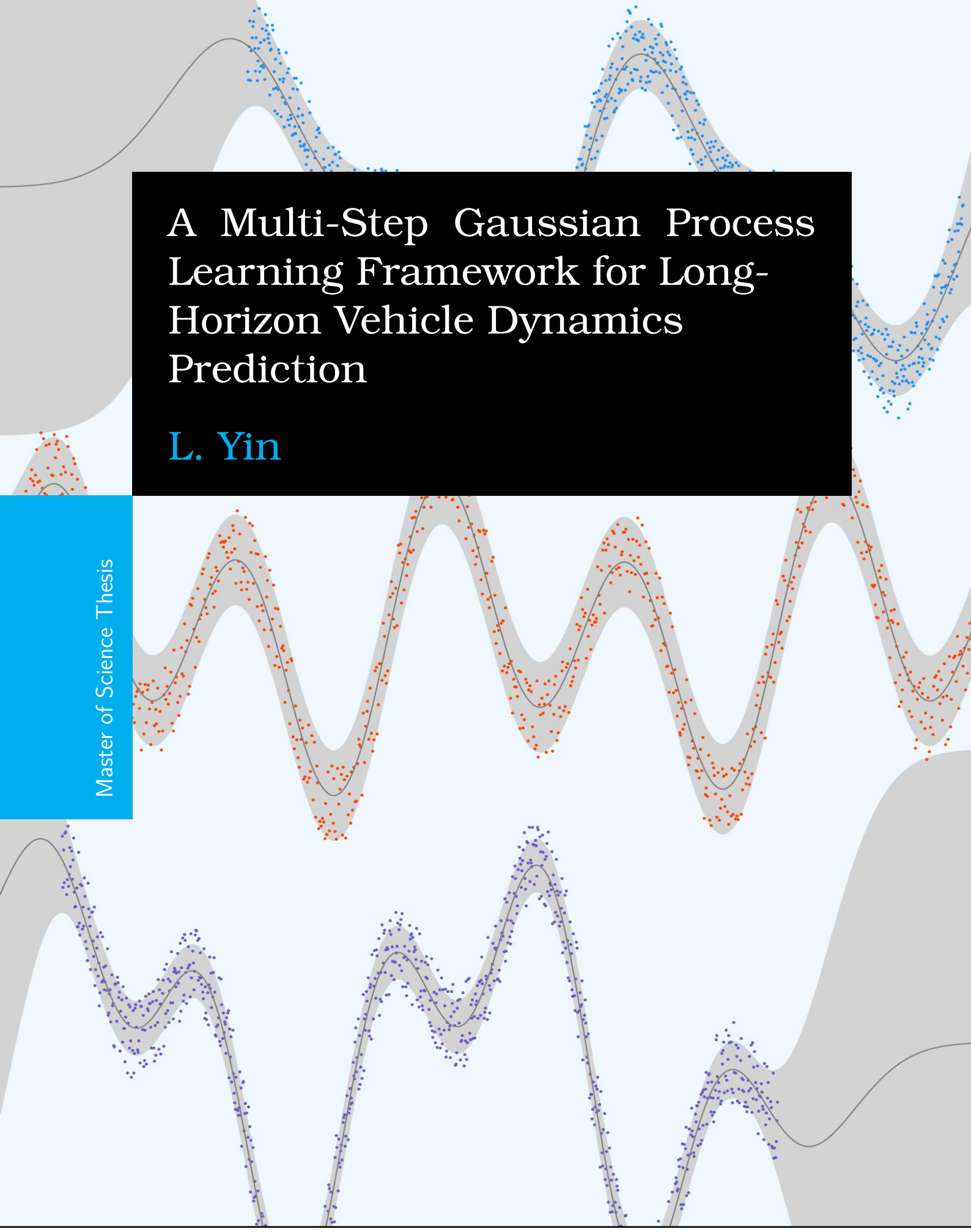


A Multi-Step Gaussian Process Learning Framework for Long-Horizon Vehicle Dynamics Prediction

L. Yin

Master of Science Thesis

TU Delft Delft University of Technology

Department of Cognitive Robotics

A Multi-Step Gaussian Process Learning Framework for Long-Horizon Vehicle Dynamics Prediction

Master of Science Thesis

For the degree of Master of Science in Mechanical Engineering at Delft University of Technology

L. Yin

April 15, 2024

Faculty of Mechanical Engineering (ME) · Delft University of Technology

Cognitive
Robotics

Abstract

This work introduces a novel training strategy for Gaussian Process (GP) models aimed at improving their predictive accuracy and uncertainty quantification capabilities over extended prediction horizons. This improvement is highly relevant for applications in model predictive control (MPC) in the autonomous driving domain. Learning-based MPC strategies typically rely on standard physics-based models augmented with GP models to account for residual nonlinearities and uncertainties not captured by the former. Nonetheless, these conventional approaches often struggle with long-term prediction accuracy, especially when faced with out-of-distribution scenarios, a phenomenon where the model encounters data points that are significantly divergent from the training set. To address these challenges, a multi-step Gaussian process training framework is proposed. This framework yields a GP model capable of making accurate long-term predictions, i.e. a multi-step Gaussian Process (MSGP) model. It achieves this by integrating the simulation of future dynamics into the training process, allowing for the model's kernel parameters to be tuned toward long-term dynamics. As a result, the MSGP model not only demonstrated the ability to make more stable and accurate long-term dynamic predictions but also with greater confidence. The efficacy of the multi-step training framework is shown by the significant improvements in long-horizon dynamics predictions by the MSGP model, achieving an average 19% reduction in mean error and a 90% reduction in variance compared to the standard GP model. Moreover, the efficacy of the MSGP model is further confirmed through its application in a Multi-Step Gaussian Process-based Model Predictive Contouring Controller (MSGP-MPCC), which outperforms a traditional GP-based MPCC (GP-MPCC) baseline controller in lap time and reliability, achieving a 100% success rate in completing laps across ten consecutive simulations without crashing.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Dr. Laura Ferranti for her invaluable guidance and support throughout the process of writing this thesis. Her assistance and critical insights have played an important role in shaping this work. I am also immensely thankful for my daily supervisor Lorenzo Lyons: Thank you for introducing me to the interesting topic of Gaussian Processes and the collaborative brainstorming sessions that have sparked numerous ideas and enriched the depth of this research. I appreciate your critical perspective and for having a listening ear for my questions. I would also like to thank the members of the thesis committee for reviewing this work.

Furthermore, I wish to express my great appreciation to my family and friends. Two years ago, I could not have envisioned reaching this milestone. I am especially grateful to my parents for their unconditional support during my rehabilitation journey, offering reassurance that this day would eventually arrive, albeit not according to my initial plans. I would also like to thank my sister and brother for being the most funny, inspiring, and supportive siblings. Lastly, I extend my gratitude to all my dear friends, both within and beyond Delft, who have made my time here truly unforgettable.

Delft University of Technology L. Yin
April 15, 2024

Chapter 1

Introduction

1-1 Background

The World Health Organization (WHO) reports that annually, road traffic crashes lead to 1.3 million deaths, over half of which involve vulnerable road users (VRUs), and result in disabilities for 20 to 50 million people in non-fatal accidents [16]. Human error was the primary cause of the accident in 93% of the cases [1]. In recent years, advanced driver assistance systems (ADAS) and automated driving (AD) functions have shown an increasing potential to prevent on-road accidents and improve on-road safety [21]. To accelerate this, both academia and industry have been actively involved in the development of autonomous vehicles (AVs) that can navigate in various environmental contexts by taking humans out of the loop altogether [8]. This thesis focuses on the modeling and control domain of autonomous vehicles to tackle the prevalent issues of road safety and accident prevention. By researching and identifying state-of-the-art advancements in these areas, the aim is to bridge existing gaps in the literature.

1-2 Related Literature

1-2-1 Model Predictive Control

Model Predictive Control (MPC) has increasingly gained more popularity as a control technique for autonomous driving because of its ability to handle nonlinear vehicle dynamics while respecting state and actuator constraints [4], [15]. The MPC control task is formulated as an online, open-loop, finite horizon optimization problem, that considers system dynamics and constraints, and is solved in a receding horizon manner [9]. The MPC predicts the dynamics of the system over a prediction horizon N_p, initialized from state measurements obtained at time t. An optimization problem is then solved to find an input sequence that minimizes the predicted cost according to a predefined performance index, which typically involves the error between a system's predicted state or output sequence and a reference sequence [11].

Feedback is incorporated by applying the first control input from the optimized sequence to the system [9]. Subsequently, the time window is shifted by one sample and new plant measurements become available, triggering the repetition of the entire prediction and optimization procedure. The above-described process is repeated continuously, which allows for the MPC to adapt to the changing system dynamics and constraints in real-time, thereby ensuring optimal control performance [17].

In order to make predictions of the system dynamics, a dynamic model of the system is required, which is described by a set of differential equations subjected to a set of input and state constraints [18]

$$\begin{aligned} \dot{\boldsymbol{z}}_t &= \mathbf{f}_c(\boldsymbol{z}_t, \boldsymbol{u}_t), \quad \mathbf{z}(0) = \mathbf{z}_0 \\ \mathbf{z}_t &\in \mathcal{Z} \subseteq \mathbb{R}^n, \ \forall t \geq 0 \\ \mathbf{u}_t &\in \mathcal{U} \subseteq \mathbb{R}^m, \ \forall t \geq 0 \end{aligned} \tag{1-1}$$

in which $\boldsymbol{z}_t$ and $\boldsymbol{u}_t$ are the state and input vectors, constrained by box constraints $\mathcal{Z}$ and $\mathcal{U}$ as follows [9],

$$\begin{aligned} \mathcal{Z} &:= \{\boldsymbol{z} \in \mathbb{R}^n \mid \boldsymbol{z}_{\min} \leq \boldsymbol{z} \leq \boldsymbol{z}_{\max}\} \\ \mathcal{U} &:= \{\boldsymbol{u} \in \mathbb{R}^m \mid \boldsymbol{u}_{\min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{\max}\} \ . \end{aligned} \tag{1-2}$$

Discretizing the continuous system dynamics with sampling time T_s yields the following discrete description of the system dynamics [18],

$$\boldsymbol{z}_{k+1} = \mathbf{f}_d(\boldsymbol{z}_k, \boldsymbol{u}_k) \ . \tag{1-3}$$

Now, the MPC optimization problem can be formulated as

$$\begin{aligned} \min_{\mathbf{u}} \quad & \textstyle\sum_{k=0}^{N_p-1} J(\mathbf{z}_{t+k}, \mathbf{u}_{t+k}) \\ \text{s.t.} \quad & \mathbf{z}_t = \mathbf{z}_{init} \\ & \mathbf{z}_{t+k+1} = \mathbf{f}_d\left(\mathbf{z}_{t+k}, \mathbf{u}_{t+k}\right) \quad \forall k = [0, N_p - 1] \\ & \mathbf{z}_{t+k} \in \mathcal{Z}_{free} \quad \forall k = [0, N_p - 1] \\ & \mathbf{u}_{t+k} \in \mathcal{U} \quad \forall k = [0, N_p - 1] \ . \end{aligned} \tag{1-4}$$

As mentioned previously, this optimization problem is solved in a receding horizon fashion. A measurement of the system's state $\mathbf{z}$ is obtained at time t, based on which a finite-horizon prediction of the system's future behavior and an optimal control input sequence $[\boldsymbol{u}]_{k=t}^{k=t+N_p-1} = \{\boldsymbol{u}_t, \ldots, \boldsymbol{u}_{t+N_p-1}\}$ are computed. Feedback is incorporated by applying the first control input $\boldsymbol{u}_t$ from the optimized sequence to the system [9]. Finally, when a new state measurement becomes available, the whole process repeats.

1-2-2 Gaussian Process-based Prediction Model

To maximize the predictive power of the MPC, its underlying prediction model must accurately represent the dynamics, whilst being sufficiently simple for real-time optimization. In the context of the research area of autonomous vehicles, traditional efforts focused on developing models tailored to various operating ranges such as the kinematic or the (non)linear dynamic bicycle model, of which the latter is suitable for complex maneuvers by considering tire-road interactions. However, these physics-based approaches are costly and time-consuming

to derive as they require expert knowledge [15]. Furthermore, they lack the flexibility to accurately enclose the full range of operating conditions encountered by vehicles, including variations in dynamics due to factors such as weather or road conditions [12], [15].

In recent years, approaches from machine learning have gained traction as a means to address the limitations of classical physics-based models and develop nonlinear models from real-world data. Gaussian Processes (GPs), in particular, have gained popularity in modeling vehicle dynamics for their ability to learn complex nonlinear functions governing the dynamics while providing uncertainty estimates alongside these functions, even in data-scarce environments [13]. Section II-B of the paper enclosed in Chapter 2 provides a comprehensive mathematical outline of Gaussian Processes. This introductory section will provide a more concise overview, focusing on conveying the main concept without delving into extensive detail yet.

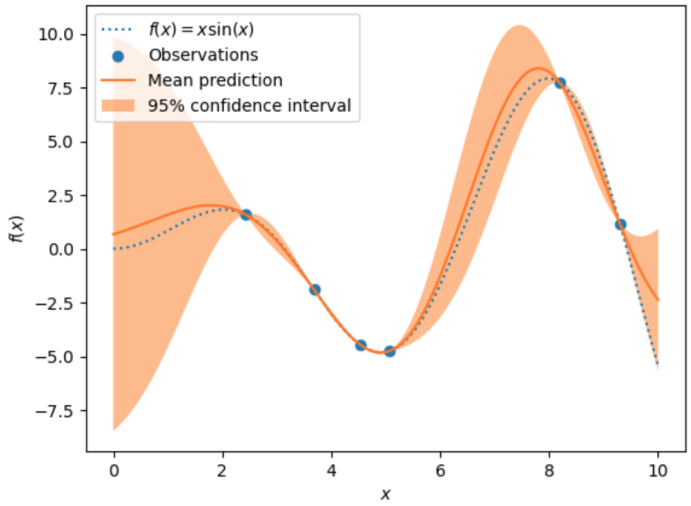


Figure 1-1: A Gaussian Process model fitted to observations marked by blue dots. The red line represents the predicted mean function, which aims to approximate the underlying real function suggested by these blue dots. The shaded area around the mean function illustrates the uncertainty estimate provided by the GP. Notably, the uncertainty increases in regions lacking data and decreases in areas densely populated with data points. [20].

A Gaussian Process (GP) is characterized by a mean function and a covariance function,

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)) \ , \tag{1-5}$$

in which $m(\boldsymbol{x})$ denotes the mean function and $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ the kernel function evaluated for each data point pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ [19], [10]. A GP extends the concept of the Multivariate Normal Distribution (MVN) over functions instead of fixed vectors, making a MVN a specific instance of a GP [15]. A sample $\boldsymbol{f}$ of such a MVN is given by a mean vector $\boldsymbol{m}$ and covariance matrix $\boldsymbol{\Sigma}$, [10]

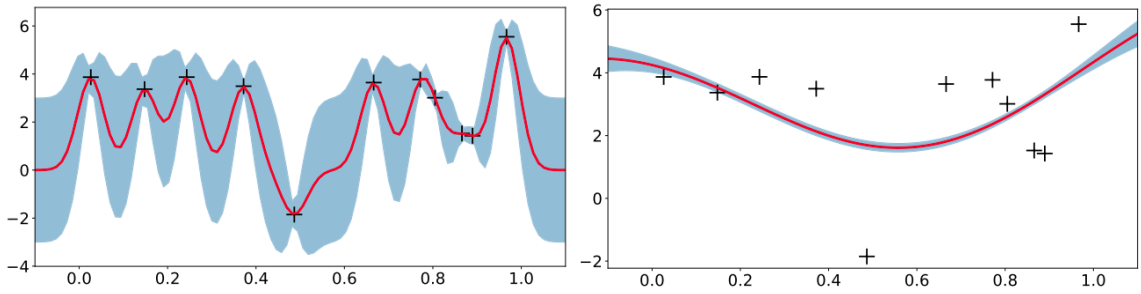
$$\boldsymbol{f} = \mathcal{N}(\boldsymbol{m}, \boldsymbol{\Sigma}) \ . \tag{1-6}$$

To work efficiently with GPs within the constraints of finite computational time, a GP is converted into a finite-dimensional MVN distribution. In this process, the kernel function is crucial as it constructs the covariance matrix, addressing the discrete characteristic of the data that compose the MVN [10].

The key component influencing the behavior of a GP model is its kernel function, which defines how input data points relate to each other. A widely used kernel to characterize the covariance is the *squared exponential* kernel, which ensures strong correlation and similarity for nearby inputs and weakens for distant ones [19],

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{n_d}\frac{(x_{i,d} - x_{j,d})^2}{\ell_d^2}\right). \tag{1-7}$$

Here, $x_{i,d} - x_{j,d}$ is the distance between the d-th feature of an input data pair and ℓ_d is the lengthscale for this feature. The kernel's hyperparameters, such as the signal variance σ_f^2 and the lengthscale ℓ_d, govern the behavior of the GP; a larger lengthscale, for example, indicates a smoother function by increasing correlation across distant inputs, as shown in Fig 1-2.



(a) small lengthscale ℓ **(b)** large lengthscale ℓ

Figure 1-2: GP model's smoothness affected by the kernel lengthscale parameter ℓ [22]

Training a GP model focuses on optimizing these hyperparameters to enhance the model's ability to generalize from training data to unseen scenarios [3], [7]. Once the model has been trained and the GP has learned its dynamics of the system, the model can then be used to make predictions at new data points.

1-3 Challenges in Gaussian Process-based MPC

In learning-based MPC for autonomous vehicles, GPs have frequently served to refine physics-based nominal models by capturing the residual errors between these models and the actual system dynamics [12], [13], [2], [14]. However, challenges emerge when GP models are extrapolated at points beyond their training data, leading to inaccuracies in the prediction of the dynamics, particularly in multi-step prediction scenarios. Findings from [2] have highlighted instances where the MPC's open-loop predictions, based on GP augmented models, deviate more rapidly from the actual trajectories compared to those based solely on the nominal model. This discrepancy arises when GP models encounter an "out-of-distribution" data point [15], which is a point in the state/action space that is not sufficiently correlated with the training data, resulting in erroneous predictions of the dynamics and a large uncertainty as a by-product.

To our knowledge, the literature has not explicitly addressed out-of-distribution behavior in the context of GP-based control. Nonetheless, there exists research focused on managing the substantial uncertainties that partly result from unstable predicted dynamics. These studies,

treat these uncertainties as an isolated problem and propose distinct methods to tackle them in the control phase. The methods often merge the uncertainty stemming from GP evaluations with the uncertainty arising from state evolution. The latter is usually obtained by employing a first-order Taylor approximation to successively linearize around the predicted mean state during each phase of the MPC's open-loop prediction [13], [12]. To address the large uncertainties, the authors of [13] hypothesized integrating a linear ancillary state feedback controller within the MPC's open-loop prediction horizon. However, they acknowledge the challenges this method might face with highly nonlinear systems, and they do not further address the computational demands of incorporating an additional controller into the optimization process. In a different study [12] from the same authors, a renewed framework was proposed for active GP model learning during the operation of an autonomous race car, incorporating a data management system for online learning of GP error models. To work around computational restrictions, the variance dynamics were precomputed based on the solution trajectory from the MPC's previous time steps and kept constant for five control loops, assuming the solution trajectories do not differ too much. To further prevent computational overload and feasibility issues, the uncertainty propagation was limited to only five steps and kept constant for the remainder of the horizon. The existing literature primarily addresses uncertainty management as an isolated problem within the control phase, assuming that the uncertainty as determined by evaluating the GP at the specific state/control input point is directly suitable for use. However, this approach may not be viable when the GP model operates outside its distribution range, as the resulting uncertainties can become excessively large and difficult to handle effectively.

