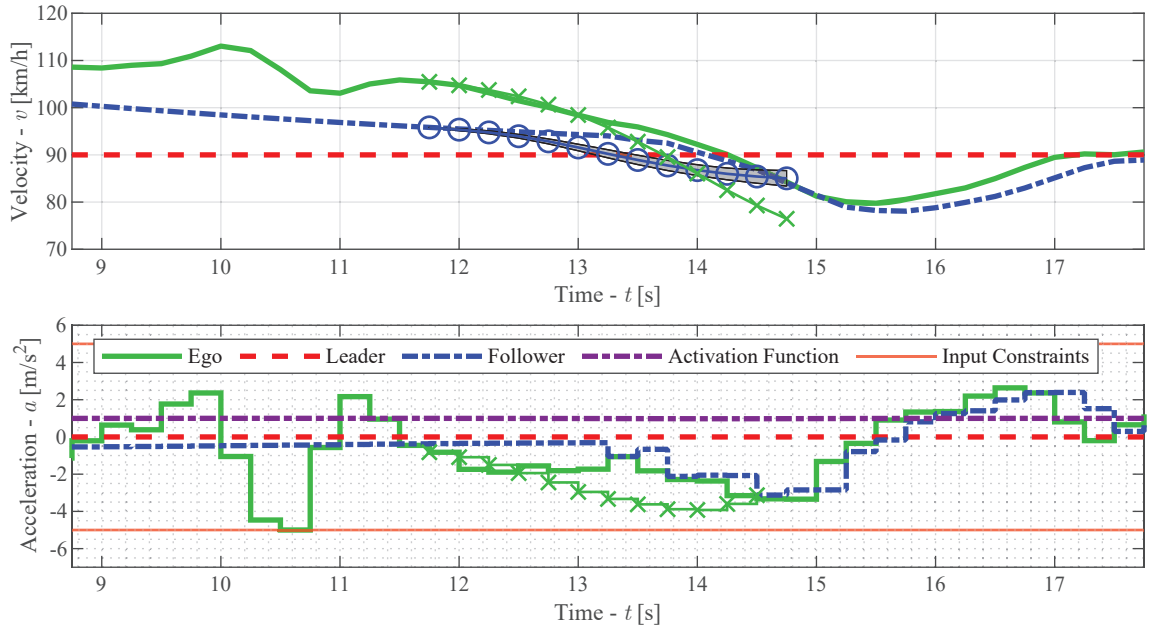

Figure 5.21: Velocity and acceleration trajectories with the predictions of active learning SPGP-MPC with two iterations at $t = 11.75$ [s] of pre-training on the primary lane merging test case with a prediction horizon of $N = 12$.

Computation Time The average solve time of active learning SPGP-MPC with pre-training over all simulations, excluding every first iteration, is 0.740 [s]. The maximal solve time of all simulations is 3.327 [s]. Future research has to determine how the solve time of the active learning SPGP-MPC scales with large amounts of pre-training, still these solve times demonstrate the potential to run this algorithm in real-time. Refer to Table B.7 for details on each simulation.

Conclusion In section 5.4.1, we found that the active SPGP-MPC without pre-training can promote the exploration of the state space and is able to improve the performance of the motion planner. It was found that pre-training confines the active learning solution much more to the primary solution, which potentially results from the convergence of the non-convex optimization problem to a local minimum. The SPGP-MPC still explores the uncertainty around these predictions, but to a lesser extent than without pre-training.

Pre-training the active SPGP-MPC with such actively learned data was not found to improve the performance of the motion planner, as the trajectories are less exploratory than without pre-training. Furthermore, iterative learning does not lead to the same consistent performance increase as seen with passive learning. In conclusion, active learning SPGP-MPC can be used to explore the state space, and has shown increased performance in some cases. However, merely maximizing uncertainty does not provide the required data nor steers the Ego to the appropriate state to consistently outperform online passive learning in this case study. We expect that active learning needs to be done in a truly control-oriented manner to consistently outperform passive learning.

5.5 Generalizability

In section 5.2 to section 5.4, we have focused on the primary test case, which is defined in section 5.1, to provide a detailed comparative study of the various motion planners. Conversely, this section considers different test cases to study the generalizability of the motion planners to provide a sense of their robustness. As the working mechanisms of the various algorithms are similar to those discussed in the previous section, we focus in this section on the high-level performance. Firstly, section 5.5.1 studies the performance of the CV-MPC, as well as the passive and active SPGP-MPC for two different initial conditions. Secondly, we study the performance of the various motion planners in two test cases that concern a more altruistic Follower that yields to the Ego vehicle, in section 5.5.2. Finally, we study the effect of the slack penalty weight on the primary use case, in section 5.5.3.

5.5.1 Different Initial Conditions

In this alternative use, we study a lane merging scenario with two different initial conditions to test the robustness of the motion planners. It is important to note that the parameterization of the objective function has not been modified from the primary test case. The Ego vehicle tries to maintain its initial velocity, while the closing merge lane incentivizes the Ego

Table 5.9: Parameters of the Interactive MR-IDM of the adversarial Follower.

IDM Parameter	Symbol	Value
Nominal reference velocity	v_{nom}	110 [km/h]
Nominal headway time	T_{nom}	1 [s]
Interactive reference velocity	v_{act}	140 [km/h]
Interactive headway time	T_{act}	0.25 [s]

to safely merge. The lateral positions are equal to those in Table 5.1. Again, note that all vehicles start with zero heading angle (ψ) and zero steering angle (δ). The I-MR-IDM of the Follower has the same parameterization as in the primary test case that we have seen thus far, and is listed in Table 2.2 and Table 5.9. We confine ourselves to a prediction horizon of length $N = 12$ and a learning period of length $K = 5$, in the case of active learning.

Test Case 1 Firstly, we consider a test case where the vehicles are initialized closer to the end of the merge lane, detailed in Table 5.10. Table 5.11 summarizes the results of the CV-MPC, and the passive and active learning SPGP-MPC in the second alternative test case. For more detailed results, refer to Table B.8. While in the previous test case the CV-MPC was able to safely merge in between, in this case, the CV-MPC does not find a gap and safely merges behind. The passive learning SPGP-MPC can learn the behavior but is unable to find a gap to safely merge in between and, hence, also safely merges behind the two vehicles. Conversely, the active learning SPGP-MPC safely merges in between the two vehicles as a result of its active exploration. Note that for active learning the upper bound on the relaxation is reduced to $\gamma_{\max} = 100$, as the nominal value of $\gamma_{\max} = 250$ resulted in Ego overtaking both vehicles with high velocity. Further improvements in the adaptability of the active learning framework are considerations for future work.

Table 5.10: Initial conditions of alternative test case 1.

State	Symbol	Value
Longitudinal pos. Ego	X^0	75 [m]
Longitudinal pos. Follower	X^1	75 [m]
Longitudinal pos. Leader	X^2	130 [m]
Velocity Ego	v^0	115 [km/h]
Velocity Follower	v^1	115 [km/h]
Velocity Leader	v^2	90 [km/h]

Table 5.11: Results of the alternative test case 1 for $N = 12$.

Algorithm	Result	$\epsilon_{\max}$ [−]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	Note
CV-MPC	Merged Behind	0.04	111	74	1.9	4.5	6.1	-
Passive SPGP-MPC	Merged Behind	0.18	111	75	1.7	4.6	6.0	-
Active SPGP-MPC	Merged in Between	0.00	116	87	1.4	3.7	5.3	$\gamma_{\max} = 100$

Test Case 2 Secondly, we consider a test case where the Ego approaches the Follower and the Leader who are both driving at a lower speed, detailed in Table 5.12. Similarly, the results of the CV-MPC, and the passive and active learning SPGP-MPC in the second alternative test case are summarized in Table 5.13. For more detailed results, refer to Table B.9. Similar to the previous test case, the CV-MPC is too optimistic and tries to merge in between, causing a collision with the Follower. In this case, the passive learning SPGP-MPC successfully learns the behavior of the Follower and safely merges in between the two target vehicles. Moreover, the active learning SPGP-MPC safely merges in between the two vehicles as a result of its active exploration. However, this active exploration leads to increased intervention by the Follower.

Table 5.12: Initial conditions of alternative test case 2.

State	Symbol	Value
Longitudinal pos. Ego	X^0	-100 [m]
Longitudinal pos. Follower	X^1	-50 [m]
Longitudinal pos. Leader	X^2	0 [m]
Velocity Ego	v^0	115 [km/h]
Velocity Follower	v^1	90 [km/h]
Velocity Leader	v^2	90 [km/h]

Table 5.13: Results of the alternative test case 2 for $N = 12$.

Algorithm	Result	$\epsilon_{\max}$ [–]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	Note
CV-MPC	Merged Behind	0.04	115	75	2.3	5.0	5.0	-
Passive SPGP-MPC	Merged in Between	0	115	87	1.8	3.4	5.7	-
Active SPGP-MPC	Merged in Between	0.24	122	83	5.0	5.0	4.2	$\gamma_{\max} = 100$

Conclusion In addition to the initial condition that is studied in detail in the previous sections, we study two additional initial conditions to assess the generalizability of the CV-MPC, and the SPGP-MPC with both passive as well as active learning. The SPGP-MPC was able to successfully learn the behavior of the Follower and identify a gap that the CV-MPC could not. Consequently, the SPGP-MPC was able to merge in between the vehicles, whereas the CV-MPC had to merge behind. In some cases, the active SPGP-MPC causes less intervention of the Follower than the passive SPGP-MPC, however, this strongly depends on the initial condition. In conclusion, the performance of the SPGP-MPC in these various initial conditions demonstrates the generalizability of the motion planner and shows its potential and robustness. However, the adaptability of the active learning SPGP-MPC requires further research to adapt its degree of learning.