This thesis proposes a shift toward a modeling-centric approach that specifically addresses the inherent instability in long-term predictions made by GPs. The focus on these improvements during the modeling phase is motivated by the opportunity it presents to decrease the computational burdens associated with managing uncertainty during MPC operation. This approach would lead to enhanced resource allocation for optimization and GP evaluation. Recognizing that instability typically occurs when the GP deviates from regions well-represented by the training data, this thesis focuses on developing a strategy to prevent the GP from venturing into poorly correlated areas during its long-term dynamic predictions, thus mitigating "out-of-distribution" scenarios. By aiming to reduce the discrepancy between the predicted dynamics and the actual dynamics, the uncertainty associated with the GP's predictions will also be reduced. The goal is to yield a GP with enhanced predictive capability for long-term dynamics, tailored for better integration with MPC's predictive nature.

1-4 Thesis Objectives and Contributions

To address the challenges mentioned in the previous section, this thesis seeks to develop a method for learning GP models with improved long-term predictive capabilities. The research question is formulated as follows:

How can the mean state and variance prediction of the long-term dynamics be stabilized to maximize Gaussian Process-based MPC performance?

The hypothesis is that refining the parameters of the GP kernel function could improve the model's capacity to capture extended dynamics and provide accurate predictions of associated uncertainties.

The main contribution of this thesis is a multi-step training framework for learning Gaussian Process models capable of making accurate and stable long-term predictions of the dynamics while reducing uncertainties. The enhanced model's superior ability to accurately describe long-term dynamics has been demonstrated through a Model Predictive Contouring Control (MPCC) example and compared to the standard GP-MPCC-based controller in a racetrack navigation context for a small car-like robot.

1-5 Thesis Outline

The thesis is outlined as follows:

- **Introduction**: This section summarizes the advancements in GP applications for predictive control of autonomous ground vehicles as found in the literature and identifies the research gaps.
- **Scientific Paper**: The main body of the thesis, formatted as a paper that presents the proposed multi-step Gaussian Process training framework, is presented in Chapter 2.
- **Supporting Information Methods**: Chapter 3 provides a detailed exposition of the Gaussian process training framework developed, accompanied by an analysis delineating the precise effects resulting from its application.
- **Supporting Information Results: Modeling**: Chapter 4 provides additional information and supporting data on the obtained modeling results.
- **Supporting Information on Results: Controller**: Chapter 5 provides additional information and supporting data on the outcomes obtained from the controller when using GP models in MPCC.
- **Discussion**: Chapter 6 critically examines the results, explains limitations encountered, and proposes potential directions for future research.

Chapter 2

Scientific Paper

The thesis is formatted in paper form, with the following document serving as its primary component.

A Multi-Step Gaussian Process Learning Framework for Long-Horizon Vehicle Dynamics Prediction

Lanke Yin
Department of Cognitive Robotics
Faculty of Mechanical Engineering · Delft University of Technology
Delft, The Netherlands

***Abstract*—This work introduces a novel training strategy for Gaussian Process (GP) models aimed at improving their predictive accuracy and uncertainty quantification capabilities over extended prediction horizons. This improvement is highly relevant for applications in model predictive control (MPC) in the autonomous driving domain. Learning-based MPC strategies typically rely on standard physics-based models augmented with GP models to account for residual nonlinearities and uncertainties not captured by the former. Nonetheless, these conventional approaches often struggle with long-term prediction accuracy, especially when faced with out-of-distribution scenarios, a phenomenon where the model encounters data points that are significantly divergent from the training set. To address these challenges, a multi-step Gaussian process training framework is proposed. This framework yields a GP model capable of making accurate long-term predictions, i.e. a multi-step Gaussian Process (MSGP) model. It achieves this by integrating the simulation of future dynamics into the training process, allowing for the model's kernel parameters to be tuned toward long-term dynamics. As a result, the MSGP model not only demonstrated the ability to make more stable and accurate long-term dynamic predictions but also with greater confidence. The efficacy of the multi-step training framework is shown by the significant improvements in long-horizon dynamics predictions by the MSGP model, achieving an average 19% reduction in mean error and a 90% reduction in variance compared to the standard GP model. Moreover, the efficacy of the MSGP model is further confirmed through its application in a Multi-Step Gaussian Process-based Model Predictive Contouring Controller (MSGP-MPCC), which outperforms a traditional GP-based MPCC (GP-MPCC) baseline controller in lap time and reliability, achieving a 100% success rate in completing laps across ten consecutive simulations without crashing.**

***Index Terms*—Gaussian process (GP), long-horizon prediction, data-driven optimal control, model predictive control (MPC), autonomous driving.**

I. Introduction

Studies from the World Health Organization (WHO) reported that each year 1.3 million lives are lost in road traffic crashes, with human error being the leading cause in 93% of these incidents [1]. As a response to this, in recent years, there has been a surge in interest in autonomous vehicles, with academia and industry actively developing automated driving (AD) functionalities to improve road safety and remove the need for human control [2], [3].

Model predictive control (MPC) has increasingly gained more popularity as a control technique for autonomous driving because of its ability to handle nonlinear vehicle dynamics while respecting state and actuator constraints [4], [5]. The MPC control task is formulated as an online, open-loop, finite horizon optimization problem, that considers system dynamics and constraints, and is solved in a receding horizon manner [6]. The MPC predicts the system's dynamic behavior over a fixed horizon based on current states and solves an optimization problem according to a certain cost. Feedback is incorporated by applying the first control input from the optimized sequence to the system [6]. Finally, when a new state measurement becomes available, the whole process repeats. To maximize the predictive power of the MPC, its underlying prediction model must accurately represent the vehicle dynamics, whilst being sufficiently simple for real-time optimization. Traditional efforts focused on developing models tailored to various operating ranges such as the simple kinematic or the nonlinear dynamic bicycle model, of which the latter is suitable for complex maneuvers by considering tire-road interactions. However, these physics-based approaches are costly and time-consuming to derive as they require expert knowledge [5]. Furthermore, they lack the flexibility

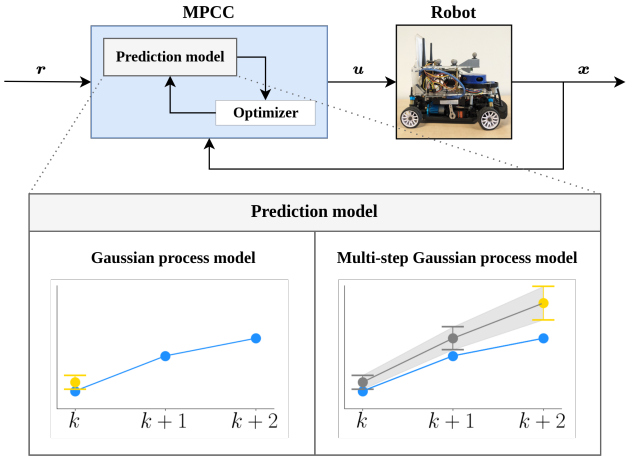


Fig. 1: Overview of the control architecture with the various possible prediction models: a Gaussian process (GP) model or the multi-step Gaussian process (MSGP) model trained using the proposed serially cascaded Gaussian process training framework.

to accurately enclose the full range of operating conditions encountered by vehicles, including variations in dynamics due to factors such as weather or road conditions [5], [8]. In recent years, approaches from machine learning have gained traction as a means to address the limitations of classical physics-based models and develop nonlinear models from real-world data. Gaussian process regression (GPR), in particular, has gained popularity in modeling vehicle dynamics for its ability to learn complex nonlinear functions governing the dynamics while providing uncertainty estimates alongside these functions, even in data-scarce environments [10]. A Gaussian process (GP) is characterized by a mean function and a covariance, of which the latter indicates the uncertainty of the mean function prediction [9]. Subsequently, the learned model can be used to make predictions at new unseen data points.

A. Related Works

In learning-based MPC for autonomous vehicles, GPs have frequently served to refine physics-based nominal models by capturing the residual errors between these models and the actual system dynamics [8], [10], [11], [12]. However, challenges emerge when GP models are extrapolated at points beyond their training data, leading to inaccuracies in the prediction of the dynamics, particularly in multi-step prediction scenarios. Findings from [11] have highlighted instances where the MPC's open-loop predictions, based on GP augmented models, deviate more rapidly from the actual trajectories compared to those based solely on the nominal model. This discrepancy arises when GP models encounter an "out-of-distribution" data point [5], which is a point in the state/action space that is not sufficiently correlated with the training data, resulting in erroneous predictions of the dynamics and a large uncertainty as a by-product.

To our knowledge, the literature has not explicitly addressed out-of-distribution behavior in the context of GP-based control. Nonetheless, there exists research focused on managing the substantial uncertainties that partly result from unstable predicted dynamics. These studies, treat those uncertainties as an isolated problem and propose distinct methods to tackle them in the control phase. The methods often merge the uncertainty stemming from GP evaluations with the uncertainty arising from state evolution. The latter is usually obtained by employing a first-order Taylor approximation to successively linearize around the predicted mean state during each phase of the MPC's open-loop prediction [10], [8]. To address the large uncertainties, the authors of [10] hypothesized integrating a linear ancillary state feedback controller within the MPC's open-loop prediction horizon. However, they acknowledge the challenges this method might face with highly nonlinear systems, and they do not further address the computational demands of incorporating an additional controller into the optimization process. In a different study [8] from the same authors, a renewed framework was proposed for active GP model learning during the operation of an autonomous race car, incorporating a data management system for online learning of GP error models. To work around computational restrictions, the variance dynamics were precomputed based on the solution trajectory from the MPC's previous time steps and kept constant for five control loops, assuming the solution trajectories in consecutive control loops do not differ too much. To further prevent computational overload and feasibility issues, the uncertainty propagation was limited to only five steps and kept constant for the remainder of the horizon. The existing literature primarily addresses uncertainty management as an isolated problem within the control phase, assuming that the uncertainty as determined by evaluating the GP at the specific state/control input point is directly suitable for use. However, this approach may not be viable when the GP model operates outside its distribution range, as the resulting uncertainties can become excessively large and difficult to handle effectively.

Our work proposes a shift toward a modeling-centric approach that specifically addresses the inherent instability in long-term predictions made by GPs. The focus on these improvements during the modeling phase is motivated by the opportunity it presents to decrease the computational burdens associated with managing uncertainty during MPC operation. This approach would lead to enhanced resource allocation for optimization and GP evaluation. Recognizing that instability typically occurs when the GP deviates from regions well-represented by the training data, we focus on developing a strategy to prevent the GP from venturing into poorly correlated areas during its long-term dynamic predictions, thus mitigating "out-of-distribution" scenarios. By aiming to reduce the discrepancy between the predicted dynamics and the actual dynamics, the uncertainty associated with the GP's predictions will also be reduced. The goal is to yield a GP with enhanced predictive capability for long-term dynamics, tailored for better integration with MPC's predictive nature.

In this paper, we present a novel training framework that encourages Gaussian Process models to learn how to make accurate and stable long-term predictions of the dynamics with calibrated uncertainties. The framework was evaluated using a training and validation dataset and showcased in an application where Gaussian processes model the full vehicle dynamics of a 1:10 scale car-like robot. The new model's superior ability to accurately describe long-term predictions has been demonstrated on an MPCC example and compared to the standard GP-MPCC-based controller.

II. Preliminaries

A. Notation

- A vector is denoted with a bold lower case letter: $\boldsymbol{x} \in \mathbb{R}^n$
- A matrix is denoted with a bold upper case letter: $\boldsymbol{X} \in \mathbb{R}^{n \times m}$
- A discrete time step is denoted with Latin letter: k
- A kernel function is denoted with Greek letter: κ
- A multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$: $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- Predicted quantities at time step k are denoted with a subscript: e.g. $\boldsymbol{x}_k$
- Predicted quantities by a GP model during training are denoted with a circumflex: $\hat{\boldsymbol{v}}_k$

- Covariance matrices during training are denoted as $\boldsymbol{K}$.
- The posterior covariance is denoted as $\boldsymbol{\Sigma}$.
- A matrix $\boldsymbol{K}$ that is updated with matrix $\tilde{\boldsymbol{K}}$: $\boldsymbol{K} \leftarrow \tilde{\boldsymbol{K}}$
- The sum of quantities over a prediction horizon is indicated with a non-bold Σ.
- Mean quantities predicted quantities by a GP are denoted with a bar: $\overline{\boldsymbol{a}}$
- The j-th element of a vector: $[\boldsymbol{v}]_j$
- The j-th column of matrix: $\boldsymbol{X}$ is $[\boldsymbol{X}_k]_{\cdot,j}$
- A vector without its last element: $[\boldsymbol{v}]_{\setminus -1}$
- A matrix without the j-th row and column: $[\boldsymbol{X}]_{\setminus j,\setminus j}$.

B. Gaussian Process Regression

A Gaussian Process (GP) is an infinite collection of random variables with a Gaussian joint distribution [9], extending the multivariate Gaussian distribution to infinite dimensions to facilitate modeling distributions over functions rather than vectors [5]. A GP is dictated by a mean function $m(\boldsymbol{x})$ and a kernel function $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that describes the covariance between all possible input data pairs $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ [7]. With these two components, a GP can describe a nonlinear real-world process $f(\boldsymbol{x})$ as

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)) \ . \tag{1}$$

Consider a collection of n input and output data pairs $(\boldsymbol{x}_i, y_i)$ forming the data set $\mathcal{D}$,

$$\begin{aligned} \mathcal{D} = \big\{ \ & \mathbf{y} = [y_0; \ldots; y_{n-1}] \in \mathbb{R}^n, \\ & \boldsymbol{X} = \left[\boldsymbol{x}_0^T; \ldots; \boldsymbol{x}_{n-1}^T\right] \in \mathbb{R}^{n \times n_d} \big\} \end{aligned} \tag{2}$$

that represents a yet to be identified nonlinear function $f : \boldsymbol{x} \to y$ with input $\boldsymbol{x} \in \mathbb{R}^{n_d}$ (n_d number of input features) and target values $y \in \mathbb{R}$,

$$y = f(\boldsymbol{x}) + \varepsilon \ , \tag{3}$$

in which $f \colon \mathbb{R}^{n_x} \to \mathbb{R}$ is the underlying nonlinear function to be modeled by the GP and $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$ is an independent identically distributed (i.i.d.) Gaussian measurement noise with zero mean and variance σ_n^2 on the observed data (likelihood noise).

Specifying a GP prior with a prior mean and kernel function results in a normal distribution of the observed data

$$\mathbf{y} = \mathcal{N}(m(\boldsymbol{X}), \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I}) \ , \tag{4}$$

in which $\boldsymbol{X}$ and $\mathbf{y}$ are the observed input-output data points, $\boldsymbol{\mu} = [m(\boldsymbol{x}_0), \ldots, m(\boldsymbol{x}_{n-1})]$ is the mean function, and $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$ is the kernel matrix with each element obtained by evaluating the kernel function for each possible data pair $[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})]_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The behavior of the GP is primarily governed by the selection of the kernel function and its parameters, which is typically determined by optimizing the likelihood of the observed data [10]. Further details on this will be elaborated in section II-C. To perform regressions using a GP model, i.e. make predictions $\mathbf{f}_*$ at a number of n_* test points (unseen data points) $\boldsymbol{X}_* = \left[\boldsymbol{x}_{0*}^T; \ldots; \boldsymbol{x}_{n-1_*}^T\right] \in \mathbb{R}^{n_* \times n_d}$ given the target values, we make use of the fact that the joint distribution of the target observations $\mathbf{y}$ and latent function values $\mathbf{f}_*$, $p(\mathbf{y}, \mathbf{f}_* | \boldsymbol{X}, \boldsymbol{X}_*)$, is Gaussian [9], [16]:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\boldsymbol{X}) \\ m(\boldsymbol{X}_*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I} & \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*) \\ \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) & \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) \end{bmatrix} \right). \tag{5}$$

$\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) \in \mathbb{R}^{n \times n}$ is the kernel matrix evaluated at all pairs of training points according to the kernel function in , i.e. $[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})]_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Similarly, $\boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) \in \mathbb{R}^{n_* \times n_*}$ between all test point pairs, $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*) \in \mathbb{R}^{n \times n_*}$ between all training and test point pairs, and $\boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) = \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*)^T$. $\left[m(\boldsymbol{X}), m(\boldsymbol{X_*})\right] = \mathbf{0}$ as no prior information is available about the system or the data has been demeaned. To obtain the posterior distribution of the latent function values $\mathbf{f}_*$ that only contains functions from the prior distribution consistent with the training data, it is conditioned on the observations $\mathbf{y}$, training inputs $\boldsymbol{X}$, and test points $\boldsymbol{X}_*$, yielding the predictive equations for GP regression,

$$\begin{aligned} & p\big(\mathbf{f}_* \mid \boldsymbol{X}, \mathbf{y}, \boldsymbol{X}_*\big) \sim \mathcal{N}\left(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*\right), \text{ in which} \\ & \boldsymbol{\mu}_* = \boldsymbol{K}\left(\boldsymbol{X}_*, \boldsymbol{X}\right) \left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I}\right]^{-1} \mathbf{y} \\ & \boldsymbol{\Sigma}_* = \boldsymbol{K}\left(\boldsymbol{X}_*, \boldsymbol{X}_*\right) \\ & \qquad - \boldsymbol{K}\left(\boldsymbol{X}_*, \boldsymbol{X}\right) \left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I}\right]^{-1} \boldsymbol{K}\left(\boldsymbol{X}, \boldsymbol{X}_*\right) \ , \end{aligned} \tag{6}$$

in which $\boldsymbol{\mu}_*$ is the posterior mean and $\boldsymbol{\Sigma}_*$ the posterior covariance [9]. The derivation of the conditional from the joint distribution is done according to Theorem I in Appendix A. In case there is only one test point $\boldsymbol{x}_*$, $\boldsymbol{X}_*$ is interchanged with a single vector $\boldsymbol{x}_*$ to obtain a univariate Gaussian with mean μ_* and variance $\mathbb{V}_*$ [9],

$$\begin{aligned} & p\big(f_* \mid \boldsymbol{X}, \mathbf{y}, \boldsymbol{x}_*\big) \sim \mathcal{N}\left(\mu_*, \mathbb{V}_*\right), \text{ in which} \\ & \mu_* = \boldsymbol{k}\left(\boldsymbol{x}_*, \boldsymbol{X}\right) \left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I}\right]^{-1} \mathbf{y} \\ & \mathbb{V}_* = \kappa\left(\boldsymbol{x}_*, \boldsymbol{x}_*\right) \\ & \qquad - \boldsymbol{k}\left(\boldsymbol{x}_*, \boldsymbol{X}\right) \left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I}\right]^{-1} \boldsymbol{k}\left(\boldsymbol{X}, \boldsymbol{x}_*\right) \ . \end{aligned} \tag{7}$$

$\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) \in \mathbb{R}^{n \times n}$, $\kappa(\boldsymbol{x}_*, \boldsymbol{x}_*) \in \mathbb{R}$, $\boldsymbol{k}(\boldsymbol{x}_*, \boldsymbol{X}) \in \mathbb{R}^n$, and $\boldsymbol{k}(\boldsymbol{X}, \boldsymbol{x}_*) = (\boldsymbol{k}(\boldsymbol{x}_*, \boldsymbol{X}))^T \in \mathbb{R}^n$, in which all elements are evaluated using the squared exponential kernel function in (8). To provide some interpretation into how the posterior variance is composed: the first term is the variance at the test point, from which a (positive) term is subtracted, which tells us how much the training data $\boldsymbol{X}$ has explained [24]. This enables comprehension of the concept of the GP model going "*out-of-distribution*", signifying that the test points do not share a sufficient correlation with the training data set, i.e. $\boldsymbol{k}(\boldsymbol{x}_*, \boldsymbol{X}) \approx 0$ and $\boldsymbol{k}(\boldsymbol{X}, \boldsymbol{x}_*) = \boldsymbol{k}(\boldsymbol{x}_*, \boldsymbol{X})^T \approx 0$. As a consequence, the predictive mean approaches zero and the uncertainty fails to be revised based on insights from the training data, therefore remaining large. Furthermore, it can be seen that the variance only depends on the inputs $\boldsymbol{X}$ and $\boldsymbol{X}_*$ (via the kernel function) and not on the target values $\mathbf{y}$, which is a property of the Gaussian distribution [16].

C. GP Model Training

As mentioned in section II-B, the behavior of a GP model is dictated by a kernel function that describes the relationship between input data points. A commonly employed kernel function to characterize the covariance is the *squared exponential* kernel function. This function indicates strong correlation and similarity for nearby inputs and weakens for distant ones [9]:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{n_d}\frac{(x_{i,d}-x_{j,d})^2}{\ell_d^2}\right) \tag{8}$$

$x_{i,d} - x_{j,d}$ is the distance between the d-th feature of an input data pair and ℓ_d is the lengthscale regarding this input feature. The kernel hyperparameters that govern the GP's behavior are the signal variance σ_f^2, which determines the vertical span of the function and the lengthscale ℓ_d which regulates the correlation pace between input points: the larger the lengthscale, the higher the correlation between distant input data points (smoother functions) [16]. The core of GP model training lies in identifying the best hyperparameters, which is crucial for the model's generalization capability [14] [15]. This training process involves iterative adjustment of the kernel hyperparameters to ensure that the GP model accurately represents the underlying real-world system. Mathematically, this task involves maximizing the log marginal likelihood (or evidence), which is the probability of observing the target values $\mathbf{y}$, given the input data points $\boldsymbol{X}$, and optimization parameters $\boldsymbol{\theta} = [\sigma_n, \sigma_f, \boldsymbol{\ell}^T]$ [9]:

$$\begin{aligned}\log p(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{X}) = &-\frac{1}{2}\mathbf{y}^T\left[\boldsymbol{K}_\theta + \sigma_n^2\boldsymbol{I}\right]^{-1}\mathbf{y} \\ &-\frac{1}{2}\log|\boldsymbol{K}_\theta + \sigma_n^2\boldsymbol{I}| - \frac{n}{2}\log 2\pi\ .\end{aligned} \tag{9}$$

As can be seen, the optimization of the kernel parameters happens through the matrix $\boldsymbol{K}_\theta = \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$, whose elements are all obtained using the kernel function (8).

To solve the optimization problem, the maximization of the log marginal likelihood is recast as a non-convex and nonlinear minimization of the negative log marginal likelihood,

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \ -\log p(\mathbf{y}|\boldsymbol{\theta}, \boldsymbol{X})\ , \tag{10}$$

which can be solved using gradient descent algorithms. This optimization is repeated for several epochs (training cycles), over which the GP model gradually learns to recognize the inherent patterns and relationships within the training data until the model has achieved the desired model fit.

D. Stochastic Variational Gaussian Process Regression

To enhance the efficiency of GP training and evaluations, we employ a Stochastic Variational Gaussian Process (SVGP) model [23], which approximates the exact GP model with a set of inducing points and their respective outputs to effectively "summarize" the data. By utilizing a number of inducing points and labels $m < n$, this approach decreases the computational burden of kernel matrix inversion from $\mathcal{O}(n^3)$ to $\mathcal{O}(mn^2)$ [9], vastly speeding up the training process and evaluation. In practice, it means that during training, the inducing points and target values are optimized alongside the kernel and likelihood parameters. Moreover, the optimization process takes place over several data batches within each epoch, meaning that for every epoch, parameter optimizations are carried out as many times as there are data batches. Even though we use SVGP models, they will simply be referred to as GPs for ease.

III. Gaussian Process model learning

This section presents the Gaussian process models that dictate the full vehicle dynamics of the 1:10-scale car-like robot. It contrasts the conventional training approach with the multi-step training framework, illustrating the effectiveness of the proposed training framework in enhancing long-term predictive capabilities.

A. Gaussian Process-based Vehicle Model Formulation

In this study, the vehicle dynamics are fully described by three GP models that describe the longitudinal acceleration a_x, lateral acceleration a_y, and angular acceleration a_ω, respectively. The input features of the GP model are the control inputs and velocity states,

$$\boldsymbol{x} = [\tau, \delta, v_x, v_y, \omega]^T\ , \tag{11}$$

in which τ is the throttle input, δ is the steering input. v_x, v_y, and ω are the longitudinal, lateral, and angular velocity states respectively.

To learn three GPs that describe the accelerations, a data set $\mathcal{D}$ with n input and output points $(\boldsymbol{x}_i, a_{x_i}, a_{y_i}, a_{\omega_i})$ collected by driving the robot around with a joystick from [5] was used:

$$\begin{aligned}\mathcal{D} = \big\{\ &\boldsymbol{a}_x = [a_{x_0}; \ldots; a_{x_n}] \in \mathbb{R}^n, \\ &\boldsymbol{a}_y = [a_{y_0}; \ldots; a_{y_n}] \in \mathbb{R}^n, \\ &\boldsymbol{a}_\omega = [a_{\omega_0}; \ldots; a_{\omega_n}] \in \mathbb{R}^n, \\ &\boldsymbol{X} = \left[\boldsymbol{x}_0^T; \ldots; \boldsymbol{x}_n^T\right] \in \mathbb{R}^{n\times n_d}\big\}.\end{aligned} \tag{12}$$

Treating each acceleration dimension separately, the GP models are trained as described in II-C. Once the GP models are trained, the accelerations that govern the robot vehicle dynamics can be represented by the following system of GP approximations,

$$\begin{cases}\boldsymbol{a}_x = \mathcal{N}(\boldsymbol{\mu}^{a_x}(\boldsymbol{X}), \boldsymbol{\Sigma}^{a_x}(\boldsymbol{X})) \\ \boldsymbol{a}_y = \mathcal{N}(\boldsymbol{\mu}^{a_y}(\boldsymbol{X}), \boldsymbol{\Sigma}^{a_y}(\boldsymbol{X})) \\ \boldsymbol{a}_\omega = \mathcal{N}(\boldsymbol{\mu}^{a_\omega}(\boldsymbol{X}), \boldsymbol{\Sigma}^{a_\omega}(\boldsymbol{X}))\end{cases}\ . \tag{13}$$

Subsequently, new predictions at n_* test points stored in $\boldsymbol{X_*}$ can be made according to

$$\begin{aligned}\boldsymbol{\mu}^a &= \boldsymbol{K}^a(\boldsymbol{X}_*, \boldsymbol{X})\left[\boldsymbol{K}^a(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2\boldsymbol{I}\right]^{-1}\mathbf{y} \\ \boldsymbol{\Sigma}^a &= \boldsymbol{K}^a(\boldsymbol{X}_*, \boldsymbol{X}_*) \\ &\quad - \boldsymbol{K}^a(\boldsymbol{X}_*, \boldsymbol{X})\left[\boldsymbol{K}^a(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2\boldsymbol{I}\right]^{-1}\boldsymbol{K}^a(\boldsymbol{X}, \boldsymbol{X}_*) \\ &\qquad\qquad \forall\ a \in \{a_x, a_y, a_\omega\}.\end{aligned} \tag{14}$$

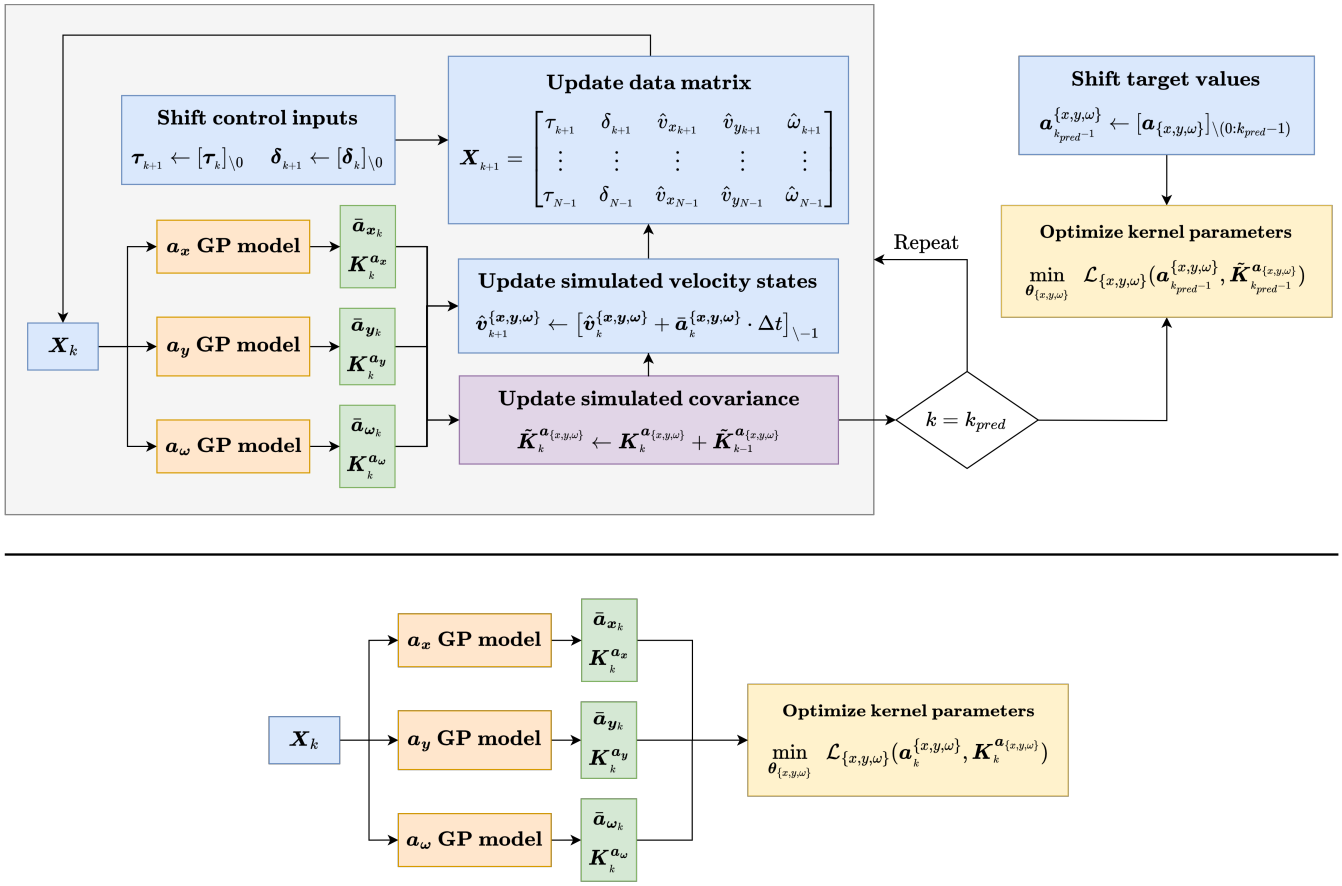


Fig. 2: Two training frameworks visualized as block diagrams. **Bottom**: In the conventional Gaussian Process (GP) training framework, the predicted covariance matrix $\boldsymbol{K}$ is directly passed to the optimization function to optimize the kernel parameters and inducing points. **Top**: In the proposed Multi-Step Gaussian Process (MSGP) training framework, the predicted mean outputs are used to simulate k_{pred} steps ahead accelerations and covariance matrix. The k_{pred} steps ahead covariance matrix is then passed into the optimization function to optimize the kernel parameters and inducing points.

with $\boldsymbol{K}^a(\boldsymbol{X},\boldsymbol{X}) \in \mathbb{R}^{n\times n}$, $\boldsymbol{K}^a(\boldsymbol{X}_*,\boldsymbol{X}_*) \in \mathbb{R}^{n_*\times n_*}$, $\boldsymbol{K}^a(\boldsymbol{X},\boldsymbol{X}_*) \in \mathbb{R}^{n\times n_*}$, $\boldsymbol{K}^a(\boldsymbol{X}_*,\boldsymbol{X}) = \boldsymbol{K}^a(\boldsymbol{X},\boldsymbol{X}_*)^T$. Note that there is now a distinct set of GP predictive equations for each acceleration dimension, each based on its own customized kernel function.

The GP models were trained using Python's GPyTorch library with 3200 data points, which corresponds to 320 seconds of recorded training data, divided into an $80/20\%$ percent ratio for training and validation, respectively. To speed up training, the exact GP is approximated using 200 inducing points.

B. Standard Gaussian Process Model Training

Fig. 2 (bottom) shows the conventional GP training framework. The GP models are evaluated at the training inputs $\boldsymbol{X}$, and the resulting covariance matrices are utilized in the optimization function as specified in (9) to update the kernel hyperparameters and inducing points and labels. This process is repeated for each minibatch of data within each epoch, resulting in # *minibatches*$\times$*epochs* of parameter optimizations.

Algorithm 1 outlines the conventional training approach in pseudocode. The model fitting results can be found in Table I. The capability of the GP models trained under this framework to make long-term predictions was assessed by initializing 30-step ahead (which is the MPC horizon length N_{pred}) predictions at various points along the training dataset and checking how well they match with actual recorded data (Fig. 3a). The plots depict recorded longitudinal, lateral, and angular velocity data from the robot car. The gray lines illustrate the long-term mean predictions of the velocity states and the surrounding shaded gray regions indicate the uncertainties provided by the GP. The velocity predictions often go out of distribution, as evidenced by the cases where predictions shoot away from where they were initialized and eventually flatten out. This behavior reflects the intrinsic workings of the GP model, highlighting its limitations when faced with unfamiliar data points. For clarity, only a brief data excerpt is displayed; readers seeking comprehensive outcomes on longer data sets including validation sets are directed to the cover report.

Algorithm 1 Standard Gaussian Process Training

Input: $\boldsymbol{a}_x, \boldsymbol{a}_y, \boldsymbol{a}_\omega, \boldsymbol{X}$, batchsize N,
  initial inducing points and labels
Output: mean, covariance, $\log p(\mathbf{y}|\boldsymbol{X}, \boldsymbol{\theta})$ for all GPs
for $i = 0$ **to** `epoch_max` **do**
  for $j = 0$ **to** `batch_iter_max` **do**
    ▷ Update data batch
    $\boldsymbol{a}_{x_j} \leftarrow [\boldsymbol{a}_x]_{jN:(j+1)N-1}$
    $\boldsymbol{X}_j \leftarrow [\boldsymbol{X}]_{jN:(j+1)N-1,.}$
    $\mathcal{D}_{x_j} \leftarrow \{\boldsymbol{a}_{x_j}, \boldsymbol{X}_j\}$
    ▷ Obtain mean and covariance
    $\overline{\boldsymbol{a}}_{x_j}, \boldsymbol{K}_j^{a_x} = \texttt{SVGP\_model\_ax}(\boldsymbol{X}_j)$
    ▷ Update optimization parameters
    $\boldsymbol{\theta}_{x_{j+1}} \leftarrow \text{argmin}_{\boldsymbol{\theta}}\ \mathcal{L}_j(\boldsymbol{a}_{x_j}, \boldsymbol{K}_j^{a_x}, \boldsymbol{\theta}_{x_j})$
  end for
  for $j = 0$ **to** `batch_iter_max` **do**
    $\boldsymbol{a}_{y_j} \leftarrow [\boldsymbol{a}_y]_{jN:(j+1)N-1}$
    $\boldsymbol{X}_j \leftarrow [\boldsymbol{X}]_{jN:(j+1)N-1,.}$
    $\mathcal{D}_{y_j} \leftarrow \{\boldsymbol{a}_{y_j}, \boldsymbol{X}_j\}$
    $\overline{\boldsymbol{a}}_{y_j}, \boldsymbol{K}_j^{a_y} = \texttt{SVGP\_model\_ay}(\boldsymbol{X}_j)$
    $\boldsymbol{\theta}_{y_{j+1}} \leftarrow \text{argmin}_{\boldsymbol{\theta}}\ \mathcal{L}_j(\boldsymbol{a}_{y_j}, \boldsymbol{K}_j^{a_y}, \boldsymbol{\theta}_{y_j})$
  end for
  for $j = 0$ **to** `batch_iter_max` **do**
    $\boldsymbol{a}_{\omega_j} \leftarrow [\boldsymbol{a}_\omega]_{jN:(j+1)N-1}$
    $\boldsymbol{X}_j \leftarrow [\boldsymbol{X}]_{jN:(j+1)N-1,.}$
    $\mathcal{D}_{\omega_j} \leftarrow \{\boldsymbol{a}_{\omega_j}, \boldsymbol{X}_j\}$
    $\overline{\boldsymbol{a}}_{\omega_j}, \boldsymbol{K}_j^{a_\omega} = \texttt{SVGP\_model\_aw}(\boldsymbol{X}_j)$
    $\boldsymbol{\theta}_{\omega_{j+1}} \leftarrow \text{argmin}_{\boldsymbol{\theta}}\ \mathcal{L}_j(\boldsymbol{a}_{\omega_j}, \boldsymbol{K}_j^{a_\omega}, \boldsymbol{\theta}_{\omega_j})$
  end for
end for

C. Proposed Multi-Step Gaussian Process Model Training

Fig. 2 (top) shows the proposed Multi-Step Gaussian Process (MSGP) training framework. This framework leverages the application of in-training simulation of future dynamics for a short horizon $k_{pred} < N_{pred}$ based on the GP's current level of understanding (i.e. obtained at the current epoch) to predict future dynamics. In essence, with this process, we effectively give feedback to the GP regarding its ability to make long-term predictions. Thereby, we encourage the GP models to be optimized not only for immediate prediction accuracy but also in the long term. The method can be explained as follows. The GP models as in (13) are evaluated at the training inputs $\boldsymbol{X}_k$ to obtain the mean and covariance matrix of the acceleration states,

$$\begin{aligned} \boldsymbol{a}_{x_k} &= \mathcal{N}(\overline{\boldsymbol{a}}_{x_k}, \boldsymbol{K}_k^{a_x}) \\ \boldsymbol{a}_{y_k} &= \mathcal{N}(\overline{\boldsymbol{a}}_{y_k}, \boldsymbol{K}_k^{a_y}) \\ \boldsymbol{a}_{\omega_k} &= \mathcal{N}(\overline{\boldsymbol{a}}_{\omega_k}, \boldsymbol{K}_k^{a_\omega}) \end{aligned} \ . \tag{15}$$

Using the mean accelerations, the one-step ahead velocity states $\hat{\boldsymbol{v}}_{k+1}$ can be predicted by multiplying the predicted mean acceleration with time step Δt and adding it to the current velocity,

$$\begin{aligned} \hat{\boldsymbol{v}}_{x_{k+1}} &= \hat{\boldsymbol{v}}_{x_k} + \overline{\boldsymbol{a}}_{x_k} \cdot \Delta t \\ \hat{\boldsymbol{v}}_{y_{k+1}} &= \hat{\boldsymbol{v}}_{y_k} + \overline{\boldsymbol{a}}_{y_k} \cdot \Delta t \\ \hat{\boldsymbol{\omega}}_{k+1} &= \hat{\boldsymbol{\omega}}_k + \overline{\boldsymbol{a}}_{\omega_k} \cdot \Delta t \ . \end{aligned} \tag{16}$$

The one-step-ahead simulated velocity states partially form the new inputs for the next GP evaluation. The throttle $\boldsymbol{\tau}$ and steering $\boldsymbol{\delta}$ input vectors are also advanced from k to $k+1$ discarding their first values to align them with the updated velocity states

$$\begin{aligned} \boldsymbol{\tau}_{k+1} &= [\boldsymbol{\tau}_k]_{\backslash 0} \\ \boldsymbol{\delta}_{k+1} &= [\boldsymbol{\delta}_k]_{\backslash 0} \ . \end{aligned} \tag{17}$$

Because the control input vectors are now reduced by one element and we lack control input data for indices beyond the existing data set or data batch size, it is necessary to omit the last elements of the simulated velocities to ensure that the dimensions of the input data matrix remain consistent

$$\begin{aligned} \hat{\boldsymbol{v}}_{x_{k+1}} &= [\hat{\boldsymbol{v}}_{x_{k+1}}]_{\backslash -1} \\ \hat{\boldsymbol{v}}_{y_{k+1}} &= [\hat{\boldsymbol{v}}_{y_{k+1}}]_{\backslash -1} \\ \hat{\boldsymbol{\omega}}_{k+1} &= [\hat{\boldsymbol{\omega}}_{k+1}]_{\backslash -1} \ . \end{aligned} \tag{18}$$

Now that the velocity states and control inputs have been advanced from k to $k+1$ and have been sized properly, the updated input data matrix $\boldsymbol{X}_{k+1}$ for the next GP evaluation can be reconstructed

$$\boldsymbol{X}_{k+1} \leftarrow \left[\boldsymbol{\tau}_{k+1}, \boldsymbol{\delta}_{k+1}, \hat{\boldsymbol{v}}_{x_{k+1}}, \hat{\boldsymbol{v}}_{y_{k+1}}, \hat{\boldsymbol{\omega}}_{k+1}\right] \ . \tag{19}$$

Note that the velocity state variables are simulated future velocities, derived from the acceleration outputs provided by the Gaussian Process (GP). In this way, we employ the GP at its current level of understanding (referred to as the current epoch) to predict future dynamics during training. This simulation process is repeated several cycles until a desired k_{pred} horizon is reached. At this point, the covariance matrix $\boldsymbol{K}_{pred-1}$, detailing the correlations between data point pairs at k_{pred-1}, can be passed into the optimization function as specified in (9). However, instead of using this matrix as it is within the optimization process, it is updated to incorporate previous uncertainties as well. For computational efficiency and to ensure the matrix remains positive definite (solver requirement), the covariance matrix from consecutive time steps $k-1$ and k were treated to be independent and summed to each other. This simplification was made under the assumptions that:

1) The horizon for simulating future dynamics is considerably short, with $k_{pred} << N_{pred}$.
2) The changes in data points between successive time instants k and $k-1$ are so small, resulting in negligible differences in the structure of their covariance matrix. This leads to an intensified correlation among data points already correlated (whether positively or negatively), and an increase in variances, which remain positive by nature.