5.5.2 Altruistic Driving Behavior

Experiments Subsequently, we consider an altruistic Follower that is more likely to yield to the Ego vehicle that is trying to merge. However, this behavior should still be sufficiently challenging. As such, the Follower still tries to close the gap with the Leader. However, the Follower is more altruistic and will yield to the Ego if he tries to overtake. Similarly to test cases 2 and 3, we adjust the active learning parameter to $\gamma_{\max} = 100$ to limit the exploration to prevent the Ego from overtaking both vehicles. The desired speeds and headway times of the Interactive MR-IDM are listed in Table 5.14. To this end, we consider two different initial conditions in this study.

Table 5.14: Parameters of the Interactive MR-IDM of the altruistic Follower.

IDM Parameter	Value
Nominal speed (v_{nom})	115 [km/h]
Nominal headway time (T_{nom})	0.25 [s]
Interactive speed (v_{act})	115 [km/h]
Interactive headway time (T_{act})	1 [s]

Test Case 3 Firstly, we consider a test case where the Ego approaches the Follower and the Leader who are both driving at a lower speed, detailed in Table 5.15. The results of the CV-MPC, and the passive and active learning SPGP-MPC in the fourth alternative test case are summarized in Table 5.16. For more detailed results, refer to Table B.10 in Appendix B. Again, the CV-MPC is able to safely merge in between. Table 5.16 shows that the minimal velocity is similar for all three algorithms. Although the KPIs are similar, the SPGP-MPC has a much smoother trajectory as it anticipates the behavior of the Follower.

Table 5.15: Initial conditions of alternative test case 3 with an altruistic driver.

State	Symbol	Value
Longitudinal pos. Ego	X^0	-100 [m]
Longitudinal pos. Follower	X^1	-50 [m]
Longitudinal pos. Leader	X^2	0 [m]
Velocity Ego	v^0	115 [km/h]
Velocity Follower	v^1	90 [km/h]
Velocity Leader	v^2	90 [km/h]

Table 5.16: Results of the alternative test case 3 with an altruistic driver for $N = 12$.

Algorithm	Result	$\epsilon_{\max}$ [−]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	Note
CV-MPC	Merged in Between	0	115	83	2.4	2.6	8.8	-
Passive SPGP-MPC	Merged in Between	0	115	83	2.4	2.2	7.2	-
Active SPGP-MPC	Merged in Between	0.02	123	82	2.4	4.4	5.7	$\gamma_{\max} = 100$

Test Case 4 Secondly, we consider the case where the Follower is approaching the Leader at a higher velocity, while the Ego is approaching both the Follower and the Leader at an even higher velocity, seen in Table 5.17. The results of the CV-MPC, and the passive and active learning SPGP-MPC in the final alternative test case are summarized in Table 5.18. For detailed results, refer to Table B.11. The active learning SPGP-MPC approaches the merge at a higher velocity, and consequently, it has to brake more during the merge. This results in a larger induced deceleration of the Follower than the CV-MPC and the passive SPGP-MPC. Although the minimal gap attained with the passive and active learning SPGP-MPC is larger than for the CV-MPC, the performance of all three algorithms is similar.

Table 5.17: Initial conditions of alternative test case 4 with an altruistic driver.

State	Symbol	Value
Longitudinal pos. Ego	X^0	0 [m]
Longitudinal pos. Follower	X^1	50 [m]
Longitudinal pos. Leader	X^2	100 [m]
Velocity Ego	v^0	125 [km/h]
Velocity Follower	v^1	110 [km/h]
Velocity Leader	v^2	90 [km/h]

Table 5.18: Results of the alternative test case 4 with an altruistic driver for $N = 12$.

Algorithm	Result	$\epsilon_{\max}$ [−]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	Note
CV-MPC	Merged in Between	0	125	83	0.6	2.4	7.8	-
Passive SPGP-MPC	Merged in Between	0	125	83	0.2	2.4	6.9	-
Active SPGP-MPC	Merged in Between	0.01	132	83	0.8	3.0	5.9	$\gamma_{\max} = 100$

Conclusion In addition to adversarial drivers, the SPGP-MPC is capable of anticipating the behavior of more altruistic drivers. Although the CV-MPC yields satisfactory results when faced with an altruistic driver that yields to the Ego, the SPGP-MPC can predict this behavior accurately and improve the smoothness of the control inputs, compared to CV-MPC. In the studied scenarios, active exploration does not provide a strong advantage over passive learning SPGP-MPC. However, it can be concluded that the SPGP-MPC also can improve performance in less challenging scenarios, like that of a yielding driver.

5.5.3 Adjusted Slack Penalty Weights

Experiments In this section, we identify an edge case in which the CV prediction model falls short, while the SPGP-MPC can successfully identify a gap. We study the CV-MPC and the online passive and active learning SPGP-MPC for the primary test case defined in section 5.1. The initial conditions for this study are defined in Table 5.1, and the I-MR-IDM is parameterized according to Table 2.2 and Table 5.2. For the SPGP-MPC, we focus on passive online learning, such that the GP starts without any training data ($\mathcal{D}_0 = \emptyset$).

We increase the penalty of the slack variables with a factor of 2.5 to create this edge case:

$$\rho = 2.5 \begin{bmatrix} 10^5 & 10^5 & 10^3 & 10^3 \end{bmatrix}^\top. \tag{5.1}$$

Results The results of the CV-MPC, and the passive and active learning SPGP-MPC are summarized in Table 5.19. For more detailed results, refer to Table B.12 in Appendix B. As a consequence of the increased penalty on the slack variable, the CV-MPC is unable to identify the gap, whereas the SPGP-MPC learns the behavior online and can successfully identify a gap and merge in between the vehicles. Furthermore, the active learning MPC leverages its exploration and induces a lower deceleration on the Follower. Also, note that the instantaneous slack is lower for active learning than for passive learning.

We compare the CV-MPC and passive learning SPGP-MPC in more detail. The CV-MPC is unable to identify a gap and brakes very lightly, assuming that the Follower will maintain its current velocity, as seen in Figure 5.23. As the CV-MPC does not anticipate the Follower braking, it maintains its speed as seen at $t = 9.75$ [s] in Figure 5.22. Consequently, it has to brake hard in order to prevent a collision with the Follower ($t = 11.25$ [s] and $t = 12.25$ [s]), and subsequently, it merges behind the Follower with a close distance to the Follower, as seen at $t = 13$ [s] in Figure 5.22. The SPGP-MPC, on the other hand, predicts that the Follower will brake. It is able to identify a gap in which the Ego can merge, while accounting for uncertainty in its predictions, as seen at $t = 9.25$ [s] in Figure 5.24, and $t = 9.75$ [s] in Figure 5.25. Just before starting the merge, the Ego briefly accelerates to maintain sufficient distance from the Follower ($t = 11.25$ [s], Figure 5.25). Subsequently, the Ego has to reduce its speed, inducing a moderately hard brake on the Follower.

Conclusion The parametrization of the problem plays a crucial role in the performance of the motion plan. As the weights on the slack variable limit the relaxation of collision avoidance constraints, the prediction models must rely on accurate predictions to identify a gap. Note that the instantaneous slack of the SPGP-MPC reduced from $\epsilon_{\max} = 0.52$ (Table 5.4) to $\epsilon_{\max} = 0.23$ (Table 5.19) after increasing the slack penalty. The SPGP-MPC can successfully learn the behavior of the Follower and leverage these predictions to complete a lane merge, while the CV-MPC cannot identify a gap and merges behind the Follower. In conclusion, the SPGP-MPC outperforms the CV-MPC in these tightly constrained scenarios. Although this test case shows promise for SPGP-MPC as an interactive motion planner, further research has to determine the true potential of GP-MPC motion planners.

Table 5.19: Results of the primary test case with increased slack penalties for $N = 12$.