The covariance matrix for each GP model is updated according to (20). To maintain correct dimensionality with the training input data matrix $\boldsymbol{X}_k$, the first row and column of the updated covariance are discarded.

$$\begin{aligned}\tilde{\boldsymbol{K}}_k^{a_x} &\leftarrow \left[\tilde{\boldsymbol{K}}_{k-1}^{a_x}\right]_{\setminus 0,\setminus 0} + \boldsymbol{K}_k^{a_x} \\ \tilde{\boldsymbol{K}}_k^{a_y} &\leftarrow \left[\tilde{\boldsymbol{K}}_{k-1}^{a_y}\right]_{\setminus 0,\setminus 0} + \boldsymbol{K}_k^{a_y} \\ \tilde{\boldsymbol{K}}_k^{a_\omega} &\leftarrow \left[\tilde{\boldsymbol{K}}_{k-1}^{a_\omega}\right]_{\setminus 0,\setminus 0} + \boldsymbol{K}_k^{a_\omega}\ ,\end{aligned} \tag{20}$$

in which $\boldsymbol{K}_k^{a_{\{x,y,\omega\}}} = \boldsymbol{K}_k^{a_{\{x,y,\omega\}}}(\boldsymbol{X}_k, \boldsymbol{X}_k)$ is the covariance matrix given upon GP evaluation at the current time step and $\tilde{\boldsymbol{K}}_{k-1}^{a_{\{x,y,\omega\}}} = \tilde{\boldsymbol{K}}_{k-1}^{a_{\{x,y,\omega\}}}(\boldsymbol{X}_{k-1}, \boldsymbol{X}_{k-1})$ is the covariance matrix given by the GP at the previous time step.

This process described in (15)-(20) repeats until the dynamics and covariance matrix have been propagated a desired k_{pred} steps ahead into the future. Then the $k_{step}-1$ covariance matrices will be fed into the optimization function along with the target values advanced the same number of k-steps ahead

$$\begin{aligned}\boldsymbol{a}_x &= [\boldsymbol{a}_x]_{\setminus(0:k_{pred}-1)} \\ \boldsymbol{a}_y &= [\boldsymbol{a}_y]_{\setminus(0:k_{pred}-1)} \\ \boldsymbol{a}_\omega &= [\boldsymbol{a}_\omega]_{\setminus(0:k_{pred}-1)}\ ,\end{aligned} \tag{21}$$

to optimize the kernel hyperparameters, inducing points and labels. This process is repeated for each minibatch of data within each epoch, resulting in *# minibatches* $\times$ *epochs* of parameter optimizations. **Algorithm 2** outlines the multi-step ahead framework through pseudocode in detail.

D. Results

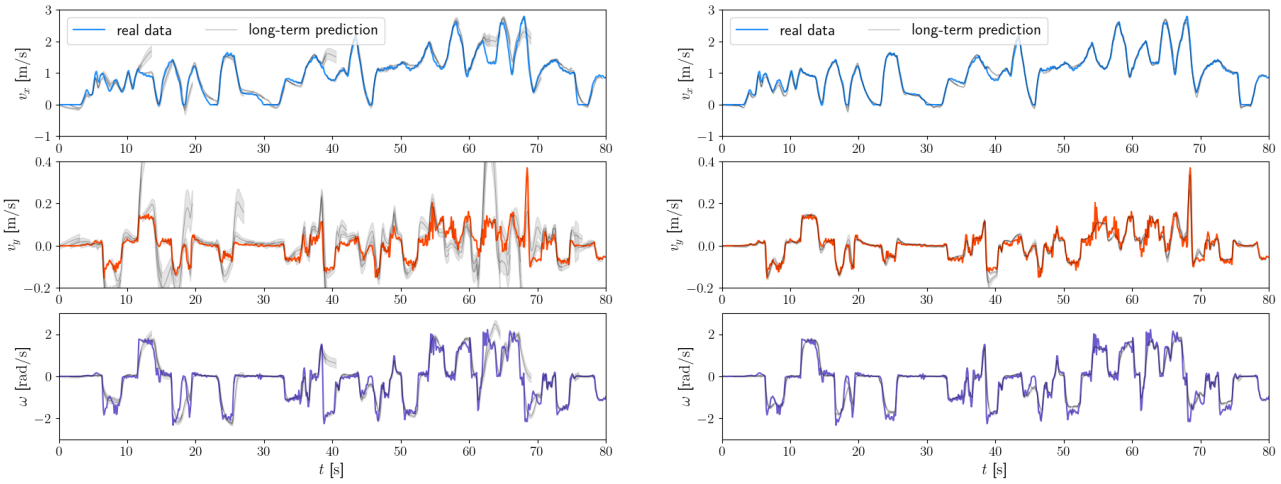
Fig. 3b shows how the predicted velocity trajectories by the MSGP models closely align with the actual velocity data. To get an understanding of the model's capability for reliable long-term predictinos, it is essential to examine the internal modifications of the model, particularly the adjustments in the kernel hyperparameters.

1) Recalibrated sensitivity: Upon looking at the kernel lengthscales, the general trend is that all kernel lengthscales have increased, meaning that the acceleration mean functions have smoothened out. For the lateral acceleration MSGP model, increased lengthscales for steering ℓ_δ, and angular velocity ℓ_ω input features reduce the model's sensitivity to these variables (Fig. 4b). The initial GP model was overly sensitive, reacting rigorously to minimal input changes as evident from the rapid predicted lateral velocity changes at $t = 12$ s for example Fig. 3a. It can be seen that at the same time in Fig. 3b, this behavior is eliminated. The lengthscales for the longitudinal (Fig. 4a) and the angular acceleration (Fig. 4c) GP model have also increased, calibrating the models to be slightly less sensitivity to the input features, aligning its responses more closely with realistic vehicle dynamics.

2) Rectified coupling: However, upon looking closely, there are specific lengthscales that have changed more than others. For the longitudinal dynamics, the lengthscales for throttle ℓ_τ and longitudinal velocity ℓ_{v_x} have decreased, meaning that

Algorithm 2 Multi-Step Gaussian Process Training (Proposed)

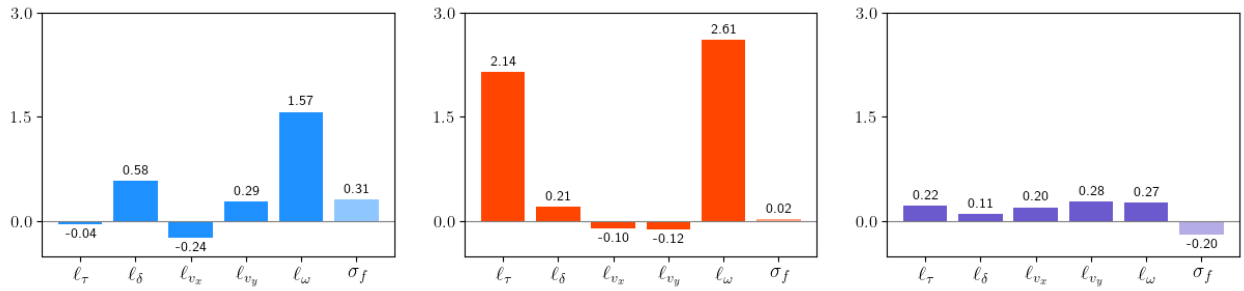
Input: $\boldsymbol{a}_x, \boldsymbol{a}_y, \boldsymbol{a}_\omega, \boldsymbol{X}, N, k_{pred}$,
  initial inducing points and labels
Output: mean, covariance, $\log p(\mathbf{y}|\boldsymbol{X}, \boldsymbol{\theta})$ for all GPs
for $i = 0$ **to** `epoch_max` **do**
  for $j = 0$ **to** `batch_iter_max` **do**
    ▷ Initialize data batches
    $\boldsymbol{X}_k \leftarrow [\boldsymbol{X}]_{jN:(j+1)N-1,\cdot}$
    $\boldsymbol{a}_{x_j} \leftarrow [\boldsymbol{a}_x]_{jN:(j+1)N-1}$
    $\boldsymbol{a}_{y_j} \leftarrow [\boldsymbol{a}_y]_{jN:(j+1)N-1}$
    $\boldsymbol{a}_{\omega_j} \leftarrow [\boldsymbol{a}_\omega]_{jN:(j+1)N-1}$
    $\mathcal{D}_{x_j} \leftarrow \{\boldsymbol{a}_{x_j}, \boldsymbol{X}_k\}$
    $\mathcal{D}_{y_j} \leftarrow \{\boldsymbol{a}_{y_j}, \boldsymbol{X}_k\}$
    $\mathcal{D}_{\omega_j} \leftarrow \{\boldsymbol{a}_{\omega_j}, \boldsymbol{X}_k\}$
    ▷ Initialize in-training simulated cov. matrices
    $\tilde{\boldsymbol{K}}_{k-1}^{a_x}, \tilde{\boldsymbol{K}}_{k-1}^{a_y}, \tilde{\boldsymbol{K}}_{k-1}^{a_\omega} \leftarrow \mathbf{0}_{N\times N}$
    ▷ Initialize velocity estimates
    $\hat{\boldsymbol{v}}_{x_k}, \hat{\boldsymbol{v}}_{y_k}, \hat{\boldsymbol{\omega}}_k \leftarrow [\boldsymbol{X}_k]_{\cdot,2}, [\boldsymbol{X}_k]_{\cdot,3}, [\boldsymbol{X}_k]_{\cdot,4}$
    ▷ Simulate future predictions
    for $k = 0$ **to** `k_preds` **do**
      ▷ Obtain mean and covariance
      $\overline{\boldsymbol{a}}_{x_k}, \boldsymbol{K}_k^{a_x} =$ `SVGP_model_ax`$(\boldsymbol{X}_k)$
      $\overline{\boldsymbol{a}}_{y_k}, \boldsymbol{K}_k^{a_y} =$ `SVGP_model_ay`$(\boldsymbol{X}_k)$
      $\overline{\boldsymbol{a}}_{\omega_k}, \boldsymbol{K}_k^{a_\omega} =$ `SVGP_model_aw`$(\boldsymbol{X}_k)$
      ▷ Compound covariance by summation
      if $k == 0$ **then**
        $\tilde{\boldsymbol{K}}_k^{a_x} \leftarrow \tilde{\boldsymbol{K}}_{k-1}^{a_x} + \tilde{\boldsymbol{K}}_k^{a_x}$
        $\boldsymbol{K}_k^{a_y} \leftarrow \tilde{\boldsymbol{K}}_{k-1}^{a_y} + \boldsymbol{K}_k^{a_y}$
        $\tilde{\boldsymbol{K}}_k^{a_\omega} \leftarrow \tilde{\boldsymbol{K}}_{k-1}^{a_\omega} + \boldsymbol{K}_k^{a_\omega}$
      else
        $\tilde{\boldsymbol{K}}_k^{a_x} \leftarrow \left[\tilde{\boldsymbol{K}}_{k-1}^{a_x}\right]_{\setminus 0,\setminus 0} + \boldsymbol{K}_k^{a_x}$
        $\tilde{\boldsymbol{K}}_k^{a_y} \leftarrow \left[\tilde{\boldsymbol{K}}_{k-1}^{a_y}\right]_{\setminus 0,\setminus 0} + \boldsymbol{K}_y^{a_y}$
        $\tilde{\boldsymbol{K}}_k^{a_\omega} \leftarrow \left[\tilde{\boldsymbol{K}}_{k-1}^{a_\omega}\right]_{\setminus 0,\setminus 0} + \boldsymbol{K}_\omega^{a_\omega}$
      end if
      ▷ Update velocity state estimates
      $\hat{\boldsymbol{v}}_{x_{k+1}} \leftarrow [\hat{\boldsymbol{v}}_{x_k} + \overline{\boldsymbol{a}}_{x_k} \cdot \Delta t]_{\setminus -1}$
      $\hat{\boldsymbol{v}}_{y_{k+1}} \leftarrow [\hat{\boldsymbol{v}}_{y_k} + \overline{\boldsymbol{a}}_{y_k} \cdot \Delta t]_{\setminus -1}$
      $\hat{\boldsymbol{\omega}}_{k+1} \leftarrow [\hat{\boldsymbol{\omega}}_k + \overline{\boldsymbol{a}}_{\omega_k} \cdot \Delta t]_{\setminus -1}$
      ▷ Update input data matrix
      $\boldsymbol{\tau}_{k+1}, \boldsymbol{\delta}_{k+1} \leftarrow [\boldsymbol{X}_k]_{\setminus 0,0}, [\boldsymbol{X}_k]_{\setminus 0,1}$
      $\boldsymbol{X}_{k+1} \leftarrow \left[\boldsymbol{\tau}_{k+1}, \boldsymbol{\delta}_{k+1}, \hat{\boldsymbol{v}}_{x_{k+1}}, \hat{\boldsymbol{v}}_{y_{k+1}}, \hat{\boldsymbol{\omega}}_{k+1}\right]$
    end for
    ▷ Shift target values `k_preds`-1 steps forward
    $\boldsymbol{a}_x \leftarrow [\boldsymbol{a}_{x_j}]_{\setminus(0:k_{pred}-1)}$
    $\boldsymbol{a}_y \leftarrow [\boldsymbol{a}_{y_j}]_{\setminus(0:k_{pred}-1)}$
    $\boldsymbol{a}_\omega \leftarrow [\boldsymbol{a}_{\omega_j}]_{\setminus(0:k_{pred}-1)}$
    ▷ Update optimization parameters
    $\boldsymbol{\theta}_{x_{j+1}} \leftarrow \text{argmin}_{\boldsymbol{\theta}}\ \mathcal{L}_j(\boldsymbol{a}_x, \tilde{\boldsymbol{K}}_{preds-1}^{a_x}, \boldsymbol{\theta}_{x_j})$
    $\boldsymbol{\theta}_{y_{j+1}} \leftarrow \text{argmin}_{\boldsymbol{\theta}}\ \mathcal{L}_j(\boldsymbol{a}_y, \tilde{\boldsymbol{K}}_{preds-1}^{a_y}, \boldsymbol{\theta}_{y_j})$
    $\boldsymbol{\theta}_{\omega_{j+1}} \leftarrow \text{argmin}_{\boldsymbol{\theta}}\ \mathcal{L}_j(\boldsymbol{a}_\omega, \tilde{\boldsymbol{K}}_{preds-1}^{a_\omega}, \boldsymbol{\theta}_{\omega_j})$
  end for
end for



(a) Velocity trajectory predictions GP model

(b) Velocity trajectory predictions by MSGP model

Fig. 3: This figure contrasts standard Gaussian process (GP) model (a) with the Multi-Step Gaussian process (MSGP) (b) for long-term velocity prediction, over a prediction horizon $N_{pred} = 30$. The velocity trajectory predictions are made using the accelerations predicted by the GP/MSGP models and integrated using forward Euler method. Colored lines denote experimental velocity data recorded over time: longitudinal velocity (blue), lateral velocity (orange), and angular velocity (purple). Gray lines illustrate the mean velocity predictions, initialized at various points along the recorded data. The velocity trajectories based on this GP model often go out-of-distribution, denoted by cases where the predictions rapidly diverge from where they were initialized and eventually flatten out. On the other hand, velocity predictions based on the MSGP model closely mirror actual data, demonstrating stable long-term predictions with reduced uncertainty. Note that even though time is indicated on the x-axis, it is important to stress that the learned dynamics do not depend on time, as it is not an input feature. The inputs are strictly the velocity states and control inputs.



(a) a_x kernel parameter changes

(b) a_y kernel parameter changes

(c) a_w kernel parameter changes

Fig. 4: The changes in kernel hyperparameters of the MSGP model relative to the standard GP models. The kernel parameters are length scale for the throttle input ℓ_τ, steering input ℓ_δ, longitudinal velocity ℓ_{v_x}, lateral velocity ℓ_{v_y}, angular velocity ℓ_ω, and output scale σ_f. The parameters influence the sensitivity, coupling of the dynamics, and the uncertainty of the predictions.

the longitudinal acceleration dynamics have become more responsive to these input features. Whereas the lengthscales for steering ℓ_δ, lateral velocity ℓ_{v_y} and angular velocity ℓ_ω have increased, meaning that the longitudinal acceleration has become less sensitive to the lateral dynamics input features. These lengthscale adaptations mitigate the longitudinal model's overreaction to lateral dynamics, as seen at $t = 12$, $t = 38$, and $t = 62$ s in Fig. 3a, where severely overestimated lateral accelerations led to exaggerated longitudinal velocities and thereby instability. We can see that with the recalibrated lengthscales, the predictions at the same time-steps in Fig. 3b remain stable. The same phenomenon can be observed in lateral acceleration dynamics (Fig. 4b), which have become less responsive to the throttle input feature. This recalibration of the kernel parameters ensures that the longitudinal and lateral dynamics have become more decoupled, ensuring that

TABLE I: Assessment of long-term predictions of the longitudinal $\boldsymbol{v_x}$, lateral $\boldsymbol{v_y}$ and angular $\boldsymbol{\omega}$ velocities by the Gaussian Process (GP) model trained under the standard framework.

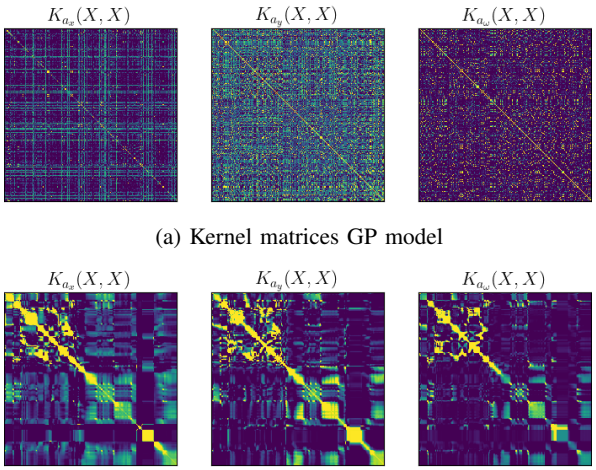
Metric	$\boldsymbol{v_x}$	$\boldsymbol{v_y}$	$\boldsymbol{\omega}$
RMSE	0.1733	0.1022	0.4499
MAE	0.1083	0.0534	0.2183
Average variance	0.6796e-3	0.3465e-3	2.334e-3
Training time [min]	1:12		
Epochs, optimization steps	400, 1600		

Metrics used are root mean square error (RMSE), mean absolute error (MAE), average variance (Avg Var).

TABLE II: Assessment of long-term predictions of the longitudinal $\boldsymbol{v_x}$, lateral $\boldsymbol{v_y}$ and angular $\boldsymbol{\omega}$ velocities by the Multi-Step Gaussian Process (MSGP) model

Metric	$\boldsymbol{v_x}$	$\boldsymbol{v_y}$	$\boldsymbol{\omega}$
RMSE	0.1244	0.0293	0.2519
MAE	0.0880	0.0165	0.1656
Avg. Var	4.0740e-5	1.1995e-5	4.6023e-4
$\Delta\%$ RMSE* [%]	-28	-71	-44
$\Delta\%$ MAE* [%]	-11	-36	-12
$\Delta\%$ Average Variance* [%]	-94	-97	-80
Training time [min]	3:57		
Epochs, optimization steps	250, 1000		

Metrics used are root mean square error (RMSE), mean absolute error (MAE), average variance (Avg Var).
*The percentual change $\Delta\%$ is computed w.r.t. the standard GP training method.



(a) Kernel matrices GP model

(b) Kernel matrices MSGP model

Fig. 5: Heatmaps of posterior kernel matrices for each acceleration dimension a_x, a_y, and a_ω. Kernel matrices of the standard GP model method show sparse correlations other than the data points with itself (the diagonal). The kernel matrix of the MSGP model, however, reveals distinct correlated clusters, demonstrating its effectiveness in uncovering vehicle dynamics patterns.

while both acceleration dimensions do influence each other, they do not disproportionately affect each other, aligning the model more closely with physical vehicle dynamics principles.

3) Reduced uncertainty: Now that the long-term predictions are stable and the predicted future dynamics consistently correlate with the training data, the uncertainty reduction term in the GP's predictive equations for the covariance (7) is activated, leading to a decrease in variance associated with the mean predictions. The MSGP model's predictions for long-term velocity trajectories have shown a significant reduction in variance, with reductions of 94%, 97%, and 80% for longitudinal, lateral, and angular velocity, respectively, compared to the uncertainties associated with the standard GP model (Table II). Furthermore, to prevent overfitting and ensure the models generalize well, the GP/MSGP models have also been tested on a validation data set, as detailed in the cover report.

The computational load during training primarily stems from inverting the matrix $\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})$ (time complexity $\mathcal{O}(n^3)$). This inversion inevitably needs to be carried out in each optimization cycle to facilitate the simulation of long-term predictions. Fortunately, this matrix inversion only needs to be done once per optimization cycle and can be reused for each consecutive future prediction within that cycle, since it only depends on the training data. Consequently, to minimize the computational burden, it is essential to maintain a small value for k_{pred} ($k_{pred} << N_{pred}$). This approach limits the number of matrix multiplications, which have a time complexity of $\mathcal{O}(n^2)$, thereby keeping the computational load manageable. To our advantage, choosing $k_{pred} = 2$ is proven to be adequate for achieving reliable long-term predictions in practice. Although training time has lengthened, there's a noticeable improvement in the accuracy of long-term predictions, as evidenced by reductions in root mean square error (RMSE), mean absolute error (MAE), and average variance (Table I II). Moreover, fewer training cycles are required to learn the dynamics — 400 for the Gaussian Process (GP) and 250 for the Multi-Step Gaussian Process (MSGP). Given that both training methodologies use an identical dataset, it is apparent that the MSGP framework is more efficient at learning insights and understanding the underlying vehicle dynamics compared to the standard GP training approach.

Another way to get deeper insights into the effects of kernel tuning is shown by looking at the kernel matrices. Fig. 5 shows the posterior kernel matrices of a data batch arranged in chronological order rather than in input space (so that is why the kernel heatmaps do not show the traditional patterns of a squared exponential kernel) to demonstrate the effects between the two training methods. Upon analyzing the posterior kernel matrices of the GP models, two primary insights were noted. The kernel matrix obtained from the conventional training approach exhibits pronounced values along its diagonal, as expected, yet it displays a sparse and unstructured correlation pattern among different data points, indicating a lack of discernible relationships beyond direct similarities (as illustrated in Fig. 5a). Conversely, the kernel matrix from the multi-step ahead training approach shows a contrasting scenario (Fig. 5b). Beyond showcasing diagonal variances, it also unveils clear covariance clusters. These clusters imply that certain data point groups share stronger correlations and that particular input

sequences have heightened predictability concerning vehicle acceleration. This exemplifies the efficacy of the training framework's adeptness at uncovering meaningful relations and input patterns from vehicle dynamics data.

IV. Multi-step Gaussian process-based Controller

This section showcases the effectiveness of the MSGP model by applying it to an MPCC controller (MSGP-MPCC). It begins with an overview of the control formulation and the setup for simulation. Finally, the performance of the MSGP-MPCC is benchmarked against a baseline controller using the standard GP model (GP-MPCC). The GP-based controller and simulation setup used is based on the works of [5].

A. Controller formulation

This study employs a Model Predictive Contouring Control (MPCC) strategy that decouples state references from a time schedule [4], [19], [20], enhancing flexibility in computing control actions as opposed to traditional MPC. The application of this controller is demonstrated in simulation on a 1:10 scale car-like robot. Predictions within the MPCC framework, rely on either the GP model or the MSGP model. Fig. 1 shows a schematic of the control architecture. The dynamics of the (MS)GP-MPCC are described as,

$$\begin{aligned} \dot{v}_x &= \overline{a}_{x_*} \\ \dot{v}_y &= \overline{a}_{y_*} \\ \dot{\omega} &= \overline{a}_{\omega_*} \\ \dot{\theta} &= \omega \\ \dot{x} &= v_x \cos(\theta) - v_y \sin(\theta) \\ \dot{y} &= v_y \sin(\theta) + v_x \cos(\theta) \ , \end{aligned} \tag{22}$$

in which the accelerations are the mean values given by evaluating the GP models at the input-state combination at hand (i.e. the test point) $\boldsymbol{x}_* = \big[\tau, \delta, v_x, v_y, \omega\big]$, according to the GP mean predictive equations in (14). Given the prediction model, the control problem can then be formulated as [21],

$$\begin{aligned} \min_{\boldsymbol{u}_{t+k} \in \mathcal{U}} \quad & \sum_{k=0}^{N-1} q_1 \cdot (\dot{s}_{t+k} - \dot{s}_{ref})^2 + q_2 \cdot (e^c_{t+k})^2 \\ & \qquad + \boldsymbol{u}_{t+k}^T R_u \boldsymbol{u}_{t+k} + p_1 \cdot \varepsilon^2 \\ \text{s.t.} \quad & \boldsymbol{x}_0 = \boldsymbol{x}(0) \\ & s_0 = s(0) \\ & \boldsymbol{x}_{t+k+1} = \boldsymbol{f}(\boldsymbol{x}_{t+k}, \boldsymbol{u}_{t+k}) \\ & s_{t+k+1} = f(s_{t+k}) \\ & \boldsymbol{u}_{t+k} \in \mathcal{U} \\ & 0 \leq s_{t+k} \leq L \\ & -(1+\varepsilon) \leq \frac{e^c_{t+k}}{\frac{l_{width}}{2}} \leq 1 + \varepsilon \\ & \forall k \in [0, N-1] \quad . \end{aligned} \tag{23}$$

In this MPCC formulation, s is the path along which the robot progresses, $\dot{s}$ is the robot's velocity along this path, and $\dot{s}_{ref}$ is the reference velocity in m/s. $\mathcal{U}$ is the set that forms the constraints on the control inputs. Lane boundaries are defined as soft constraints on the normalized contouring error e^c using a slack variable ε. q_1, q_2 are weights on the velocity and contouring error, respectively. $R_u = \text{diag}(r_1, r_2)$ is a diagonal weighting matrix on the control inputs. p_1 is the weight on the slack variable ε.

B. Controller implementation

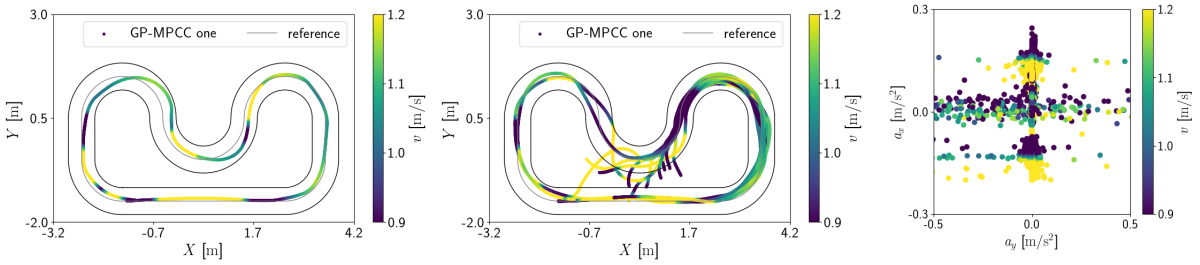
The cost function is adaptable to different control objectives, with varying priorities based on predefined weights. This flexibility allows for tuning specific dynamic behaviors, such as precise velocity tracking and minimizing lateral error for conservative driving, as well as more aggressive maneuvers where the same objectives would be less prioritized. This study aims to evaluate the models' performance under highly aggressive conditions. To achieve this objective, an emphasis is placed on promoting high speeds. This is accomplished by intentionally keeping the weight assigned to the throttle input low. This strategy encourages the controller to seek its limits and observe the response of each model in such demanding situations. The controller settings are specified in Table III. The prediction horizon of the MPCCs is set to $N_{pred} = 30$. The sampling rate is set to $T_s = 0.1$s, enabling the controller to predict three seconds ahead. ForcesPro [22] is employed for solving the nonlinear optimization problem. Control commands are transmitted at a frequency of 10 Hz via a Robot Operating System (ROS) network to control the robot within an RViz simulation environment from [5]. The robot odometry data is returned through the same ROS network.

TABLE III: MPCC controller settings

Weights					**Input constraints**		**Other**	
q_1	q_2	r_1	r_2	p_1	$\underline{\delta}, \bar{\delta}$	$\underline{\tau}, \bar{\tau}$	N_{pred}	$\dot{s}_{ref}$
1	1.2	1	0.1	1000	[-1, 1]	[0, 0.6]	30	1.0

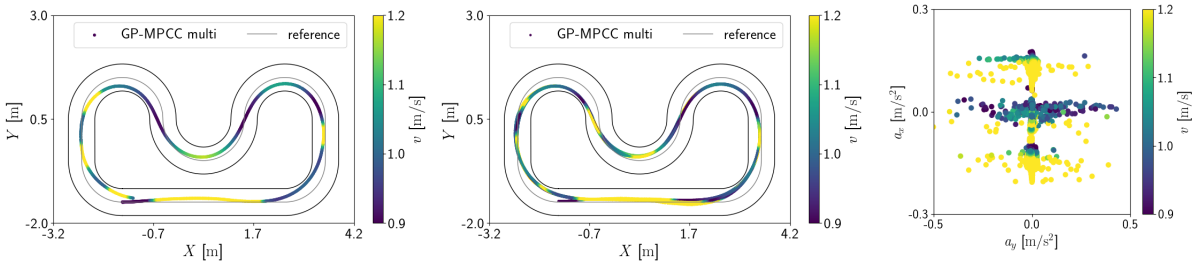
C. Results

The multi-step Gaussian Process training framework has proven its efficacy within an MPCC applied to racetrack navigation, showcasing superior performance compared to the baseline GP-MPCC. Through simulations of the control task, significant performance differences among the models were observed when looking at key performance indices such as lap time, velocity, and acceleration. The GP-MPCC (Fig. 6a) exhibits indications of "irregular" driving behavior, as evidenced by the wiggly path. This is attributed to its limited long-term predictive capabilities inherent to the method by which it was trained. In contrast, the MSGP model (Fig. 7a) which has learned to accurately predict long-term dynamics, showcased higher adeptness at navigating track curves efficiently, evident from its smooth path and tight corner cutting. The lap times support these observations: the MSGP-MPCC clocked a lap time of 15.23 s, outpacing the GP-MPCC that took 17.14 s to complete a lap (Table IV). Reliability tests provide additional insights. Upon conducting repeated simulations (10 per controller) to evaluate their reliability, it becomes apparent that the GP-MPCC struggles to navigate the robot through sharp corners and crashes most of the time



(a) 1 lap (b) 10 laps (c) GG-diagram

Fig. 6: GP-MPCC (baseline controller): (a) 1 lap tracking results (a lucky try), (b) 10 lap tracking results, and (c) GG-diagram of the accelerations achieved by the robot car. Acceleration values are presented in m/s^2 for clarity, without conversions to Gs. The grey center line represents the path reference. The inner and outer lines denote the track boundaries. The color gradient illustrates the velocity magnitude at each recorded vehicle position.



(a) 1 lap (b) 10 laps (c) GG-diagram

Fig. 7: MSGP-MPCC (proposed controller): (a) 1 lap tracking results, (b) 10 lap tracking results, and (c) GG-diagram of the accelerations achieved by the robot car.

(Fig. 6b), achieving a control success rate of only 20%. This lack of effectiveness in lateral control can be attributed to the training outcomes of the standard GP model (Figure 3a), where it is evident that the lateral and angular GP models frequently exhibit instability, leading to rapid changes in acceleration and pushing the robot into velocity states that are out of distribution. In contrast, the MSGP-MPCC demonstrates the capability to maneuver confidently through sharp turns safely seeking its limits without crashing (Fig. 7b), achieving a control success rate of 100%. The GG diagrams, depicting the achieved accelerations during the runs, provide further clarity on this matter. In the case of the GP-MPCC, a wide range of accelerations is observed across the entire acceleration plane. However, these accelerations are erratic and unsafe, as evidenced by the frequent crashes experienced by the GP-MPCC. Conversely, the GG diagram of the MSGP-MPCC portrays coupled accelerations of the same order of magnitude and at higher velocities, depicted by the color gradient, (Figure 6c, 7c), following the circular contours typical of GG diagrams. Furthermore, the velocity profiles (Fig. 8) show that the MSGP-MPCC safely maintains higher velocities, leveraging learned acceleration capabilities from the dataset. In contrast, the GP-MPCC displays more pronounced velocity variations, stemming from an erratic model that is overly sensitive.

TABLE IV: Simulation results control performance of a model predictive contouring controller based on a Gaussian process (GP) model (baseline) vs. Multi-Step Gaussian process (MSGP) model

Metric	***GP-MPCC***	***MSGP-MPCC***
Average velocity $\overline{v}$ [m/s]	0.98	1.07
Lap time 1st round t_1 [s]	17.21	15.95
Average time remainder laps $\overline{t}_{2-10}[s]$	-	15.23
Control success rate*	20%	100%

*safe lap completion without crashing

V. Conclusion and Future Work

This study introduced a new framework that incorporates in-training simulation of future dynamics and uncertainty, effectively enabling the Gaussian Process models to gain predictive foresight and uncertainty quantification capabilities, tailored for better integration with MPC. This approach significantly improves the models' predictive accuracy by adjusting kernel parameters to recalibrate the vehicle dynamics' sensitivity to various state and input variables. Our results demonstrate a notable improvement in the model's long-term predictive capabilities, evidenced by a 19% average reduction in mean error and a 90% decrease in prediction variance for the full dynamics. Such recalibration allows the extraction of deeper

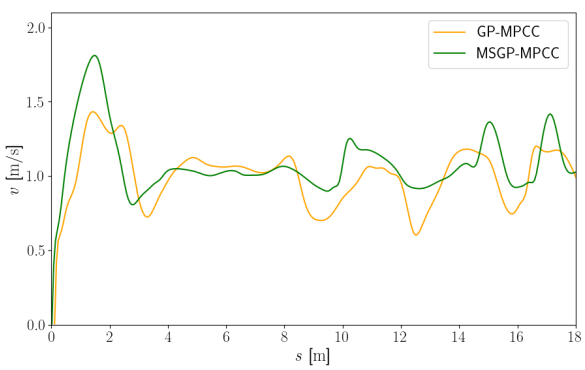


Fig. 8: Velocity profiles of a successful lap of the two controllers along the path's center line parametrized by the path progress parameter s.

insights into vehicle dynamics from the same dataset and learning more complex data-driven vehicle dynamics models, with zero added complexity at controller run-time. Its efficacy has been further demonstrated in an MPCC application. The MSGP-MPCC not only performed better by reducing lap times but also achieved a 100% control success rate. From a modeling point of view, future research could explore the possibilities of using physics-based and Gaussian process-based models jointly and using the proposed training framework to learn stable error dynamics. From a control perspective, potential directions include integrating variance dynamics into the MSGP-MPCC to leverage the calibrated uncertainties to introduce specific cautious behaviors, such as lane boundary tightening or collision avoidance applications.

REFERENCES

[1] World Health Organization. "Road Traffic Injuries", 2022.

[2] L. Svensson, M. Bujarbaruah, A. Karsolia, C. Berger, and M. Törngren, "Traction Adap- tive Motion Planning and Control at the Limits of Handling", IEEE Transactions on Control Systems Technology, vol. 30, pp. 1888–1904, 2022.

[3] D. J. Fagnant and K. Kockelman, "Preparing a Nation for Autonomous Vehicles: Opportunities, Barriers and Policy Recommendations," Transportation Research Part A: Policy and Practice, vol. 77, pp. 167-–181, 2015.

[4] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments", IEEE Robotics and Automation Letters, vol. 4, 2019.

[5] L. Lyons, G. Franzese, R Chiesa, J Kober and L. Ferranti, "Promoting safety under localized data: A Prudent Model Predictive Control for Autonomous Ground Vehicles", IEEE International Conference on Robotics and Automation (ICRA), 2024.

[6] R. Findeisen and F. Allgöwer, "An introduction to nonlinear model predictive control", 2002.

[7] G. Franzese, "Introduction to Gaussian Processes for Robotics and Control Lecture Notes", 2024.

[8] 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, 2019.

[9] C. K. Williams and C. E. Rasmussen, Gaussian processes for machine learning", vol. 2, no. 3. MIT press Cambridge, MA, 2006, pp. 13–19.

[10] L. Hewing, J. Kabza and M.N. Zeilinger, "Cautious Model Predictive Control Using Gaussian Process Regression", IEEE Transactions on Control Systems Technology, vol. 28, no. 6. 2020.

[11] R. R. Arany, "Gaussian Process Model Predictive Control for Autonomous Driving in Safety-Critical Scenarios", Master Thesis, Linköping University, 2019.

[12] W. Liu, C. Liu, G. Chen, A. Knoll, "Gaussian Process Based Model Predictive Control for Overtaking in Autonomous Driving", frontiers in Neurorobotics, 2021.

[13] L. Hewing, E. Arcari. L. P. Frohlich, M. N. Zeilinger, "On Simulation and Trajectory Prediction with Gaussian Process Dynamics", Institute for Dynamics Systems and Control, ETH Zurich, Switzerland. Bosch Center for Artificial Intelligence, Renningen, Germany. 2020.

[14] D. Duvenaud, Automatic model construction with Gaussian processes, Ph.D. thesis, University of Cambridge (2014).

[15] M. Blum, "Optimization of Gaussian Process Hyperparameters using Rprop", European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2013

[16] J. Wang, "An Intuitive Tutorial to Gaussian Process Regression", 2022.

[17] D. Bindell, "Numerical Methods for Data Science: Kernels", Cornell University, 2018

[18] TheoremDep, "Sum of positive definite matrices is positive definite", https://sharmaeklavya2.github.io/theoremdep/nodes/linear-algebra/matrices/pd-sum-is-pd.html, accessed 22 February 2024

[19] W. Schwarting, J. Alonso-Mora, L.Pauli, S. Karaman, and D. Rus, "Parallel Autonomy in automated vehicles: Safe motion generation with minimal intervention", in 2017 IEEE Intelligent Vehicles Symposium (IV), pp. 812-818, 2017.