Algorithm	Result	$\epsilon_{\max}$ [−]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	Note
CV-MPC	Merged Behind	0	115	68	1.7	5.0	6.8	-
Passive SPGP-MPC	Merged in Between	0.23	115	85	1.5	6.3	4.3	Moderate brake Follower
Active SPGP-MPC	Merged in Between	0.09	120	84	2.8	5.1	4.7	$\gamma_{\max} = 250$

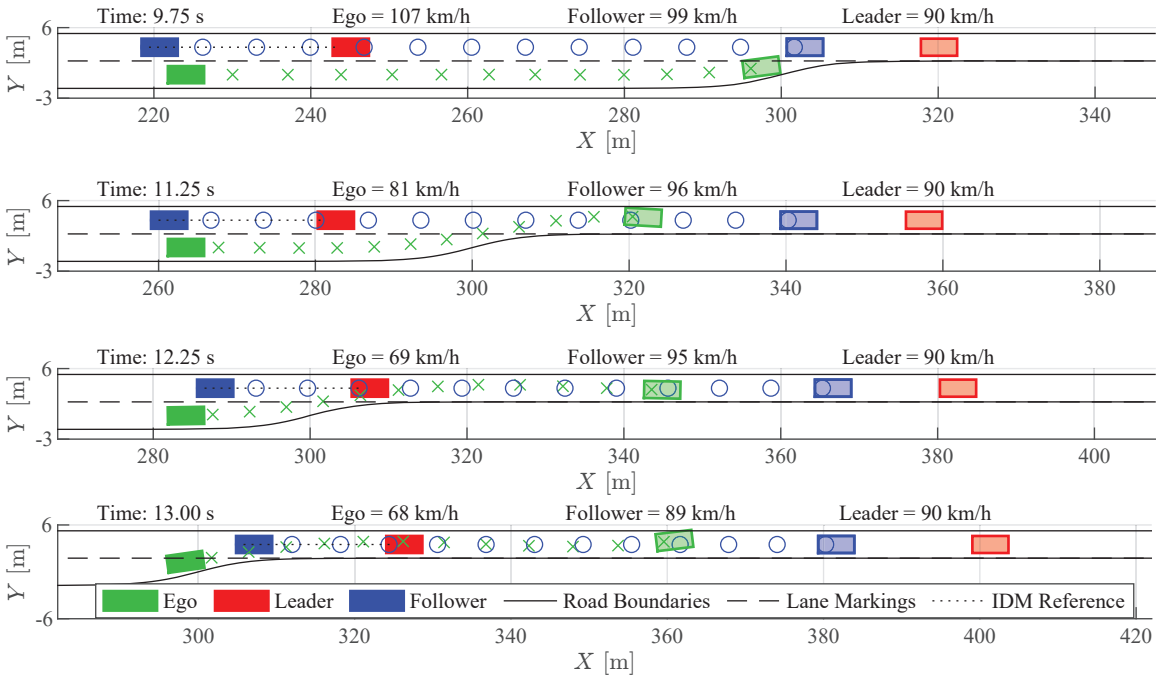

Figure 5.22: Time-lapse of CV-MPC on the primary lane merging test case with increased slack penalty weight and a prediction horizon of $N = 12$.

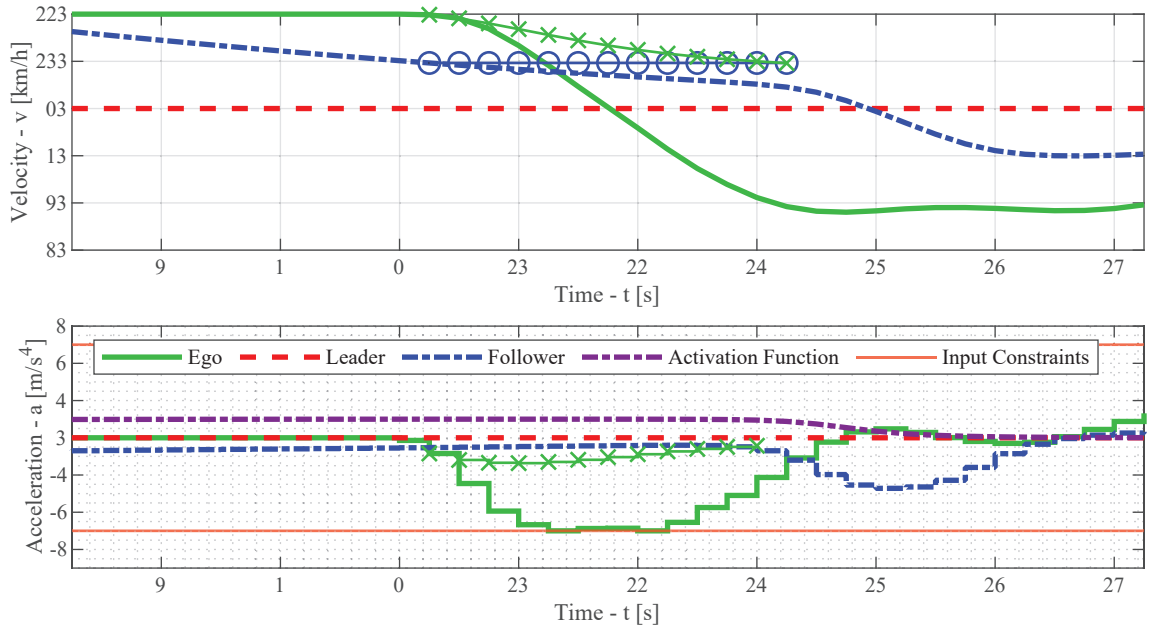

Figure 5.23: Velocity and acceleration trajectories with the predictions of the CV-MPC at $t = 9.25$ [s] on the primary lane merging test case with increased slack penalty weight and a prediction horizon of $N = 12$.

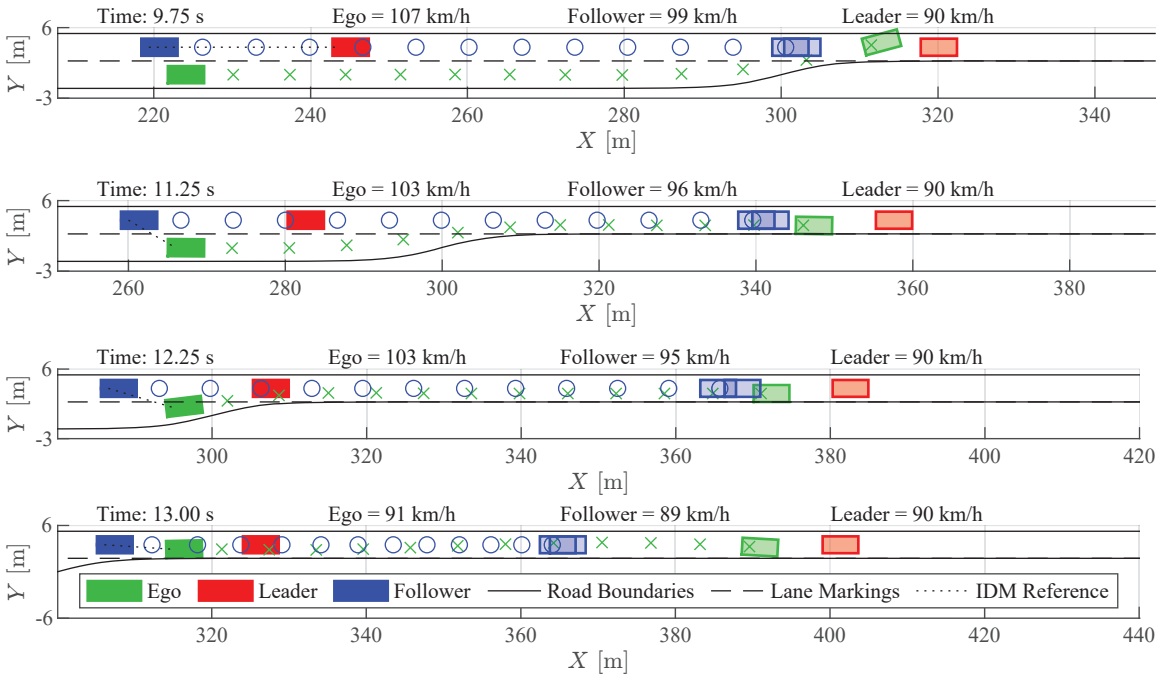

Figure 5.24: Time-lapse of passive learning SPGP-MPC on the primary lane merging test case with increased slack penalty weight and a prediction horizon of $N = 12$.

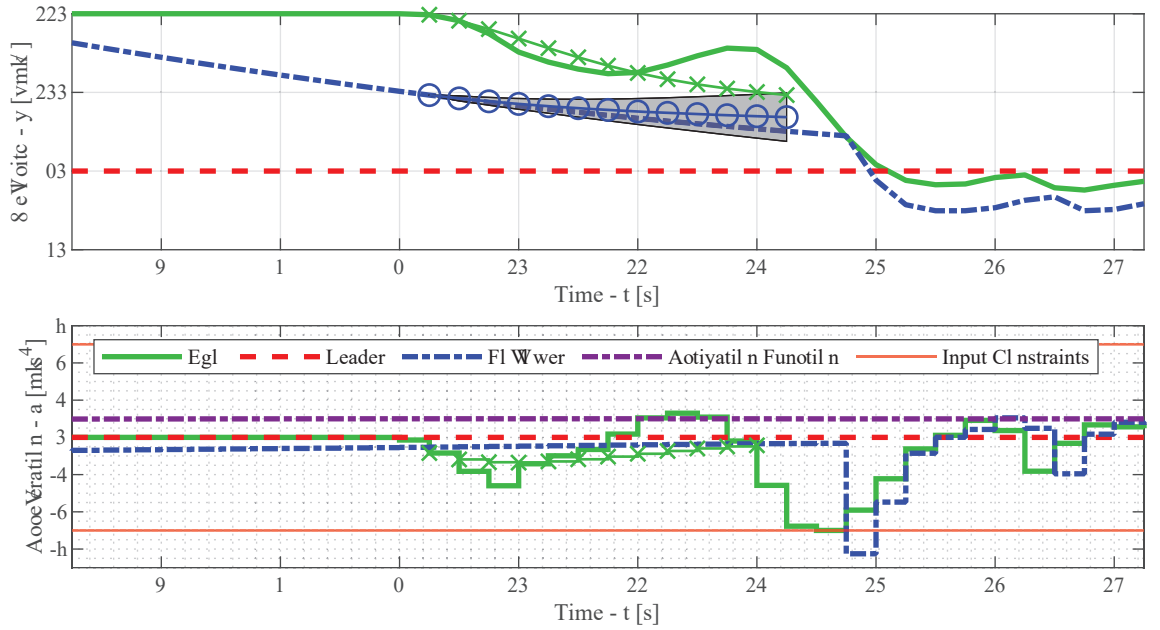

Figure 5.25: Velocity and acceleration trajectories with the predictions of the passive learning SPGP-MPC at $t = 9.25$ [s] on the primary lane merging test case with increased slack penalty weight and a prediction horizon of $N = 12$.

5.6 Discussion

In section 5.2 to section 5.5, the results of the CV-MPC and the passive as well as active learning SPGP-MPC are presented and discussed. In this section, we conclude on these results and discuss the findings of the simulation studies on the lane merging test cases.

5.6.1 Constant-Velocity MPC

The CV-MPC uses a deterministic constant-velocity prediction model to predict the motion of the Following vehicle. While in many cases such a prediction model will suffice, it limits the possibilities to learn and adapt to uncertain and unseen behavior. In our primary test case, the CV-MPC was able to safely merge between the vehicles with short horizons. This behavior results from the overconservative predictions that assume that the Follower is driving at a constant velocity, while actually it is braking. For longer horizons, the CV-MPC was unable to identify a gap. Due to the current parameterization, the CV-MPC was able to safely merge. However, preliminary results show that its constant velocity predictions can fall short. The qualitative predictions are often inaccurate and do not capture the dynamics of the interactive agents, limiting the adaptability to interact with them. Furthermore, an interaction-aware model is essential to enable active learning.

5.6.2 Passive Learning with SPGP-MPC

The SPGP-MPC can anticipate the behavior of the Following vehicle by leveraging past observations, and adapt to the current situation. The simulation studies show that the performance of the passive learning SPGP-MPC is robust to different horizon lengths. Moreover, the SPGP-MPC can successfully learn the interactions and safely plan the Ego's motion in a variety of cases, showing a degree of robustness of the method. The SPGP can adapt its predictions and the associated covariance based on the training data. As a result, the GP-based prediction model, on the one hand, takes more caution when the scenario is unknown or has great uncertainty. As such, long prediction horizons feature greater covariance, accounting for the uncertainty associated with those predictions and taking caution when necessary. However, it can be more assertive if the scenario is familiar and the GP is confident in its predictions. In conclusion, the SPGP-MPC has strong potential to learn the interactive behavior of other vehicles both with and without pre-training.

5.6.3 Active Learning with SPGP-MPC

Soloperto *et al.* [13] provide a simple and intuitive, yet powerful framework to extend a MPC with active learning. In this study, we exploit this framework using the GP-based MPC. The active learning framework enables active and tunable exploration of the state space. Although this exploration can improve the performance of the motion plan, the relaxation of the primary MPC problem is constant (through $\gamma_{\max}$) in our current formulation. Ideally, the active learning framework should allow more relaxation (through $\bar{\beta}$ and $\beta_{\max}$) if it

expects to improve the primary objective in the long run. We expect that due to our problem formulation, we cannot exploit the active learning framework to its full potential.

The active learning framework uses notions from economic MPC to enforce an absolute or average decrease of the objective function over time. As such, the active learning framework relies on an MPC problem whose objective function is — at least on average — decreasing, for example as is typically found in tracking problems. As mentioned before, for the sake of analysis we do not incentivize the AV to merge, firstly, as to prevent any bias of the solution, and secondly, to identify the shortcomings and potential of the various motion planners. Due to our current problem formulation, the MPC's objective function is very low until we have to deviate from our reference, that is, as the Ego vehicle approaches the merge point. Consequently, it is very difficult to enforce the absolute or average decrease of the objective function over the horizon over time. It is important to note that the primary problem remains feasible, however, the learning problem that is to be solved subsequently becomes infeasible as it cannot satisfy the cost decrease. Hypothetically, in a dense traffic scenario, we can incentivize the AV to merge into the target lane. The objective function is initially high, due to the cost of not merging, and by exploring the state space it perturbs the target vehicles which may or may not yield to the AV that is trying to merge. During this exploration, the GP-MPC could leverage the observations on these interactions to merge in between. Conclusively, due to the current construction of the MPC its objective function, the active learning framework by [13] could not be exploited to its full potential.

5.6.4 Deterministic vs. Stochastic Predictions

In this thesis, the CV-MPC that serves as a baseline for the passive and active learning SPGP-MPC uses a deterministic prediction model, whereas the SPGP-MPC relies on a stochastic prediction model. One could argue that a stochastic CV prediction model could account for the inaccuracy and uncertainty of its predictions and, hence, improve the generalizability and safety of the CV-MPC. However, we would argue that the primary limitation is not the uncertainty of a potential stochastic CV prediction model. Merely adding some bounded uncertainty to the CV predictions is equivalent to a reformulation of the CV-MPC problem, in which the collision avoidance constraints perhaps become tighter over the prediction horizon. As seen in section 5.5.3, the CV-MPC is unable to identify a gap in tight constraints and consequently has to merge behind. Moreover, the approximate distribution of the uncertainty will strongly dictate the behavior of the CV-MPC and it cannot be adapted to unseen behavior. We believe that the primary limitation of the CV model is the lack of adaptability. Although a stochastic CV-MPC could improve safety as the resulting distribution captures more potential trajectories, this can also be a significant limitation in cases that require an interactive and *assertive* motion plan, rather than a conservative motion plan. Conversely, the SPGP-MPC can predict the Follower's motion through Bayesian inference using current observations, and adapts its predictions accordingly. As such, the covariance or 'confidence' of the predictions is governed by the GP and therefore is variable. Consequently, the SPGP-MPC has the potential to identify a gap for merging. This adaptability shows the potential of SPGP-MPC as a safer and better motion planner, as seen in

the case study. In conclusion, as opposed to the reactive motion plan of the CV-MPC, the SPGP-MPC provides an interactive and assertive motion plan that anticipates the motion of another vehicle while considering the uncertainty of its predictions.

5.7 Reflection

The results of this study ask for some reflection. In addition to a proof of concept for the learning and prediction capabilities of Gaussian process-based MPC, this thesis also aimed to demonstrate how it can outperform a baseline MPC, a constant velocity MPC. As mentioned before, the performance of the MPC is strongly dependent on the parameterization of the MPC problem. An important aspect of this is the use of slack variables on the softly constrained collision avoidance constraints. As such, the primary and alternative test cases were selected to demonstrate an edge case in which the CV-MPC fails, and the SPGP-MPC is able to leverage its predictions to complete the merging maneuver. Unfortunately, due to a coding error, the simulations of the baseline CV-MPC were flawed. Consequently, these simulations had to be redone after the error was corrected. Tuning the parameters to demonstrate this edge case, and performing all simulation studies again, was not feasible in the remaining time of the project. As seen in this chapter, with the current problem formulation and the corrected results, the quantitative performance of the CV-MPC is comparable to that of the SPGP-MPC. Nevertheless, some alternative test cases demonstrate where the CV-MPC is lacking, as seen in section 5.5. Furthermore, the qualitative results of the online learning SPGP-MPC show great potential that asks for further research.

In addition, we simulated the primary test case with increased slack penalty weights and demonstrated its potential, in section 5.5.3, to provide a glimpse of the capabilities of SPGP-MPC in such an edge case. During the study, it was found that the penalty weight of the slack variables is a crucial parameter for the solution of the motion planner. The SPGP-MPC outperforms the CV-MPC in its prediction quality, for the CV-MPC merely relies on a constant velocity prediction. As such, the CV-MPC requires more slack to identify a gap as compared to the SPGP-MPC. This can be seen, for example, in the alternative case study in which we increase the slack penalty weights. The CV-MPC is unable to identify a gap without violating the soft constraints and accordingly, has to merge behind the two target vehicles. The SPGP-MPC is able to identify a gap with that same parameterization. Considering the complexity of the problem, it still uses the slack variables. However, with its superior predictions, the SPGP-MPC improves its decision-making and planning capabilities and relies less on soft constraints. Consequently, it can successfully merge between the target vehicles. This is a strong example of how online Gaussian process MPC can outperform a constant velocity MPC. Unfortunately, we can only discuss this phenomenon briefly. Still, the ability to learn these predictions online without the need for pre-training demonstrates the potential of GP-MPC for interaction-aware motion planning. Its prediction capabilities can be leveraged to reduce conservatism and open the door for passive and active learning-based MPC in interaction-driven traffic scenarios. Future research has to validate the true capabilities and impact of GP-MPC on the safety and performance of motion planning.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

Autonomous vehicles operate in complex dynamic environments that are characterized by strong interactions between vehicles and are faced with uncertain and unseen behavior. When navigating in complex traffic scenarios, the AV needs to account for these interactions and their associated uncertainty when planning its motion. In this thesis, interaction-aware motion planning methods using learning-based model predictive control have been presented that enable autonomous vehicles to learn uncertain and unseen behavior of other vehicles in a simulated lane merging scenario. We leveraged MPC to jointly optimize the motion of the AV and the predictions of a target vehicle through an interactive Gaussian process prediction model. Based on quantitative and qualitative analysis, it can be concluded that GP-MPC can passively and actively learn the interactions between two vehicles. The GP-MPC accurately predicts and leverages these interactions to complete a lane merge without the need for pre-training. Simulation results demonstrate its generalizability to various scenarios. In conclusion, GP-MPC shows strong potential as an interaction-aware motion planner for uncertain and unseen traffic scenarios that extend beyond the training data set.