[20] L. Ferranti, B. Brito, E. Pool, Y. Zheng, R.M. Ensing, R. Happee, B. Shyrokau, J.F.P. Kooij, J. Alonso-Mora, and D.M. Gavrila, "SafeVRU: A research platform for interaction of self-driving vehicles with vulnerable road users", in 2019 IEEE Intelligent Vehicles Symposium (IV), pp. 1660-1666, 2019.

[21] L. Lyons and L. Ferranti, "Curvature Aware Model Predictive Contouring Control", IEEE International Conference on Robotics And Automation (ICRA), 2023.

[22] A. Domahidi and J. Jerez, "Forces professional, embotech GmbH (http://embotech.com/forces-pro)," July 2014.

[23] J. Hensman, A. Matthews, Z. Ghahramani, "Scalable Variational Gaussian Process Classification", Artificial Intelligence and Statistics. PMLR, 2015, pp. 351-360.

[24] C. E. Rasmussen, "Gaussian Processes for Machine Learning", Max Planck Institute for Biological Cybernetics, May 2006.

APPENDIX

A. Theorem I

This theorem from [16] shows how to derive the conditional distribution of a multivariate normal distribution. Suppose $X = (x_a, x_b)$ is jointly Gaussian distributed with the mean, covariance, and inverse of the covariance defined as follows:

$$\mu = \begin{bmatrix} \mu_a \\ \mu_b \end{bmatrix},$$

$$\Sigma = \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix},$$

$$\Lambda = \Sigma^{-1} = \begin{bmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{bmatrix}.$$

Then the posterior conditional can be derived according to the following:

$$\begin{aligned} p\left(x_a \mid x_b\right) &= \mathcal{N}\left(x_a \mid \mu_{a|b}, \Sigma_{a|b}\right) \\ \mu_{a|b} &= \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}\left(x_b - \mu_b\right) \\ &= \mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}\left(x_b - \mu_b\right) \\ &= \Sigma_{a|b}\left(\Lambda_{aa}\mu_a - \Lambda_{ab}\left(x_b - \mu_b\right)\right) \\ \Sigma_{a|b} &= \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba} \end{aligned}$$

B. Proof I

This proof from [18] shows that the sum of two positive definite matrices is also positive definite. Let us first specify a positive definite matrix: A matrix A is positive definite if, for any non-zero column vector x, the condition $x^T Ax > 0$ holds.

Let A and B be two positive definite matrices.

The matrix $C = A + B$ is positive definite.

By definition, since A and B are positive definite, for any non-zero vector x, we have: $x^T Ax > 0$ $x^T Bx > 0$

Consider the matrix $C = A + B$. We need to show that for any non-zero vector x, $x^T Cx > 0$.

$$x^T Cx = x^T (A + B)x = x^T Ax + x^T Bx$$

Since both $x^T Ax$ and $x^T Bx$ are greater than zero (from step 1), their sum is also greater than zero:

$$x^T Ax + x^T Bx > 0$$

Therefore, for any non-zero vector x, $x^T Cx > 0$, which means $C = A + B$ is positive definite.

Chapter 3

Supporting Information on Methods

This chapter expands on the methodologies employed throughout this thesis. Section 3-1 outlines the conventional framework for training Gaussian Process (GP)s. Following this, Section 3-2 delves into the development of the Multi-Step Gaussian Process (MSGP) training framework, illustrated through comparisons across different concepts using a straightforward 1D example. Finally, Section 3-2-2 offers a comprehensive mathematical explanation of the proposed MSGP learning framework and the code implementation.

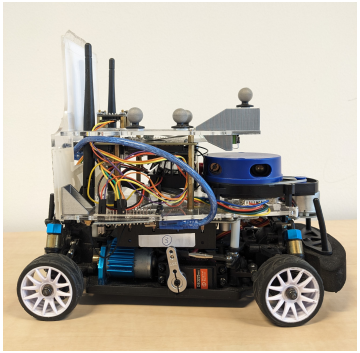


Figure 3-1: The 1:10 scale car-like robot targeted for control.

3-1 Standard Gaussian Process Model Training

This section elaborates on the standard GP training framework for training the three GP models that govern the total vehicle dynamics, which is conducted over multiple epochs. Leveraging Stochastic Variational Gaussian Process (SVGP) to reduce computational load, the algorithm operates over multiple data batches within each epoch.

Consider a selection of N input-output data pairs forming the data set,

$$\begin{aligned}\mathcal{D} = \Big\{ \; & \boldsymbol{a}_x = [a_{x_0}; \ldots; a_{x_{N-1}}] \in \mathbb{R}^N, \\ & \boldsymbol{a}_y = [a_{y_0}; \ldots; a_{y_{N-1}}] \in \mathbb{R}^N, \\ & \boldsymbol{a}_\omega = [a_{\omega_0}; \ldots; a_{\omega_{N-1}}] \in \mathbb{R}^N, \\ & \boldsymbol{X} = \left[\boldsymbol{x}_0^T; \ldots; \boldsymbol{x}_{N-1}^T\right] \in \mathbb{R}^{N \times n_d} \; \Big\} \, ,\end{aligned} \tag{3-1}$$

with input data batch $\boldsymbol{X}$ in which $\boldsymbol{x} = \left[\tau, \delta, v_x, v_y, \omega\right]^T$. And $\boldsymbol{a}_x$, $\boldsymbol{a}_y$ and $\boldsymbol{a}_\omega$ are vectors containing the longitudinal, lateral, and yaw acceleration target values, respectively. Then for each batch iteration within one epoch a mini-batch of data of size $N_b < N$ is initialized,

$$\begin{aligned}\mathcal{D}_{batch} = \Big\{ \; & \boldsymbol{a}_x = \boldsymbol{a}_{x_{0:(N_b-1)}}, \\ & \boldsymbol{a}_y = \boldsymbol{a}_{y_{0:(N_b-1)}}, \\ & \boldsymbol{a}_\omega = \boldsymbol{a}_{\omega_{0:(N_b-1)}}, \\ & \boldsymbol{X} = \left[\boldsymbol{\tau}_{0:(N_b-1)}, \boldsymbol{\delta}_{0:(N_b-1)}, \boldsymbol{v}_{x_{0:(N_b-1)}}, \boldsymbol{v}_{y_{0:(N_b-1)}}, \boldsymbol{\omega}_{0:(N_b-1)}\right]_{N_b \times n_d} \; \Big\} \, ,\end{aligned} \tag{3-2}$$

with $\boldsymbol{a}_x \in \mathbb{R}^{N_b}$ the mini-batch of longitudinal acceleration target values, $\boldsymbol{a}_y \in \mathbb{R}^{N_b}$ the mini-batch of lateral acceleration target values, $\boldsymbol{a}_\omega \in \mathbb{R}^{N_b}$ the mini-batch of angular acceleration target values, and $\boldsymbol{X} \in \mathbb{R}^{N_b \times n_d}$ the mini-batch of input training data.

Followingly, the algorithm uses the mini-batch of input training data to evaluate the GP models, from which it derives the mean and covariance matrix for each acceleration dimension. The covariance is then used in the optimization function $\mathcal{L}$ (the loss function defined as the log marginal likelihood), to optimize the kernel parameters and inducing points and labels.

For the longitudinal acceleration,

$$\begin{aligned}& \boldsymbol{a}_x = \mathcal{N}\left(\overline{\boldsymbol{a}}_x, \boldsymbol{K}^{a_x}\right) \text{ with} \\ & \overline{\boldsymbol{a}}_x = \begin{bmatrix} \overline{a}_{x_0} \\ \overline{a}_{x_1} \\ \vdots \\ \overline{a}_{x_{N_b-1}} \end{bmatrix} \in \mathbb{R}^{N_b}, \quad \boldsymbol{K}^{a_x} = \begin{bmatrix} \sigma^{a_x}_{0,0} & \sigma^{a_x}_{0,1} & \cdots & \sigma^{a_x}_{0,(N_b-1)} \\ \sigma^{a_x}_{1,0} & \sigma^{a_x}_{1,1} & \cdots & \sigma^{a_x}_{1,(N_b-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{a_x}_{(N_b-1),0} & \sigma^{a_x}_{(N_b-1),1} & \cdots & \sigma^{a_x}_{(N_b-1),(N_b-1)} \end{bmatrix} \in \mathbb{R}^{N_b \times N_b} \, ,\end{aligned} \tag{3-3}$$

$$\boldsymbol{\theta}_x^+ \leftarrow \text{argmin}_{\boldsymbol{\theta}} \; \mathcal{L}(\boldsymbol{a}_x, \boldsymbol{K}^{a_x}, \boldsymbol{\theta}_x) \tag{3-4}$$

For the lateral acceleration,

$$\begin{aligned}& \boldsymbol{a}_y = \mathcal{N}\left(\overline{\boldsymbol{a}}_y, \boldsymbol{K}^{a_y}\right) \text{ with} \\ & \overline{\boldsymbol{a}}_y = \begin{bmatrix} \overline{a}_{y_0} \\ \overline{a}_{y_1} \\ \vdots \\ \overline{a}_{y_{N_b-1}} \end{bmatrix} \in \mathbb{R}^{N_b}, \quad \boldsymbol{K}^{a_y} = \begin{bmatrix} \sigma^{a_y}_{0,0} & \sigma^{a_y}_{0,1} & \cdots & \sigma^{a_y}_{0,(N_b-1)} \\ \sigma^{a_y}_{1,0} & \sigma^{a_y}_{1,1} & \cdots & \sigma^{a_y}_{1,(N_b-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{a_y}_{(N_b-1),0} & \sigma^{a_y}_{(N_b-1),1} & \cdots & \sigma^{a_y}_{(N_b-1),(N_b-1)} \end{bmatrix} \in \mathbb{R}^{N_b \times N_b} \, ,\end{aligned} \tag{3-5}$$

$$\boldsymbol{\theta}_y^+ \leftarrow \text{argmin}_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{a}_y, \boldsymbol{K}^{a_y}, \boldsymbol{\theta}_y) \tag{3-6}$$

For the angular acceleration,

$$\boldsymbol{a}_\omega = \mathcal{N}\left(\overline{\boldsymbol{a}}_\omega, \boldsymbol{K}^{a_\omega}\right) \text{ with}$$
$$\overline{\boldsymbol{a}}_\omega = \begin{bmatrix} \overline{a}_{\omega_0} \\ \overline{a}_{x_1} \\ \vdots \\ \overline{a}_{\omega_{N_b-1}} \end{bmatrix} \in \mathbb{R}^{N_b}, \quad \boldsymbol{K}^{a_\omega} = \begin{bmatrix} \sigma^{a_\omega}_{0,0} & \sigma^{a_\omega}_{0,1} & \dots & \sigma^{a_\omega}_{0,(N_b-1)} \\ \sigma^{a_\omega}_{1,0} & \sigma^{a_\omega}_{1,1} & \dots & \sigma^{a_\omega}_{1,(N_b-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{a_\omega}_{(N_b-1),0} & \sigma^{a_\omega}_{(N_b-1),1} & \dots & \sigma^{a_\omega}_{(N_b-1),(N_b-1)} \end{bmatrix} \in \mathbb{R}^{N_b \times N_b} , \tag{3-7}$$

$$\boldsymbol{\theta}_\omega^+ \leftarrow \text{argmin}_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{a}_\omega, \boldsymbol{K}^{a_\omega}, \boldsymbol{\theta}_\omega) \tag{3-8}$$

Equations (3-2)-(3-8) are executed for data batches. Upon processing all mini-batches of input data, the algorithm progresses to the subsequent epoch, repeating the whole procedure again, etc.

3-2 Multi-Step Gaussian Process Model Training

Before applying the multi-step training framework to the full model, the hypothesis was initially tested on a simple one-dimensional system with one input and one output variable. This initial test was conducted to understand the effects and potential benefits of employing a multi-step training approach. This section explores how adjusting the training methodology could enhance the predictive capabilities of Gaussian Processes (GPs). For clarity, a GP that is trained under a multi-step training framework was referred to as a MSGP.

The evaluation of the method focused on several key metrics: accuracy of long-term predictions, variance reduction, training duration, and robustness. Additionally, the changes in the internal model, specifically the kernel function, were analyzed under different training frameworks. By testing the proposed method on a simple example first, the study aimed to identify the enhancements brought about by the multi-step training framework.

3-2-1 A one-dimensional Gaussian Process model

The simple dynamic system for testing the different training strategies is the robot from Fig. 3-1 rolling out from an initial velocity to a standstill due to friction. Its velocity is described by

$$v(k+1) = v(k) + \Delta v(k) \ , \tag{3-9}$$

where $\Delta v(k)$ is the decrease in velocity at each time interval, $v(k)$ is the velocity at the current time step and $v(k+1)$ is the velocity at the next time step.

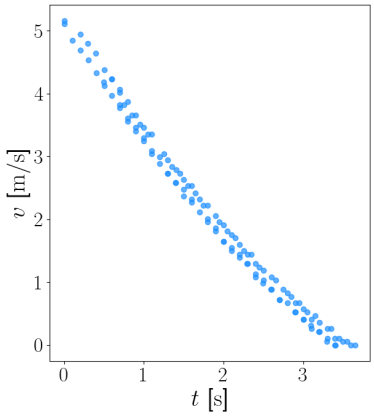


Figure 3-2: Experimental data illustrating the nonlinear velocity decay of a 1:10 scale car-like robot when it is decelerating from an initial velocity to a stand-still, showcasing faster velocity reduction in the initial stages followed by a slower decrease towards the end.

Data preparation

The goal was to learn a GP model capable of predicting the decrease in velocity for a given velocity. Data from the robot decelerating from various initial velocities to a standstill solely through rolling out due to friction (i.e. without the use of brakes), was used for training (Fig. 3-2). To generate the target values $\mathbf{y}$, representing velocity decrements for each time step, the original velocity vector was shifted forward by one step, and then subtracting the original velocity vector from the shifted version. The resulting vector, which represents the velocity decrement for each time step, along with the original velocity data vector, together form the dataset $\mathcal{D}$

$$\begin{aligned} \mathcal{D} = \{ \ \mathbf{y} = [\Delta v_0; \ldots; \Delta v_{N-1}] \in \mathbb{R}^N, \\ \boldsymbol{X} = [v_0; \ldots; v_{N-1}] \in \mathbb{R}^N \} \end{aligned} \quad , \tag{3-10}$$

in which $\boldsymbol{X}$ denotes the input data and $\mathbf{y}$ the target values. Note that $\boldsymbol{X}$ is not a matrix but a vector of input training data, but a capital letter is used to remain consistent with the rest of the report. The dataset was used to train a GP that models the relationship between velocity and velocity change $f : v \to \Delta v$ with input velocity $v \in \mathbb{R}$ and target values velocity change $\Delta v \in \mathbb{R}$:

$$\Delta \boldsymbol{v} = \mathcal{N}(m(\boldsymbol{X}), \boldsymbol{\Sigma}(\boldsymbol{X})) \ . \tag{3-11}$$

This GP model serves as a baseline comparison for the proposed MSGP training framework.

To assess the GP model's predictive capability over future time steps later on, vectors for two-step $\Delta \boldsymbol{v}_2$ and ten-step ahead velocity decrements $\Delta \boldsymbol{v}_{10}$ were constructed. This involved shifting the initial velocity data forward by two and ten steps and subtracting the original velocity data from it to create velocity decrement vectors. Due to shifting the velocity vectors forward, the empty vector elements at the end are filled with zeros, denoting that the robot car has come to a standstill. Note that the final values of the velocity decrement vectors, especially in the ten-step ahead case ($\Delta \boldsymbol{v}_{10}$), often reach zero prematurely. This leads to a large part of the $\Delta \boldsymbol{v}_{10}$ vector being filled with zeros, reflecting that no further velocity reduction occurs once the robot car stops. Therefore, this limitation prevents the model from

providing meaningful ten-step velocity decrements for low initial velocities. As a result, the data visualizations in Fig. 3-3 and 3-4 only include mappings from initial velocities of 1 m/s or greater.

Sequential GP evaluation for future velocity prediction

To make future predictions using the trained GP model, the GP is evaluated at input velocities $\boldsymbol{X}_0 = [v_0, v_1, \ldots, v_{N-1}]^T$ to obtain the change in velocity at time step k: $\Delta\boldsymbol{v}_0 = [\Delta v_0, \Delta v_1, \ldots, \Delta v_{N-1}]^T$. These velocity changes are then added to the current velocities $\boldsymbol{v}_0$ to obtain the velocities at the next time step $\boldsymbol{X}_1 = [v_1, v_2, \ldots, v_N]^T$. This process enables subsequent predictions of velocity decreases and can be repeated until predictions for velocities up to a desired number of steps ahead are made:

$$\begin{aligned} \boldsymbol{v}_{k+1} &= \boldsymbol{v}_k + \Delta\boldsymbol{v}_k \\ \begin{bmatrix} v_{k+1} \\ \vdots \\ v_{N+k} \end{bmatrix} &= \begin{bmatrix} v_k \\ \vdots \\ v_{N+k-1} \end{bmatrix} + \begin{bmatrix} \Delta v_k \\ \vdots \\ \Delta v_{N+k-1} \end{bmatrix} \quad \forall k \in [0, \ldots, k_{end}) \end{aligned} \tag{3-12}$$

After preparing the data and defining how the trained GP models can be used to make sequential future predictions of future velocities, the next step is examining how different training methodologies can affect the predictive capabilities of the GP models.

Standard GP model training

To obtain a baseline GP model, it is trained using the standard framework (which is the framework as described in Section 3-1 but only for one input feature and one output dimension). The training algorithm uses the input training data to evaluate the GP at these data points, from which it derives the mean and covariance matrix:

$$\Delta\boldsymbol{v} = \mathcal{N}\left(\overline{\Delta\boldsymbol{v}}, \boldsymbol{K}\right) \ . \tag{3-13}$$

The covariance is then used in the optimization function $\mathcal{L}$ (log marginal likelihood) to optimize the kernel parameters and inducing points and labels, which are represented by parameter vector $\boldsymbol{\theta}$

$$\boldsymbol{\theta}^+ \leftarrow \text{argmin}_{\boldsymbol{\theta}} \ \mathcal{L}(\Delta\boldsymbol{v}, \boldsymbol{K}, \boldsymbol{\theta}) \ . \tag{3-14}$$

This procedure is repeated across a predetermined number of training cycles until the model adequately fits the data.

GP models trained using the standard framework are typically optimized for predicting short-term dynamics. This is evidenced in Fig. 3-3, which demonstrates that while the GP model accurately predicts immediate velocity reductions (Δv_1 and Δv_2), it tends to overestimate decreases over longer time steps (Δv_{10}). Additionally, the overestimation is visually depicted by the predicted mean velocities (gray line) ending before the actual data (blue line), indicating a more rapid velocity reduction than what occurs in reality. The experimental data (Fig. 3-2) show that the robot car's velocity decreases in a slightly nonlinear manner, characterized by a quicker initial decrease and a slower reduction towards the end. This pattern often

results from nonlinear factors such as uneven ground contact, air resistance, and nonlinear friction between the gears and electric motor, which have a more pronounced effect at higher speeds. The tendency of the GP to overpredict velocity decrease can be traced back to its training approach in which it has only learned to map current velocities to immediate velocity changes, thereby not learning the diminished rate of decrease at further (here: lower) velocities. Consequently, using a standard GP model to predict long-term future dynamics results in inaccurate long-term predictions. Using a standard GP model for making predictions of long-term future dynamics will therefore lead to erroneous long-term predictions, as evident from Fig. 3-3. This limitation highlights the challenges of using standard GP models for long-term predictions in applications such as Model Predictive Control (MPC), which depend on accurate predictions of future dynamics. Such challenges emphasize the need for developing new training methods designed to improve the GP's multi-step predictive accuracy and meet the MPC demands more effectively.