Gaussian process-based MPC has become a popular approach for learning-based control as it provides a direct assessment of the uncertainty of the predictions. As such, GP-based predictions have been used in a stochastic MPC for autonomous overtaking in racing, however, this method does not jointly optimize the motion of the AV and the predictions of the target vehicle. Although, more recently, interaction-aware GP-MPC has been used jointly to optimize the motion of the AV and the predictions of a target vehicle in autonomous racing, they rely on offline training data from observations of similar scenarios. Online learning could improve the generalizability and performance of interaction-aware planning in uncertain and unseen scenarios. We extended this line of research by utilizing both passive and active learning with GP-MPC. By learning the interactions between vehicles online, our motion planner can anticipate uncertain and even unseen behavior. The adaptability of GP-MPC could contribute to the robustness and practical generalizability of motion planning to challenging traffic scenarios that are outside the planner's training data.

This thesis makes the following contributions to the field of interaction-aware motion planning. Firstly, we presented a novel online learning-based Gaussian process model predictive controller that jointly optimizes the motion of the AV and the interacting vehicle. Secondly, we integrated our GP-MPC in the active learning MPC framework from Soloperto *et al.* [13]. Furthermore, a novel GP-MPC motion planning algorithm was presented that enables the use of Sparse GP-based prediction models in the active learning framework [13]. Finally, we proposed an extension to the (Merge-Reactive) Intelligent Driver Model which models interactions through state-dependent parameters of the IDM. This Interactive IDM enables the simulation of complex traffic scenarios characterized by strong interactions.

The performance and potential of our proposed methods were evaluated through simulation studies, and a qualitative and quantitative analysis of the baseline MPC and the learning-based GP-MPCs were conducted. In this study, the Ego vehicle tried to merge into a lane with a Following and a Leading target vehicle. We considered three prediction methods to predict the motion of the Follower, while the Leader maintained a constant velocity. Our baseline CV-MPC uses a deterministic constant velocity model to predict the future velocity of the Follower. Our online learning SPGP-MPC uses a sparse pseudo-input GP to learn the interactions from past and online observations. Through Bayesian inference, SPGP-MPC jointly plans the motion of the Ego and predicts the motion of the Follower while directly assessing the covariance associated with these predictions. The SPGP-MPC can accurately predict the motion of another vehicle while accounting for its joint uncertainty. In addition, the SPGP-MPC can successfully leverage the active learning framework from [13] to explore the state space and improve the performance of the motion plan. The passive and active learning SPGP-MPC are compared with the baseline CV-MPC. It was found that the performance of the CV-MPC is strongly dependent on the problem formulation. Its inferior predictions can pose limitations in tightly constrained problems. Although in many cases, the performance and safety of CV-MPC are adequate, simulations show that in specific edge cases, the constant velocity predictions are lacking. As safety is about the tail of the distribution, it is in these edge cases where we find the shortcomings and strengths of the different methods. In these edge cases, the SPGP-MPC can leverage its predictions to identify a gap in which the Ego vehicle can safely merge. Moreover, preliminary results on active learning showed that SPGP-MPC is promising, yet, further research is required to use the active learning framework [13] with SPGP-MPC to its full potential.

In conclusion, this thesis shows the capabilities of interaction-aware motion planning for a lane merging scenario and demonstrates how online learning SPGP-MPC can learn uncertain and unseen driving behavior without the need for pre-training, thereby extending the potential generalizability of these motion planning methods. However, this thesis merely serves as a proof of concept, and while simulation results demonstrate the potential of SPGP-MPC, additional studies have to determine the capabilities of SPGP-MPC motion planners. While the results of the study hold promise for interaction-aware motion planning with online learning Gaussian process MPC, we identify several directions for future research, in particular for active learning methods as well as theoretical safety guarantees. In the next section, recommendations and an outlook for future work are provided.

6.2 Recommendations

In this thesis, a first step has been made to online learning GP-MPC for interaction-aware motion. Finally, we give recommendations for future further research. The next research steps for the online learning-based GP-MPC include (i) validation through a real-life data set or an interactive simulation environment, (ii) enabling real-time motion planning while accommodating more vehicles and large sets of training data, (iii) advancing the method to operate in denser and more complex traffic scenarios, and (iv) providing theoretical guarantees on recursive feasibility and (probabilistic) safety. This section identifies several directions for these research steps that have been identified during the study.

6.2.1 Validation

A qualitative and quantitative analysis of online learning Gaussian process MPC was conducted to study the mechanisms of the GP-based prediction models and their relevance for interaction-driven traffic scenarios. To this end, small training sets and a limited number of initial conditions have been considered. Although section 5.5 provides a glimpse of how the SPGP-MPC could improve driving behavior in other traffic scenarios, future research could extend this method to various traffic scenarios.

A natural next step would be to validate these methods with a data set with real traffic data or in a high-fidelity simulation environment. A frequently used data set in AV research is the highD data set [44]. A more recent data set, called exiD [45], focuses on entering and exiting scenarios on highways and contains more cut-in scenarios, and has a higher density of interactive and challenging scenarios than highD [45]. Such data sets could be used to validate the accuracy and relevance of the simulation scenarios in this study. In addition, high-fidelity simulators, such as NuPlan [46] and Waymax [47], can be used to test our methods in closed-loop simulation environments, and could provide important insights into planning methods that need to cope with interactions. While a lot of work needs to be done before we can safely deploy AVs, proper validation of our methods can help demonstrate their capabilities and limitations, and provide new insights for further research.

6.2.2 Real-Time Capability

While the current implementation shows potential to run the proposed algorithms in real-time, further development is required to achieve this real-time capability. Firstly, the computation is strongly dependent on the prediction horizon length, the size of the input and output to the GP, and the amount of training data, which is a fundamental challenge in GP-MPC [8]. Although further research is required to determine how much and what training data is necessary to have good performance in a large variety of scenarios, it can be expected that the size of the training data will exceed that of this study. As the GP has a relatively high computational complexity, using a dedicated toolbox for GPs, such as GPyTorch [48] could accelerate computations. However, the integration of such toolboxes in the optimization problem is considered part of future work.

6.2.3 Problem Formulation

This thesis aimed to provide a better understanding of an online learning Gaussian process-based MPC for motion planning. For the sake of analysis, the study focused on a simplified test case with only two target vehicles, however, this posed some limitations.

Active Learning in Dense Traffic The simulation studies were limited to sparse traffic scenarios to limit the complexity of the problem. Although simplified test cases enhance the interpretability of the method and the results, it also imposes some limitations on the capabilities. The sparse traffic prohibited an explicit incentive for the AV to merge as this would bias the motion plan to merge as quickly as possible. However, increasing the density of the traffic increases the complexity of the decision-making, e.g. between which vehicles to merge, and is a natural next step to further explore the potential of our GP-MPC methods.

In denser traffic, the AV has to negotiate with the other vehicles on a social level to see what vehicle will yield to the AV, as seen in [6, 21]. As such, with denser traffic, the AV can be incentivized to merge into the target lane through the objective function, without introducing significant bias to the solution. Consequently, by only allowing a certain degree of active exploration if this exploration is expected to improve the primary objective, it is expected that the active learning framework from [13] can be leveraged to its full potential. For example, the motion planner could perturb other road users by letting the AV nudge into the target lane in order to see what vehicle will yield to the AV. It has been shown that the active learning framework from [13] can be used to actively explore the state space using our GP-MPC motion planner, however, the simplified test case limits its relevance and performance. In conclusion, it is expected that explicitly incentivizing the AV to merge, through a reformulated objective function, enables the GP-MPC to better exploit the active learning framework from [13] and leverage the framework to its full potential in a dense traffic scenario.

Cooperative Driving Optimizing the flow of the entire traffic is a natural driving objective of many human drivers. However, this requires some understanding of the interactions between vehicles. Interaction-aware prediction models provide a sense of indirect control over other vehicles which can be leveraged to improve the performance of all road users and cooperate with other drivers towards a shared objective. By imposing a social cost, e.g. limiting the decelerations of other vehicles, or maximizing the average velocity of all road users, the motion planner can consider such social factors while controlling the AV.

External Predictors As discussed in Chapter 1, lack of generalizability and interpretability are major challenges for deep-learning-based planners and predictors. However, they can be integrated with control theoretical frameworks, like MPC. For example, deep-learning-based planners can be used to provide a reference trajectory. Subsequently, the MPC could be used to plan the motion of the AV while considering the interactions between the AV and the target vehicles and possibly providing guarantees on the safety of the motion plan.

6.2.4 Safety Guarantees

Recursive Feasibility Providing safety guarantees of a motion planning algorithm requires the feasibility of all safety-related constraints of the MPC problem for all future time steps. Assuming that the MPC problem is feasible at the current time, one needs to guarantee that it remains feasible for all time steps. In general, establishing theoretical properties, such as recursive feasibility and stability of the stochastic optimal control problem poses a major challenge [24]. In the absence of uncertainty, such guarantees can be provided by selecting the appropriate *terminal ingredients* for the MPC, namely an appropriate terminal cost for the objective function as well as terminal constraints [7]. The difficulty in systems subject to uncertainty, like in motion planning, lies in changing operating conditions and a lack of control over another agent. Providing recursive feasibility guarantees for a system subject to uncertainty requires a notion of a safe set. Loosely speaking, in order to guarantee safety we need to determine what another agent might do and account for this. One could give robust safety guarantees, however, this may lead to overconservative behavior.

Stochastically Reachable Sets In the case of stochastic MPC, the recursive feasibility of constraints can be guaranteed, analogous to robust MPC, using the concept of stochastically safe sets. In this study, the chance constraints were approximated. Although one can provide guarantees on constraint satisfaction with a user-defined probability by tightening the chance constraints, assuming one can find the necessary terminal ingredients. However, these methods typically require some knowledge of the stochastic disturbance [49]. Such distributional information can be estimated, but it is difficult to exactly know this distribution, in particular when trying to cope with unseen behavior. Although virtually any controller can be enhanced with safety filters to provide rigorous guarantees, the performance of such systems remains an open question [8].

Contingency MPC As seen in the results of this study, some scenarios require a certain degree of risk to successfully merge in between the two vehicles. Naturally, human drivers accept a certain degree of risk as long as they have a contingency plan that they think is safe. For example, a human driver may attempt to merge in between two vehicles, as long as he can also brake and safely merge behind the two vehicles. Ideally, if the safe contingency plan tends to become infeasible, the human driver aborts its attempt to merge in between and decides to safely merge behind. Much like human drivers, contingency MPC [50] could serve as an effective framework to leverage our GP-MPC, which relies on approximate distributional information to plan a performance trajectory while having a contingency trajectory in parallel that is guaranteed to be safe. While safety guarantees are outside the scope of this thesis, contingency MPC is a viable option for future research to combine the approximate online learning GP-MPC with a guaranteed safe reachable set formulation. Actual driving data is essential to determine the relevance of stochastic distributions before one can make meaningful claims to guarantee safety. Until then, robust reachable sets could be leveraged to guarantee the safety of such a contingency-based MPC. Future research has to determine the extent of the imposed conservatism.

References

[1] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and Decision-Making for Autonomous Vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, May 2018.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A Survey of Autonomous Driving: *Common Practices and Emerging Technologies*," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[3] M. During and P. Pascheka, "Cooperative decentralized decision making for conflict resolution among autonomous agents," in *2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings*, Alberobello, Italy: IEEE, Jun. 2014, pp. 154–161.

[4] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH Journal*, vol. 1, no. 1, p. 1, Dec. 2014.

[5] Y. Chen, S. Veer, P. Karkus, and M. Pavone, "Interactive Joint Planning for Autonomous Vehicles," *IEEE Robotics and Automation Letters*, pp. 1–8, Nov. 2023.

[6] K. Liu, N. Li, H. E. Tseng, I. Kolmanovsky, and A. Girard, "Interaction-Aware Trajectory Prediction and Planning for Autonomous Vehicles in Forced Merge Scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 474–488, Jan. 2023.

[7] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, Dec. 2014.

[8] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-Based Model Predictive Control: Toward Safe Learning in Control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 269–296, May 2020.

[9] T. Brüdigam, A. Capone, S. Hirche, D. Wollherr, and M. Leibold, "Gaussian Process-based Stochastic Model Predictive Control for Overtaking in Autonomous Racing," *arXiv preprint arXiv:2105.12236*, May 2021.

[10] E. L. Zhu, F. L. Busch, J. Johnson, and F. Borrelli, *A Gaussian Process Model for Opponent Prediction in Autonomous Racing*, arXiv:2204.12533 [cs, eess], Mar. 2023.

[11] J. Bethge, M. Pfefferkorn, A. Rose, J. Peters, and R. Findeisen, "Model Predictive Control with Gaussian-Process-Supported Dynamical Constraints for Autonomous Vehicles," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 507–512, 2023.

[12] A. Mesbah, "Stochastic model predictive control with active uncertainty learning: A Survey on dual control," *Annual Reviews in Control*, vol. 45, pp. 107–117, 2018.

[13] R. Soloperto, J. Kohler, and F. Allgöwer, "Augmenting MPC Schemes With Active Learning: Intuitive Tuning and Guaranteed Performance," *IEEE Control Systems Letters*, vol. 4, no. 3, pp. 713–718, Jul. 2020.

[14] T. Beckers and S. Hirche, "Prediction With Approximated Gaussian Process Dynamical Models," *IEEE Transactions on Automatic Control*, vol. 67, no. 12, pp. 6460–6473, Dec. 2022.

[15] A. Mesbah, K. P. Wabersich, A. P. Schoellig, M. N. Zeilinger, S. Lucia, T. A. Badgwell, and J. A. Paulson, "Fusion of Machine Learning and MPC under Uncertainty: What Advances Are on the Horizon?" In *2022 American Control Conference (ACC)*, Atlanta, GA, USA: IEEE, Jun. 2022, pp. 342–357.

[16] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious Model Predictive Control Using Gaussian Process Regression," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, Nov. 2020.

[17] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-Based Model Predictive Control for Autonomous Racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, Oct. 2019.

[18] R. Soloperto, M. A. Muller, and F. Allgöwer, "Guaranteed Closed-Loop Learning in Model Predictive Control," *IEEE Transactions on Automatic Control*, vol. 68, no. 2, pp. 991–1006, Feb. 2023.

[19] M. Buisson-Fenet, F. Solowjow, and S. Trimpe, "Actively Learning Gaussian Process Dynamics," 2020.

[20] B. Evens, M. Schuurmans, and P. Patrinos, "Learning MPC for Interaction-Aware Autonomous Driving: A Game-Theoretic Approach," in *2022 European Control Conference (ECC)*, London, United Kingdom: IEEE, Jul. 2022, pp. 34–39.

[21] J. Knaup, J. D'sa, B. Chalaki, T. Naes, H. N. Mahjoub, E. Moradi-Pari, and P. Tsiotras, *Active Learning with Dual Model Predictive Path-Integral Control for Interaction-Aware Autonomous Highway On-ramp Merging*, arXiv:2310.07840 [cs, math], Oct. 2023.

[22] D. Holley, J. D'sa, H. N. Mahjoub, G. Ali, B. Chalaki, and E. Moradi-Pari, *MR-IDM - Merge Reactive Intelligent Driver Model: Towards Enhancing Laterally Aware Car-following Models*, arXiv:2305.12014 [cs, eess], May 2023.

[23] Y. Liu, P. Wang, and R. Tóth, *Learning For Predictive Control: A Dual Gaussian Process Approach*, arXiv:2211.03699 [cs, eess], Nov. 2022.

[24] A. Mesbah, "Stochastic Model Predictive Control: An Overview and Perspectives for Future Research," *IEEE Control Systems*, vol. 36, no. 6, pp. 30–44, Dec. 2016.

[25] E. Snelson and Z. Ghahramani, "Sparse Gaussian Processes using Pseudo-inputs," *Advances in Neural Information Processing Systems*, vol. 18, 2005.

[26] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, pp. 1805–1824, Aug. 2000.

[27] F. Zong, M. Wang, M. Tang, X. Li, and M. Zeng, "An Improved Intelligent Driver Model Considering the Information of Multiple Front and Rear Vehicles," *IEEE Access*, vol. 9, pp. 66 241–66 252, 2021.

[28] A. Kesting, M. Treiber, and D. Helbing, "Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1928, pp. 4585–4605, Oct. 2010.

[29] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar. 2006.

[30] I. Wolfram Research, *Mathematica*, Champaign, Illinois, 2023.

[31] R. A. Adams and C. Essex, *Calculus: a complete course*, 9th ed. Pearson, 2018.

[32] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen, "A Survey on Trajectory-Prediction Methods for Autonomous Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 652–674, Sep. 2022.

[33] M. Schuurmans, A. Katriniok, C. Meissen, H. E. Tseng, and P. Patrinos, "Safe, learning-based MPC for highway driving under lane-change uncertainty: A distributionally robust approach," *Artificial Intelligence*, vol. 320, p. 103 920, Jul. 2023.

[34] A. Carvalho, Y. Gao, S. Lefevre, and F. Borrelli, "Stochastic Predictive Control of Autonomous Vehicles in Uncertain Environments," in *Proceedings of the 39th International Conference on Machine Learning*, Tokyo, Japan, Sep. 2014, pp. 712–719.

[35] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning* (Adaptive computation and machine learning). Cambridge, MA, USA: MIT Press, 2006.

[36] T. Beckers and S. Hirche, "Stability of Gaussian process state space models," in *2016 European Control Conference (ECC)*, Aalborg, Denmark: IEEE, Jun. 2016, pp. 2275–2281.

[37] C. A. Micchelli, Y. Xu, and H. Zhang, "Universal Kernels," *Journal of Machine Learning Research 7*, Dec. 2006.

[38] J. Umlauft, T. Beckers, and S. Hirche, "Scenario-based Optimal Control for Gaussian Process State Space Models," in *2018 European Control Conference (ECC)*, Limassol, Cyprus: IEEE, Jun. 2018, pp. 1386–1392.

[39] L. Hewing, E. Arcari, L. P. Frohlich, and M. N. Zeilinger, "On Simulation and Trajectory Prediction with Gaussian Process Dynamics," in *Proceedings of Machine Learning Research*, vol. 120:1-11, 2020.

[40] J. Kocijan, *Modelling and Control of Dynamic Systems Using Gaussian Process Models* (Advances in Industrial Control). Cham: Springer International Publishing, 2016.

[41] R. Fletcher, *Practical Methods of Optimization*. Wiley, 2000.

[42] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, Mar. 2019.

[43] HSL, *Coin-HSL*, May 2023.

[44] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, USA: IEEE, Nov. 2018, pp. 2118–2125.

[45] T. Moers, L. Vater, R. Krajewski, J. Bock, A. Zlocki, and L. Eckstein, "The exiD Dataset: A Real-World Trajectory Dataset of Highly Interactive Highway Scenarios in Germany," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, Aachen, Germany: IEEE, Jun. 2022, pp. 958–964.