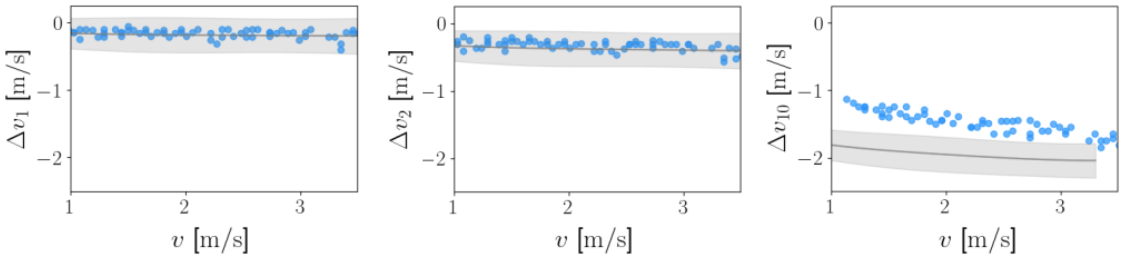


Figure 3-3: Predicted velocity reductions over **1** time step Δv_1 (first column), **2** time steps Δv_2 (second column), and **10** time steps Δv_{10} (third column), predicted by the GP model trained under the standard training framework. While short-term forecasts align closely with the actual data, far future predictions display overconfident velocity reductions.

Multi-Step GP model training

The multi-step ahead training method improves on the standard one-step ahead training framework by using the GP to predict multiple future steps, effectively performing "in-training" simulations of future velocities. This approach involves using the GP to generate simulated future velocities based on its currently learned predictive capabilities. These predictions are then compared with the actual future observations based on which adjustments are made to the kernel hyperparameters and inducing points. This iterative process is carried out across epochs, with each set of simulated future predictions based on the increasingly refined model from previous iterations until an adequate model fit has been reached. The outline of this training strategy is as follows. In each training cycle, the future velocity is simulated by advancing the mean velocity k_{pred} steps (here $k_{pred} = 2$) through sequential "imaginary" GP evaluations. The training algorithm uses the input training data to evaluate the GP at these data points, from which it derives the mean and covariance matrix:

$$\Delta \boldsymbol{v}_k = \mathcal{N}\left(\overline{\Delta \boldsymbol{v}_k}, \boldsymbol{K}_k\right) . \tag{3-15}$$

By summing the predicted mean velocity decrements to the current velocity, the velocity at the next time step can be obtained as

$$\hat{\boldsymbol{v}}_{k+1} = \hat{\boldsymbol{v}}_k + \overline{\Delta \boldsymbol{v}_k} \quad \forall k \in \{0, \ldots, k_{pred} - 1\} . \tag{3-16}$$

This process of sequentially evaluating the Gaussian Process (GP) and updating velocity continues until the simulated velocity reduction reaches the predetermined horizon, denoted as $k = k_{pred}$. Concurrently, the covariance matrix is also advanced. Rather than using the covariance matrix directly as predicted by the GP at step k, it is revised to incorporate prior uncertainties. To keep computations efficient and to ensure the covariance matrices remain positive definite — a prerequisite for the solver — it is assumed that the matrices from consecutive steps $k-1$ and k are independent, such that they can be summed up:

$$\boldsymbol{K}_k = \boldsymbol{K}(\boldsymbol{X}_k, \boldsymbol{X}_k) \leftarrow \boldsymbol{K}(\boldsymbol{X}_{k-1}, \boldsymbol{X}_{k-1}) + \boldsymbol{K}(\boldsymbol{X}_k, \boldsymbol{X}_k) \;\; \forall k \in \{0, \dots, k_{pred} - 1\} \; . \tag{3-17}$$

Given the short forward simulation horizon ($k_{pred} = 2$), this simplification is expected to introduce only a minor error. Moreover, this method is guaranteed to function, as it is supported by the properties of matrix operations. Specifically, the sum of two symmetric positive definite matrices always yields another positive definite matrix. This result holds as the squared exponential kernel function (used here to construct the covariance matrix), always produces positive definite matrices [5]. This updated covariance is then used in the optimization function $\mathcal{L}$ (log marginal likelihood), along with the forward shifted target values $[\Delta \boldsymbol{v}]_{\setminus 0:k_{pred}}$ to optimize the kernel parameters and inducing points and labels, which are represented by parameter vector $\boldsymbol{\theta}$

$$\boldsymbol{\theta}^+ \leftarrow \text{argmin}_{\boldsymbol{\theta}} \;\; \mathcal{L}([\Delta \boldsymbol{v}]_{\setminus 0:k_{pred}}, \boldsymbol{K}_k, \boldsymbol{\theta}) \; . \tag{3-18}$$

Fig. 3-4 shows that the MSGP model trained under the proposed framework can effectively predict both near-term (Δv_1 and Δv_2) and long-term velocity reductions (Δv_{10}). Table 3-1 further highlights the effectiveness of this newly trained model, which achieves a 44% decrease in RMSE and a 50% decrease in MAE, along with a 29% reduction in variance. These improvements can be attributed to key adjustments in the internal model: the kernel function. An increase in the kernel lengthscale from 1.29 to 1.66 indicates a stronger correlation among more distant points, contributing to a smoother predictive function. Moreover, the output scales reduced from 0.32 to 0.25, suggesting a decreased reach of the function, making its sensitivity to inputs more mild. This recalibration helps prevent overestimations and enhances the accuracy of predictions over long horizons.

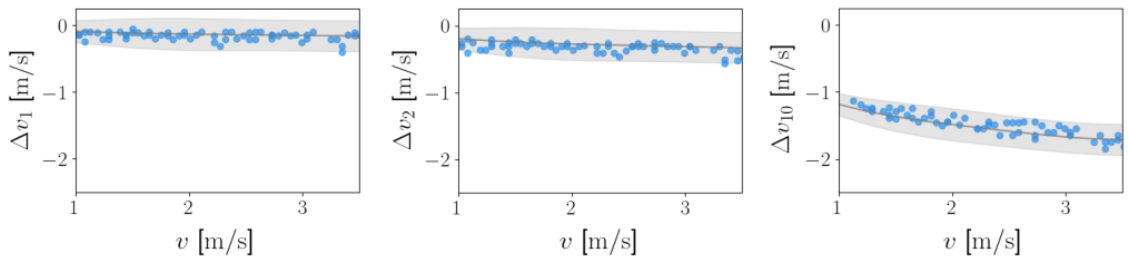


Figure 3-4: Predicted velocity reductions over **1** time step Δv_1 (first column), **2** time steps Δv_2 (second column), and **10** time steps Δv_{10} (third column), predicted by the MSGP model. Both short-term and far-future predictions now align closely with the actual data.

It is observed that the dynamics have stabilized over the long term, and the adjustments to the kernel parameters have rendered the model less responsive. This change can be linked to the new behavior observed in the near-term dynamics. Specifically, an analysis of the velocity

Table 3-1: Comparison of various MSGP training framework benchmarked against the standard GP training framework (baseline). Metrics include Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Average variance (Avg Var) (and their relative differences w.r.t the baseline method), computation time, and the positive definiteness of the covariance matrix.

Metrics	**GP model**	**MSGP model**
RMSE [m/s]	0.4214	0.2372
MAE [m/s]	0.3946	0.1969
Avg Var $[(\mathrm{m/s})^2]$	0.0151	0.0107
$\Delta\%$ RMSE* [%]	-	-44
$\Delta\%$ MAE* [%]	-	-50
$\Delta\%$ Avg Var* [%]	-	-29
kernel length scale ℓ	1.29	1.66
kernel output scale σ_f	0.32	0.25
Training time [s]	0.17	0.30
Positive definiteness $\boldsymbol{K}$	guaranteed	guaranteed

*The percentual change $\Delta\%$ is computed w.r.t. the standard GP training method (baseline).

reduction over one time step Δv_1 in Fig. 3-5 shows that the retuned MSGP model adopts a slightly more conservative approach in its near-term predictions compared to the previously overly sensitive GP model, thereby maintaining stability over the long term.

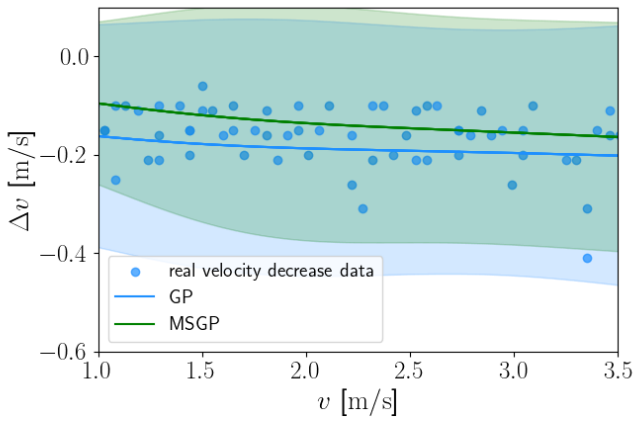


Figure 3-5: A close-up comparison of predicted and actual velocity decreases over 1 time step Δv_1. The blue line and shaded area represent the mean and uncertainty of predictions from the GP trained under the standard framework. The green line and green shaded area show predictions from the MSGP model. The blue dots denote the actual velocity reduction over one time step.

3-2-2 Multi-Step Gaussian Process Model Training Of The Full Dynamics

Section 3-2-1 illustrated using a simple 1D example how standard GP training encountered issues with long-term prediction accuracy. These issues were effectively mitigated by adopting a multi-step ahead training strategy, which involves implicitly tuning the kernel hyperparameters to accommodate for long-term dynamics, as hypothesized. The approach is now extended to model the complete vehicle dynamics, utilizing the multi-step GP framework to refine the GP acceleration models for the robot. This section provides a detailed mathematical description of the proposed MSGP training framework for the three GP models that govern the full robot dynamics.

Leveraging SVGP to reduce computational load, the algorithm operates over various data batches within each epoch. Consider a selection of N input-output data pairs forming the data set,

$$\begin{aligned} \mathcal{D} = \Big\{ \; & \boldsymbol{a}_x = [a_{x_0}; \dots; a_{x_{N-1}}] \in \mathbb{R}^N, \\ & \boldsymbol{a}_y = [a_{y_0}; \dots; a_{y_{N-1}}] \in \mathbb{R}^N, \\ & \boldsymbol{a}_\omega = [a_{\omega_0}; \dots; a_{\omega_{N-1}}] \in \mathbb{R}^N, \\ & \boldsymbol{X} = \left[\boldsymbol{x}_0^T; \dots; \boldsymbol{x}_{N-1}^T\right] \in \mathbb{R}^{N \times n_d} \; \Big\} \; , \end{aligned} \tag{3-19}$$

with input data batch $\boldsymbol{X}$ in which $\boldsymbol{x} = \left[\tau, \delta, v_x, v_y, \omega\right]^T$. And $\boldsymbol{a}_x$, $\boldsymbol{a}_y$ and $\boldsymbol{a}_\omega$ are vectors containing the longitudinal, lateral, and yaw acceleration target values, respectively. Then for each batch iteration within one epoch a mini-batch of data of size $N_b < N$ is initialized,

$$\begin{aligned} \mathcal{D}_{batch} = \Big\{ \; & \boldsymbol{a}_{x_k} = \boldsymbol{a}_{x_{0:(N_b-1)}}, \\ & \boldsymbol{a}_{y_k} = \boldsymbol{a}_{y_{0:(N_b-1)}}, \\ & \boldsymbol{a}_{\omega_k} = \boldsymbol{a}_{\omega_{0:(N_b-1)}}, \\ & \boldsymbol{X}_k = \left[\boldsymbol{\tau}_{0:(N_b-1)}, \boldsymbol{\delta}_{0:(N_b-1)}, \boldsymbol{v}_{x_{0:(N_b-1)}}, \boldsymbol{v}_{y_{0:(N_b-1)}}, \boldsymbol{\omega}_{0:(N_b-1)}\right]_{N_b \times n_d} \; \Big\} \; , \end{aligned} \tag{3-20}$$

with $\boldsymbol{a}_{x_k} \in \mathbb{R}^{N_b}$ are the longitudinal acceleration target values, $\boldsymbol{a}_{y_k} \in \mathbb{R}^{N_b}$ the lateral acceleration target values, $\boldsymbol{a}_{\omega_k} \in \mathbb{R}^{N_b}$ the angular acceleration target values, and $\boldsymbol{X}_k \in \mathbb{R}^{N_b \times n_d}$ the input training data.

Followingly, the previous covariance matrix is initialized as a zero matrix,

$$\begin{aligned} \boldsymbol{K}_{k-1}^{a_x} &= \boldsymbol{0} \in \mathbb{R}^{N_b \times N_b} \\ \boldsymbol{K}_{k-1}^{a_y} &= \boldsymbol{0} \in \mathbb{R}^{N_b \times N_b} \\ \boldsymbol{K}_{k-1}^{a_\omega} &= \boldsymbol{0} \in \mathbb{R}^{N_b \times N_b} \; . \end{aligned} \tag{3-21}$$

And the velocity estimates are initialized as the velocities given in the training input matrix $\boldsymbol{X}_k$,

$$\begin{aligned} \hat{\boldsymbol{v}}_{x_k} &= [\boldsymbol{X}_k]_{\cdot,2} \in \mathbb{R}^{N_b} \\ \hat{\boldsymbol{v}}_{y_k} &= [\boldsymbol{X}_k]_{\cdot,3} \in \mathbb{R}^{N_b} \\ \hat{\boldsymbol{\omega}}_k &= [\boldsymbol{X}_k]_{\cdot,4} \in \mathbb{R}^{N_b} \; . \end{aligned} \tag{3-22}$$

When the velocity input data vectors and covariance matrix have been initialized, the in-training simulation can start. Here we iterate through the equations (3-23) till (3-34) for k from 0 up to a desired future steps ahead k_{pred}. The mini-batch of input training data is used to evaluate the GP models, from which it derives the main and covariance matrix for each dimension of acceleration. For the longitudinal acceleration,

$$\boldsymbol{a}_{x_k} = \mathcal{N}\left(\overline{\boldsymbol{a}}_{x_k}, \boldsymbol{K}_k^{a_x}\right) \text{ with}$$

$$\overline{\boldsymbol{a}}_{x_k} = \begin{bmatrix} \overline{a}_{x_k} \\ \overline{a}_{x_{k+1}} \\ \vdots \\ \overline{a}_{x_{N_b-1}} \end{bmatrix} \in \mathbb{R}^{N_b-k}, \quad \boldsymbol{K}_k^{a_x} = \begin{bmatrix} \sigma^{a_x}_{k,k} & \sigma^{a_x}_{k,k+1} & \cdots & \sigma^{a_x}_{k,(N_b-1)} \\ \sigma^{a_x}_{k+1,k} & \sigma^{a_x}_{k+1,k+1} & \cdots & \sigma^{a_x}_{k+1,(N_b-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{a_y}_{(N_b-1),k} & \sigma^{a_y}_{(N_b-1),k+1} & \cdots & \sigma^{a_y}_{(N_b-1),(N_b-1)} \end{bmatrix}\Bigg|_k \in \mathbb{R}^{(N_b-k)\times(N_b-k)} \tag{3-23}$$

For the lateral acceleration,

$$\boldsymbol{a}_{y_k} = \mathcal{N}\left(\overline{\boldsymbol{a}}_{y_k}, \boldsymbol{K}_k^{a_y}\right) \text{ with}$$

$$\overline{\boldsymbol{a}}_{y_k} = \begin{bmatrix} \overline{a}_{y_k} \\ \overline{a}_{y_{k+1}} \\ \vdots \\ \overline{a}_{y_{N_b-1}} \end{bmatrix} \in \mathbb{R}^{N_b-k}, \quad \boldsymbol{K}_k^{a_y} = \begin{bmatrix} \sigma^{a_y}_{k,k} & \sigma^{a_y}_{k,k+1} & \cdots & \sigma^{a_y}_{k,(N_b-1)} \\ \sigma^{a_y}_{k+1,k} & \sigma^{a_y}_{k+1,k+1} & \cdots & \sigma^{a_y}_{k+1,(N_b-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{a_y}_{(N_b-1),k} & \sigma^{a_y}_{(N_b-1),k+1} & \cdots & \sigma^{a_y}_{(N_b-1),(N_b-1)} \end{bmatrix}\Bigg|_k \in \mathbb{R}^{(N_b-k)\times(N_b-k)} \tag{3-24}$$

For the angular acceleration,

$$\boldsymbol{a}_{\omega_k} = \mathcal{N}\left(\overline{\boldsymbol{a}}_{\omega_k}, \boldsymbol{K}_k^{a_\omega}\right) \text{ with}$$

$$\overline{\boldsymbol{a}}_{\omega_k} = \begin{bmatrix} \overline{a}_{x_k} \\ \overline{a}_{\omega_{k+1}} \\ \vdots \\ \overline{a}_{\omega_{N_b-1}} \end{bmatrix} \in \mathbb{R}^{N_b-k}, \quad \boldsymbol{K}_k^{a_\omega} = \begin{bmatrix} \sigma^{a_\omega}_{k,k} & \sigma^{a_\omega}_{k,k+1} & \cdots & \sigma^{a_\omega}_{k,(N_b-1)} \\ \sigma^{a_\omega}_{k+1,k} & \sigma^{a_\omega}_{k+1,k+1} & \cdots & \sigma^{a_\omega}_{k+1,(N_b-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{a_\omega}_{(N_b-1),k} & \sigma^{a_\omega}_{(N_b-1),k+1} & \cdots & \sigma^{a_\omega}_{(N_b-1),(N_b-1)} \end{bmatrix}\Bigg|_k \in \mathbb{R}^{(N_b-k)\times(N_b-k)} \tag{3-25}$$

Followingly, the covariance matrices for all acceleration dimensions are compounded by summation. If k is equal to zero, the covariance matrices will be updated according to (3-26).

$$\begin{aligned} \boldsymbol{K}_k^{a_x} \in \mathbb{R}^{N_b\times N_b} &\leftarrow \boldsymbol{K}_{k-1}^{a_x} + \boldsymbol{K}_k^{a_x} \\ \boldsymbol{K}_k^{a_y} \in \mathbb{R}^{N_b\times N_b} &\leftarrow \boldsymbol{K}_{k-1}^{a_y} + \boldsymbol{K}_k^{a_y} \\ \boldsymbol{K}_k^{a_\omega} \in \mathbb{R}^{N_b\times N_b} &\leftarrow \boldsymbol{K}_{k-1}^{a_\omega} + \boldsymbol{K}_k^{a_\omega} \end{aligned} \tag{3-26}$$

If k is larger than 0, the covariance matrix for each GP model is updated according to (3-27). To maintain correct dimensionality with the training input data matrix $\boldsymbol{X}_k$, the first row and column of the updated covariance are discarded. The covariance matrix update for the

longitudinal acceleration,

$$\boldsymbol{K}_k^{a_x} \in \mathbb{R}^{(N_b-k)\times(N_b-k)} \leftarrow \left[\boldsymbol{K}_{k-1}^{a_x}\right]_{\backslash 0,\backslash 0} + \boldsymbol{K}_k^{a_x} =$$
$$\left[\begin{matrix} \sigma^{a_x}_{k+1,k+1} & \sigma^{a_x}_{k+1,k+2} & \cdots & \sigma^{a_x}_{k+1,(N_b-1)} \\ \sigma^{a_x}_{k+2,k+1} & \sigma^{a_x}_{k+2,k+2} & \cdots & \sigma^{a_x}_{k+2,(N_b-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{a_x}_{(N_b-1),k+1} & \sigma^{a_x}_{(N_b-1),k+2} & \cdots & \sigma^{a_x}_{(N_b-1),(N_b-1)} \end{matrix}\right]\Bigg|_{k-1} + \left[\begin{matrix} \sigma^{a_x}_{k+1,k+1} & \sigma^{a_x}_{k+1,k+2} & \cdots & \sigma^{a_x}_{k+1,(N_b-1)} \\ \sigma^{a_x}_{k+2,k+1} & \sigma^{a_x}_{k+2,k+2} & \cdots & \sigma^{a_x}_{k+1,(N_b-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{a_x}_{(N_b-1),k+1} & \sigma^{a_x}_{(N_b-1),k+2} & \cdots & \sigma^{a_x}_{(N_b-1),(N_b-1)} \end{matrix}\right]\Bigg|_{k} \tag{3-27}$$