[46] H. Caesar *et al.*, *NuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles*, arXiv:2106.11810 [cs], Feb. 2022.

[47] C. Gulino *et al.*, *Waymax: An Accelerated, Data-Driven Simulator for Large-Scale Autonomous Driving Research*, arXiv:2310.08710 [cs], Oct. 2023.

[48] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, *GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration*, arXiv:1809.11165 [cs, stat], Jun. 2021.

[49] L. Hewing, K. P. Wabersich, and M. N. Zeilinger, "Recursively feasible stochastic model predictive control using indirect feedback," *Automatica*, vol. 119, p. 109 095, Sep. 2020.

[50] J. P. Alsterda, M. Brown, and J. C. Gerdes, "Contingency Model Predictive Control for Automated Vehicles," in *2019 American Control Conference (ACC)*, Philadelphia, PA, USA: IEEE, Jul. 2019, pp. 717–722.

Appendix A

Collision Avoidance Ellipse

This thesis makes uses of a model predictive control (MPC) for motion planning of autonomous vehicles. This MPC relies on a set of constraints that aim to avoid any collisions of the Ego vehicle with the other target vehicles in the scenario. Collision avoidance is governed by a safety ellipse that constrains the centroid of the target vehicles to be outside an ellipse surrounding the Ego vehicle. Here, we provide the detailed calculations that are used to parameterize this bounding ellipse.

As mentioned in the thesis, this ellipse is defined by the following constraint function:

$$\begin{aligned} h_c\left(\mathbf{x}_i^0, \mathbf{x}_i^j\right) &= -\frac{\left(c_{x,i}^1 - c_{x,i}^0\right)^2}{\mathcal{E}_{c,A}^2} - \frac{\left(c_{y,i}^1 - c_{y,i}^0\right)^2}{\mathcal{E}_{c,B}^2} + 1 \\ h_c\left(\mathbf{x}_i^0, \mathbf{x}_i^j\right) &\leq 0, \quad \text{for } i = 0, \ldots, N, \end{aligned} \tag{A.1}$$

where $c_{x,i}^j$ and $c_{y,i}^j$ denote the longitudinal and lateral component of the geometric center of the j^{th} vehicle at prediction step i, respectively. The major and minor semi-axis of the ellipse are denoted by $\mathcal{E}_{c,A}$ and $\mathcal{E}_{c,B}$, respectively. This ellipse accounts for the size of the Ego vehicle and the target vehicle. Furthermore, the relative heading angle between the vehicles can be accounted for by enlarging the ellipse such that under a maximum allowed heading angle, no collision shall occur provided that our prediction model is correct.

The minor axis of the ellipse is fixed such that the vehicles can safely pass one another without unnecessary interference. The minor axis length is determined by the vehicle width $W = 2.18$, with some additional margin:

$$\mathcal{E}_{c,B} = W + 0.82 = 3 \text{ [m]}. \tag{A.2}$$

Subsequently, we calculate the major axis length of the ellipse using a heuristic. In order to determine the major semi-axis length, we define the most critical location of collision provided the minor axis length of ellipse $\mathcal{E}_{c,B}$. Consequently, we have defined a minor axis length and a point $(X_{\text{crit}}, Y_{\text{crit}})$ that coincides with the ellipse, and we can uniquely determine the major axis length of the ellipse.

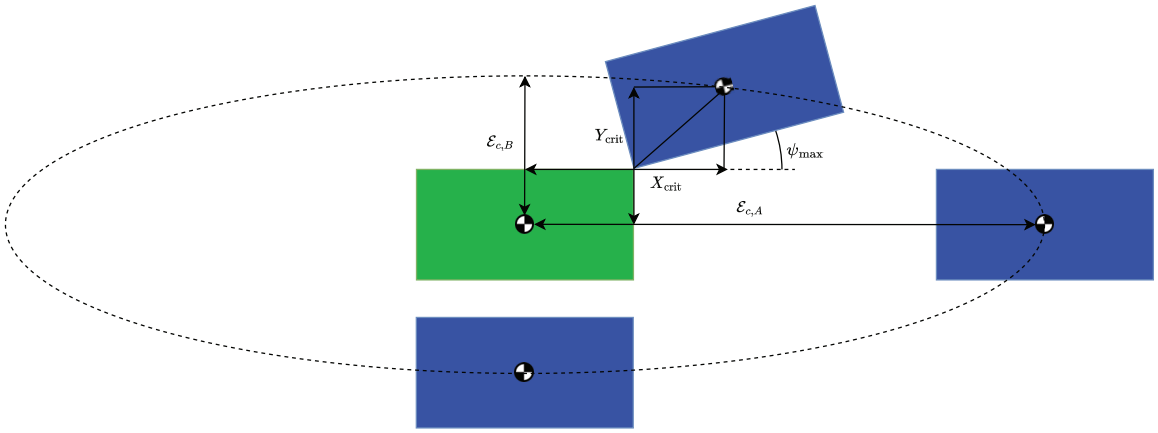

Figure A.1: Parameterization of the collision avoidance ellipse with the Ego vehicle in green, and target vehicles in blue.

We assume that the target vehicles drive in a straight line and, hence, have a heading angle of zero. Furthermore, the Ego vehicle is constrained by the motion planner to attain a maximum heading angle $\mid \psi_{\max} \mid = 0.2618$ [rad.] ≈ 15 [deg.]. The critical location of collision is where the corners of the two rectangles representing the vehicles coincide, as shown in Figure A.1. The longitudinal distance from the Ego's centroid to the critical point X_{crit} is calculated as follows:

$$X_{\text{crit}} = \frac{L}{2} + \sqrt{\left(\frac{L}{2}\right)^2 + \left(\frac{W}{2}\right)^2} \cos\left(\psi_{\max} + \arctan\frac{W}{L}\right) = 4.26 \text{ [m]}. \tag{A.3}$$

Similarly, the lateral distance from the Ego's centroid to the critical point Y_{crit} equals:

$$Y_{\text{crit}} = \frac{W}{2} + \sqrt{\left(\frac{L}{2}\right)^2 + \left(\frac{W}{2}\right)^2} \sin\left(\psi_{\max} + \arctan\frac{W}{L}\right) = 2.74 \text{ [m]}, \tag{A.4}$$

where $L = 4.62$ denotes the vehicle length and $W = 2.18$ denotes the vehicle width. The critical point coincides with an ellipse that satisfies the equation:

$$\left(\frac{X_{\text{crit}}}{\mathcal{E}_{c,A}}\right)^2 + \left(\frac{Y_{\text{crit}}}{\mathcal{E}_{c,B}}\right)^2 = 1. \tag{A.5}$$

Now that we have defined the minor axis length $\mathcal{E}_{c,B}$ and the critical point $(X_{\text{crit}}, Y_{\text{crit}})$, we can determine the minimal major axis length of the ellipse required to enforce collision avoidance:

$$\mathcal{E}_{c,A} = \left(\frac{1 - \left(\frac{Y_{\text{crit}}}{\mathcal{E}_{c,B}}\right)^2}{X_{\text{crit}}}\right)^{-\frac{1}{2}} = 10.47 \text{ [m]}. \tag{A.6}$$

Appendix B

Results

The MSc thesis on Online Learning for Interaction-Aware Motion Planning with Gaussian Process Model Predictive Control is supported by simulation studies performed in MATLAB. In this appendix, the results of these simulations are tabulated in Table B.2 to Table B.12. These tables provide a high-level description of the results with some quantitative key-performance indicators. In addition to the figures in the thesis, the video footage of the simulations can help the reader in the qualitative assessment of the motion planners and can be requested from the author.

The simulation studies are labeled by a six-digit identification number, and can be used to trace the particular simulation that is displayed in the corresponding video. The first digit denotes the learning type. The second digit denotes the prediction model for the Following agent. The third and fourth digit denote the prediction horizon length N. Then, the fifth digit denotes the number of iterations used for pre-training, where 0 denotes no pre-training. Finally, the sixth digit is used to denote different variants of a similar solution to study the effects of other hyper parameters. The legend, seen in Table B.1, details the composition of the identification number. Parameter settings that deviate from their nominal value are provided in Table B.2 to Table B.12.

Table B.1: Legend of Simulation ID Numbers.

Digit No.	Meaning	Value
1	No Learning	0
	Passive Learning	1
	Active Learning	2
2	CV Model	1
	Stochastic Full GP	2
	Stochastic Sparse GP	3
3 4	Horizon Length	N
5	Learning Iteration	-
6	Variant	-

B.1 Results of Constant Velocity MPC

Table B.2: Detailed Results of Constant Velocity MPC.

ID No.	N	Result	$\epsilon_{\max}$ [–]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	t_{avg} [s]	$t_{\max}$ [s]	Note
010601	6	Merged in Between	0.26	115	86	1.5	4.2	4.4	0.015	0.030	-
010801	8	Merged in Between	0.29	115	86	1.5	4.8	4.2	0.015	0.030	-
011001	10	Merged in Between	0.46	115	78	2.6	3.3	3.1	0.015	0.022	-
011201	12	Merged in Between	0.47	115	78	2.4	4.0	3.0	0.016	0.023	-
011401	14	Merged Behind	0.04	115	72	1.6	4.0	5.6	0.031	0.099	-
011601	16	Merged Behind	0	115	75	1.5	3.8	6.7	0.018	0.029	-
011801	18	Merged Behind	0	115	71	1.5	3.3	8.4	0.021	0.041	-
012001	20	Merged Behind	0	115	72	1.5	3.3	9.8	0.022	0.040	-
012201	22	Merged Behind	0	115	72	1.5	2.7	9.4	0.022	0.035	-
012401	24	Merged Behind	0	115	73	1.5	2.3	10.2	0.026	0.061	-

B.2 Results of Passive Learning SPGP-MPC

Table B.3: Detailed Results of Passive Learning SPGP-MPC without Pre-Training.

ID No.	N	Result	$\epsilon_{\max}$ [–]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	t_{avg} [s]	$t_{\max}$ [s]	Note
130601	6	Merged in Between	0.25	115	85	1.5	4.3	4.4	0.041	0.067	-
130801	8	Merged in Between	0.41	115	78	2.7	5.0	3.2	0.067	0.213	-
131001	10	Merged in Between	0.48	115	77	2.4	5.3	2.9	0.105	0.265	-
131201	12	Merged in Between	0.52	115	74	3.1	5.2	2.6	0.152	0.306	-
131401	14	Merged Behind	0	115	71	1.5	4.4	5.7	0.175	0.337	-
131601	16	Merged Behind	0	115	68	2.0	3.9	8.1	0.244	0.455	-
131801	18	Merged Behind	0	115	66	2.4	3.7	10.5	0.368	0.841	-
132001	20	Merged Behind	0	115	65	2.6	3.3	14.6	0.549	1.519	-
132201	22	Merged Behind	0	115	68	2.2	3.2	15.2	0.676	1.702	-
132401	24	Merged Behind	0	115	71	1.7	2.9	15.5	0.853	3.574	-

Table B.4: Detailed Results of Passive Learning SPGP-MPC with Pre-Training.

ID No.	N	Iteration	Result	$\epsilon_{\max}$ [−]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	t_{avg} [s]	$t_{\max}$ [s]	Note
131211	12	1	Merged in Between	0.33	115	86	1.5	5.0	3.7	0.183	0.384	-
131221	12	2	Merged in Between	0.30	115	84	1.5	6.5	3.5	0.217	0.411	Harsh brake Follower
131411	14	1	Merged in Between	0.52	115	72	3.7	6.4	2.6	0.279	0.682	Harsh brake Follower
131421	14	2	Merged in Between	0.34	115	86	1.5	5.4	3.5	0.316	0.671	Moderate brake Follower
131431	14	3	Merged in Between	0.32	115	80	1.6	4.0	3.7	0.373	1.048	-
131611	16	1	Merged Behind	0	115	74	1.5	3.8	6.7	0.372	0.994	-
131621	16	2	Merged Behind	0	115	75	1.5	3.8	6.6	0.410	1.046	-
131622	16	2	Collision with Follower	0.97	115	90	5.0	5.0	0.0	0.400	1.595	Higher risk: $\sigma = 1$
131612	16	1 with $N = 12$	Merged Behind	0.40	115	86	1.5	5.9	3.3	0.361	0.728	Moderate brake Follower

B.3 Results of Active Learning SPGP-MPC

Table B.5: Detailed Results of Active Learning SPGP-MPC without Pre-Training.

ID No.	N	K	Result	$\epsilon_{\max}$ [–]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	t_{avg} [s]	$t_{\max}$ [s]	Note
230601	6	5	Collision with Leader	0.87	128	66	4.8	5.0	0.0	0.177	0.385	-
230801	8	5	Merged in Between	0.15	124	78	3.7	7.1	4.3	0.286	0.695	Harsh brake Follower
231001	10	5	Merged in Between	0.10	121	85	2.2	5.0	4.6	0.411	0.761	-
231201	12	5	Merged in Between	0.11	120	85	2.0	5.0	4.7	0.539	0.913	-
231401	14	5	Merged in Between	0.11	123	84	2.3	5.0	4.5	0.865	1.874	-
231601	16	5	Merged in Between	0.04	126	84	3.8	5.0	4.3	0.928	1.521	-
231801	18	5	Merged in Between	0.06	128	81	5.0	5.0	3.8	1.262	2.269	-
232001	20	5	Merged Behind	0	116	67	3.6	5.0	17.7	1.789	4.316	-
232201	22	5	Merged Behind	0	116	66	3.6	4.5	19.3	2.282	3.946	-
232401	24	5	Merged Behind	0	115	70	2.9	3.8	21.2	4.282	8.014	-

Table B.6: Detailed Results of Pre-trained Active Learning SPGP-MPC for Various Learning Periods.

ID No.	N	K	Iteration	Result	$\epsilon_{\max}$ [−]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	t_{avg} [s]	$t_{\max}$ [s]	Note
231212	12	1	1	Merged Behind	0	115	71	3.4	5.0	5.8	0.568	0.986	Oscillatory Behavior
231213	12	3	1	Merged in Between	0.34	115	84	5.0	6.6	3.8	0.625	1.030	Moderate brake Follower
231211	12	5	1	Merged in Between	0.46	115	78	5.0	6.4	3.4	0.625	1.553	Moderate brake Follower
231214	12	10	1	Merged in Between	0.29	121	82	5.0	5.0	4.2	0.598	0.998	-

Table B.7: Detailed Results of Active Learning SPGP-MPC with Pre-Training.

ID No.	N	K	Iteration	Result	$\epsilon_{\max}$ [−]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	t_{avg} [s]	$t_{\max}$ [s]	Note
231211	12	5	1	Merged in Between	0.46	115	78	5.0	6.4	3.4	0.625	1.553	Moderate brake Follower
231221	12	5	2	Merged in Between	0.44	115	78	2.6	5.0	3.0	0.723	1.370	-
231411	14	5	1	Merged Behind	0	115	70	4.0	4.6	10.7	0.803	1.596	-
231421	14	5	2	Merged in Between	0.35	118	85	5.0	5.3	3.7	1.056	1.791	-
231611	16	5	1	Merged Behind	0	115	65	4.8	4.3	6.4	1.142	2.188	-
231621	16	5	2	Merged Behind	0	115	77	3.2	4.3	8.1	1.326	3.327	-

B.4 Results on Various Initial Conditions

Table B.8: Detailed Results of Alternative Test Case 1.

ID No.	N	K	Algorithm	Result	$\epsilon_{\max}$ [−]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	t_{avg} [s]	$t_{\max}$ [s]	Note
011203	12	-	CV-MPC	Merged Behind	0.04	111	74	1.9	4.5	6.0	0.019	0.041	-
131203	12	-	Passive SPGP-MPC	Merged Behind	0.18	111	75	1.7	4.6	6.0	0.163	0.286	-
231203	12	5	Active SPGP-MPC	Merged in Between	0.00	116	87	1.4	3.7	5.3	0.575	1.106	$\gamma_{\max} = 100$

Table B.9: Detailed Results of Alternative Test Case 2.

ID No.	N	K	Algorithm	Result	$\epsilon_{\max}$ [−]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	t_{avg} [s]	$t_{\max}$ [s]	Note
011204	12	-	CV-MPC	Merged Behind	0.04	115	75	2.3	5.0	5.0	0.015	0.023	-
001204	12	-	Passive SPGP-MPC	Merged in Between	0	115	87	1.8	3.4	5.7	0.137	0.334	-
231204	12	5	Active SPGP-MPC	Merged in Between	0.24	122	83	5.0	5.0	4.2	0.406	1.033	$\gamma_{\max} = 100$

B.5 Results on Altruistic Driving Behavior

Table B.10: Detailed Results of Alternative Test Case 3.

ID No.	N	K	Algorithm	Result	$\epsilon_{\max}$ [–]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	t_{avg} [s]	$t_{\max}$ [s]	Note
011205	12	-	CV-MPC	Merged in Between	0	115	83	2.4	2.6	8.8	0.015	0.023	-
131205	12	-	Passive SPGP-MPC	Merged in Between	0	115	83	2.4	2.2	7.2	0.202	0.533	-
231205	12	5	Active SPGP-MPC	Merged in Between	0.02	123	82	2.4	4.4	5.7	0.601	1.392	$\gamma_{\max} = 100$

Table B.11: Detailed Results of Alternative Test Case 4.

ID No.	N	K	Algorithm	Result	$\epsilon_{\max}$ [–]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	t_{avg} [s]	$t_{\max}$ [s]	Note
011206	12	-	CV-MPC	Merged in Between	0	125	83	0.6	2.4	7.8	0.019	0.134	-
131206	12	-	Passive SPGP-MPC	Merged in Between	0	125	83	0.2	2.4	6.9	0.177	0.293	-
231206	12	5	Active SPGP-MPC	Merged in Between	0.01	132	83	0.8	3.0	5.9	0.611	1.177	$\gamma_{\max} = 100$

B.6 Results on Adjust Slack Penalty Weights

Table B.12: Detailed Results of Primary Test Case with Adjust Slack Penalty Weights.

ID No.	N	K	Algorithm	Result	$\epsilon_{\max}$ [–]	$v_{\max}$ [km/h]	$v_{\min}$ [km/h]	$a_{\max}$ [m/s^2]	$a_{\min}$ [m/s^2]	$s_{\min}$ [m]	t_{avg} [s]	$t_{\max}$ [s]	Note
011207	12	-	CV-MPC	Merged Behind	0	115	79	2.4	2.6	7.1	0.015	0.023	-
131207	12	-	Passive SPGP-MPC	Merged in Between	0	115	83	2.4	2.2	7.2	0.202	0.533	-
231207	12	5	Active SPGP-MPC	Merged in Between	0.02	123	82	2.4	4.4	5.7	0.601	1.392	$\gamma_{\max} = 250$