The covariance matrix update for the lateral acceleration,

$$\boldsymbol{K}^{a_y} \in \mathbb{R}^{(N_b-k)\times(N_b-k)} \leftarrow \left[\boldsymbol{K}_{k-1}^{a_y}\right]_{\backslash 0,\backslash 0} + \boldsymbol{K}_k^{a_y} =$$
$$\left[\begin{matrix} \sigma^{a_y}_{k+1,k+1} & \sigma^{a_y}_{k+1,k+2} & \cdots & \sigma^{a_y}_{k+1,(N_b-1)} \\ \sigma^{a_y}_{k+2,k+1} & \sigma^{a_y}_{k+2,k+2} & \cdots & \sigma^{a_y}_{k+2,(N_b-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{a_y}_{(N_b-1),k+1} & \sigma^{a_y}_{(N_b-1),k+2} & \cdots & \sigma^{a_y}_{(N_b-1),(N_b-1)} \end{matrix}\right]\Bigg|_{k-1} + \left[\begin{matrix} \sigma^{a_y}_{k+1,k+1} & \sigma^{a_y}_{k+1,k+2} & \cdots & \sigma^{a_y}_{k+1,(N_b-1)} \\ \sigma^{a_y}_{k+2,k+1} & \sigma^{a_y}_{k+2,k+2} & \cdots & \sigma^{a_y}_{k+1,(N_b-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{a_y}_{(N_b-1),k+1} & \sigma^{a_y}_{(N_b-1),k+2} & \cdots & \sigma^{a_y}_{(N_b-1),(N_b-1)} \end{matrix}\right]\Bigg|_{k} \tag{3-28}$$

The covariance matrix update for the angular acceleration,

$$\boldsymbol{K}_k^{a_\omega} \in \mathbb{R}^{(N_b-k)\times(N_b-k)} \leftarrow \left[\boldsymbol{K}_{k-1}^{a_\omega}\right]_{\backslash 0,\backslash 0} + \boldsymbol{K}_k^{a_\omega} =$$
$$\left[\begin{matrix} \sigma^{a_\omega}_{k+1,k+1} & \sigma^{a_\omega}_{k+1,k+2} & \cdots & \sigma^{a_\omega}_{k+1,(N_b-1)} \\ \sigma^{a_\omega}_{k+2,k+1} & \sigma^{a_\omega}_{k+2,k+2} & \cdots & \sigma^{a_\omega}_{k+2,(N_b-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{a_\omega}_{(N_b-1),k+1} & \sigma^{a_\omega}_{(N_b-1),k+2} & \cdots & \sigma^{a_\omega}_{(N_b-1),(N_b-1)} \end{matrix}\right]\Bigg|_{k-1} + \left[\begin{matrix} \sigma^{a_\omega}_{k+1,k+1} & \sigma^{a_\omega}_{k+1,k+2} & \cdots & \sigma^{a_\omega}_{k+1,(N_b-1)} \\ \sigma^{a_\omega}_{k+2,k+1} & \sigma^{a_\omega}_{k+2,k+2} & \cdots & \sigma^{a_\omega}_{k+1,(N_b-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma^{a_\omega}_{(N_b-1),k+1} & \sigma^{a_\omega}_{(N_b-1),k+2} & \cdots & \sigma^{a_\omega}_{(N_b-1),(N_b-1)} \end{matrix}\right]\Bigg|_{k} \tag{3-29}$$

Using the mean accelerations, the one-step ahead velocity states $\hat{\boldsymbol{v}}_{k+1}$ can be predicted by multiplying the predicted mean acceleration with time step Δt and adding it to the current velocity,

$$\begin{aligned} \hat{\boldsymbol{v}}_{x_{k+1}} &= \hat{\boldsymbol{v}}_{x_k} + \overline{\boldsymbol{a}}_{x_k} \cdot \Delta t \\ \begin{bmatrix} \hat{v}_{x_{k+1}} \\ \vdots \\ \hat{v}_{x_{N_b}} \end{bmatrix} &= \begin{bmatrix} \hat{v}_{x_k} \\ \vdots \\ \hat{v}_{x_{N_b-1}} \end{bmatrix} + \begin{bmatrix} \overline{a}_{x_k} \\ \vdots \\ \overline{a}_{x_{N_b-1}} \end{bmatrix} \cdot \Delta t \quad \in \mathbb{R}^{N_b-k} \end{aligned} \tag{3-30}$$

$$\begin{aligned} \hat{\boldsymbol{v}}_{y_{k+1}} &= \hat{\boldsymbol{v}}_{y_k} + \overline{\boldsymbol{a}}_{y_k} \cdot \Delta t \\ \begin{bmatrix} \hat{v}_{y_{k+1}} \\ \vdots \\ \hat{v}_{y_{N_b}} \end{bmatrix} &= \begin{bmatrix} \hat{v}_{y_k} \\ \vdots \\ \hat{v}_{y_{N_b-1}} \end{bmatrix} + \begin{bmatrix} \overline{a}_{y_k} \\ \vdots \\ \overline{a}_{y_{N_b-1}} \end{bmatrix} \cdot \Delta t \quad \in \mathbb{R}^{N_b-k} \end{aligned} \tag{3-31}$$

$$\begin{aligned} \hat{\boldsymbol{\omega}}_{k+1} &= \hat{\boldsymbol{\omega}}_{k} + \overline{\boldsymbol{a}}_{\omega_k} \cdot \Delta t \\ \begin{bmatrix} \hat{\omega}_{k+1} \\ \vdots \\ \hat{\omega}_{N_b} \end{bmatrix} &= \begin{bmatrix} \hat{\omega}_{k} \\ \vdots \\ \hat{\omega}_{N_b-1} \end{bmatrix} + \begin{bmatrix} \overline{a}_{\omega_k} \\ \vdots \\ \overline{a}_{\omega_{N_b-1}} \end{bmatrix} \cdot \Delta t \quad \in \mathbb{R}^{N_b-k} \end{aligned} \tag{3-32}$$

The one-step-ahead simulated velocity states partially form the new inputs for the next GP evaluation. The throttle $\boldsymbol{\tau}$ and steering $\boldsymbol{\delta}$ input vectors are also advanced from k to $k+1$ by discarding their first values to align them with the updated velocity states. Because the control input vectors are now reduced by one element and we lack control input data for indices beyond the existing data set or data batch size, it is necessary to omit the last elements of the simulated velocities to ensure that the dimensions of the input data matrix remain consistent.

$$\begin{aligned} \boldsymbol{\tau}_{k+1} &\leftarrow \left[\boldsymbol{\tau}_{k+1}\right]_{\setminus 0} && \in \mathbb{R}^{N_b-1-k} \\ \boldsymbol{\delta}_{k+1} &\leftarrow \left[\boldsymbol{\delta}_{k+1}\right]_{\setminus 0} && \in \mathbb{R}^{N_b-1-k} \\ \hat{\boldsymbol{v}}_{x_{k+1}} &\leftarrow \left[\boldsymbol{v}_{x_{k+1}}\right]_{\setminus -1} && \in \mathbb{R}^{N_b-1-k} \\ \hat{\boldsymbol{v}}_{y_{k+1}} &\leftarrow \left[\boldsymbol{v}_{y_{k+1}}\right]_{\setminus -1} && \in \mathbb{R}^{N_b-1-k} \\ \hat{\boldsymbol{\omega}}_{k+1} &\leftarrow \left[\boldsymbol{\omega}_{k+1}\right]_{\setminus -1} && \in \mathbb{R}^{N_b-1-k} \end{aligned} \tag{3-33}$$

Now that the velocity states and control inputs have been advanced from k to $k+1$ and have been sized properly, the updated input data matrix $\boldsymbol{X}_{k+1}$ for the next GP evaluation can be reconstructed,

$$\boldsymbol{X}_{k+1} = \begin{bmatrix} \tau_{k+1} & \delta_{k+1} & \hat{v}_{x_{k+1}} & \hat{v}_{y_{k+1}} & \hat{\omega}_{k+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \tau_{N_b-1} & \delta_{N_b-1} & \hat{v}_{x_{N_b-1}} & \hat{v}_{y_{N_b-1}} & \hat{\omega}_{N_b-1} \end{bmatrix} \quad \in \mathbb{R}^{(N_b-1-k)\times n_d} \tag{3-34}$$

Note that the velocity state variables are simulated future velocities, derived from the acceleration outputs provided by the (GPs). In this way, we employ the GP at its current level of understanding (referred to as the current epoch) to predict future dynamics during training. This simulation process is repeated several cycles until a desired k_{pred} horizon is reached. Equations (3-23) till (3-34) are repeated until the mean and covariance have been propagated the desired number of future steps ahead, i.e. $k = k_{pred} - 1$.

When the dynamics have been propagated several steps forward, the $k_{step} - 1$ covariance matrices will be fed into the optimization function along with the target values advanced the same number of k-steps ahead such that they are aligned with the simulated future dynamics

$$\begin{aligned} \boldsymbol{a}_x &\leftarrow \left[\boldsymbol{a}_x\right]_{\setminus (0:k_{pred}-1)} \\ \boldsymbol{a}_y &\leftarrow \left[\boldsymbol{a}_y\right]_{\setminus (0:k_{pred}-1)} \\ \boldsymbol{a}_\omega &\leftarrow \left[\boldsymbol{a}_\omega\right]_{\setminus (0:k_{pred}-1)} \end{aligned} \tag{3-35}$$

The kernel hyperparameters and inducing points are then optimized according to

$$\begin{aligned}\boldsymbol{\theta}_x^+ &\leftarrow \text{argmin}_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{a}_x, \boldsymbol{K}^{a_x}_{k_{pred}-1}, \boldsymbol{\theta}_x) \\ \boldsymbol{\theta}_y^+ &\leftarrow \text{argmin}_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{a}_y, \boldsymbol{K}^{a_y}_{k_{pred}-1}, \boldsymbol{\theta}_y) \\ \boldsymbol{\theta}_\omega^+ &\leftarrow \text{argmin}_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{a}_\omega, \boldsymbol{K}^{a_\omega}_{k_{pred}-1}, \boldsymbol{\theta}_\omega) \ ,\end{aligned} \tag{3-36}$$

and the process repeats for the next data batch.

Chapter 4

Supporting Information on Results: Modeling

This section provides supplementary information on the outcomes of fitting the Gaussian Process (GP) model, discussing the specific configurations employed during training and the criteria used for method validation and evaluation. Section 4-1 presents the training results under both the standard GP model training framework and the proposed GP model training framework, using a small dataset first for easy visualization and comparison of the methods. Meanwhile, Section 4-2 details the training outcomes on the complete dataset, which have been used for the models in the Model Predictive Contouring Control (MPCC) controller simulations.

4-1 Small Dataset GP training insights

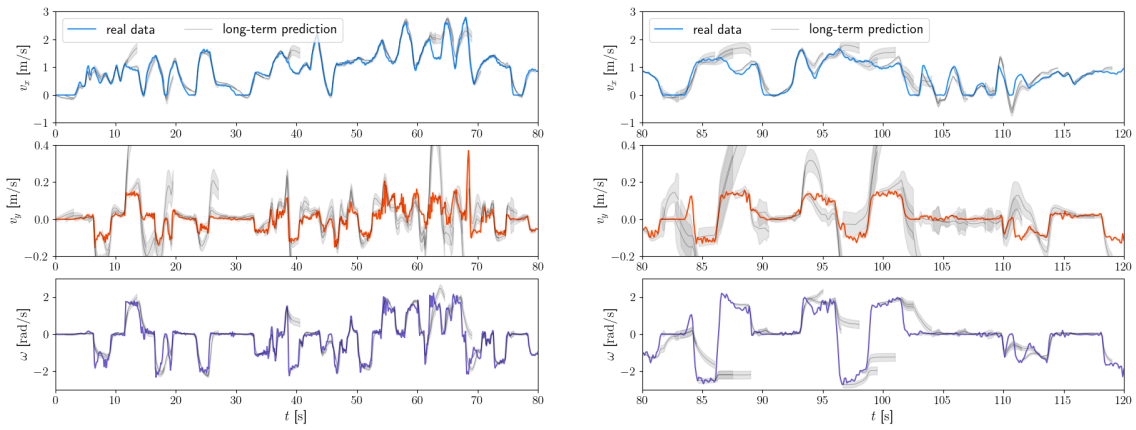
In this chapter, the data sections are presented to highlight the distinctions among the various training methods. The plots showcase real recorded data of longitudinal, lateral, and angular velocities, depicted by colored lines in blue, orange, and purple, respectively. Subsequently, GP models are employed to predict long-term velocities, initialized from different points along the recorded data. The GP utilizes the state/action data point at which it is initialized (longitudinal, lateral, angular velocity, throttle, and steering input) to predict an acceleration. The predicted acceleration is then multiplied by the time step, $\Delta t = 0.1$ s, to estimate the subsequent velocity state. This is done for all acceleration dimensions. The models are trained on 120 seconds of data, which is divided into a 70% training set and a 30% validation set. The GP posterior is approximated using Stochastic Variational Gaussian Process (SVGP) with $N_{ind} = 50$ inducing points. The implementation and training have been done using Python's GPyTorch library.

The evaluation of the methods was based on several metrics:

- Root Mean Square Error (RMSE) of the long-term velocity predictions

- Mean Absolute Error (MAE) of the long-term velocity predictions
- Average variance (Avg Var) of the long-term velocity predictions
- Training time
- Number of training iterations (epochs)

Fig. 4-1a shows the velocity trajectory predictions made using the accelerations predicted by the GP models trained under the standard framework as described in Section 3-1 and integrated using forward Euler method. Colored lines denote experimental velocity data recorded over time: longitudinal velocity (blue), lateral velocity (orange), and angular velocity (purple). Gray lines illustrate the mean velocity predictions, initialized at various points along the recorded data. The velocity trajectories based on this GP model often go out-of-distribution, denoted by cases where the predictions rapidly diverge from where they were initialized and eventually flatten out. Note that even though time is indicated on the x-axis, it is important to stress that the learned dynamics do not depend on time, as it is not an input feature; the inputs are strictly the velocity states and control inputs. To assess the generalization capabilities of the models, the training methods were first tested on a training set, comprising of 70% of the data. Additionally, a validation set comprising 30% of the data, which was not used during the training phase, was used to evaluate the performance of the trained Gaussian Process (GP) models on new, unseen data points. Fig. 4-1b further confirms that the GP model trained under the standard training framework is poor at making stable long-term predictions. Table 4-1 shows the training time, number of training cycles and the metrics by which the long-term velocity trajectory predictions have been assessed.



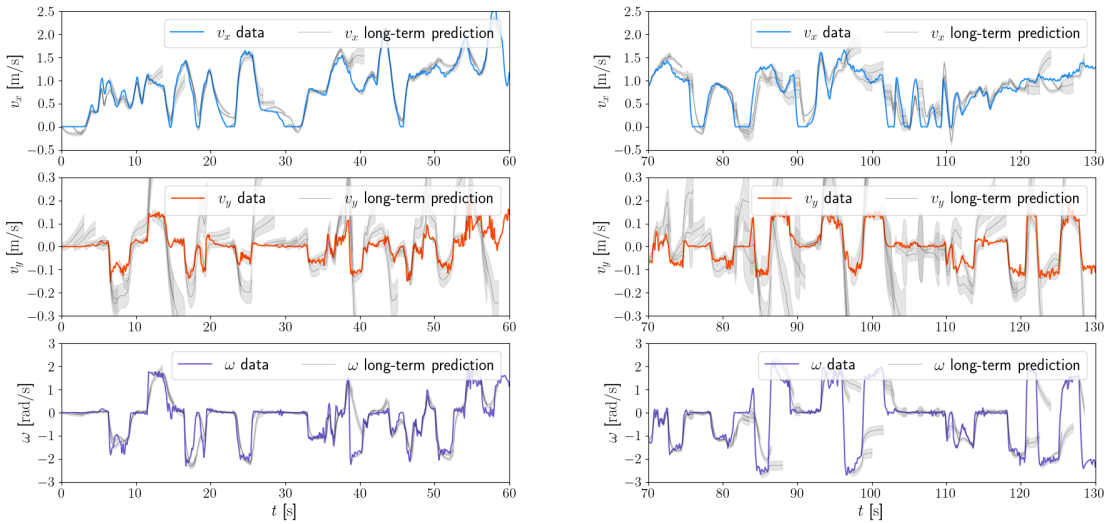
(a) Training data set **(b)** Validation data set

Figure 4-1: $N_{pred} = 30$ steps ahead long-term predictions of velocity trajectories based on the GP model that was trained under the standard training framework. 400 epochs.

To demonstrate that the GP trained using the standard framework has inherent limitations in making stable long-term predictions we also conducted training with fewer epochs, with the aim to demonstrate that the limitations observed in the previous model were not simply a result of overfitting or underfitting. As shown in Fig. 4-2a and 4-2b when the training was limited to 300 epochs, the long-term prediction performance further deteriorated.

Table 4-1: Assessment of long-term predictions of the longitudinal v_x, lateral v_y and angular ω velocities by the Gaussian Process (GP) model trained under the standard framework.

Metrics	***Longitudinal***	***Lateral***	***Angular***
RMSE	0.1733	0.1022	0.4499
MAE	0.1083	0.0534	0.2183
Avg. Var	0.6796e-3	0.3465e-3	2.334e-3
Training time [min]	1:12		
Epochs	400		

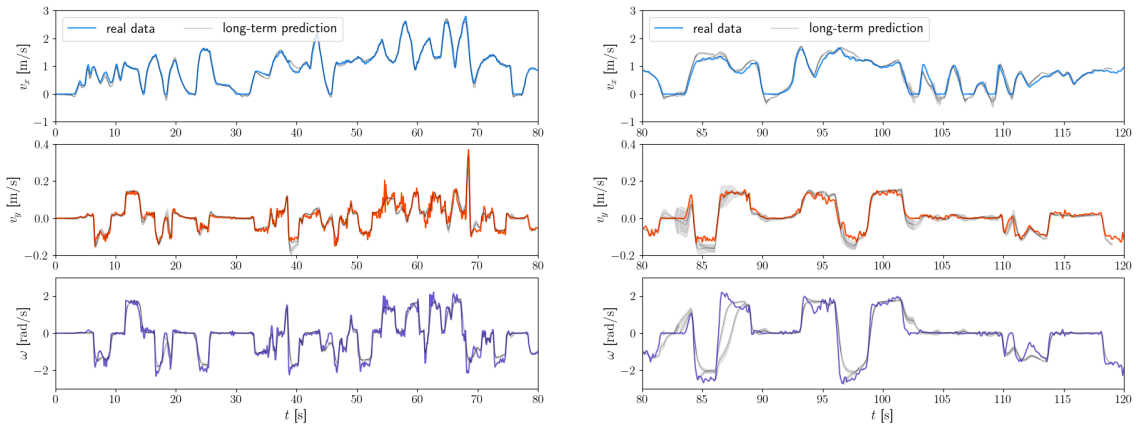


(a) Training data set **(b)** Test data set

Figure 4-2: $N_{pred} = 30$ steps ahead long-term predictions of velocity trajectories based on the GP model that was trained under the standard training framework. 300 epochs.

Figure 4-3a illustrates the velocity trajectories predicted by the MSGP models, as trained according to the method outlined in Section 3-2-2. Unlike the standard GP models which tend to deviate from the recorded data, the MSGP model's predictions closely align with the actual data, thus demonstrating more stable long-term predictions with significantly reduced uncertainty. Table 4-2 presents the training duration, number of training cycles, and the metrics used to evaluate the long-term velocity trajectory predictions. Both the mean prediction error and the associated uncertainties have decreased. To verify the efficacy of the proposed training framework, the MSGP model trained under this framework has not only been tested on the training data but also on a separate validation set (Fig. 4-3b). In contrast to the standard GP model, which exhibited failures on both training and test datasets, the MSGP model has successfully made stable long-term velocity trajectory predictions using unseen data, indicating effective generalization.

Additionally, the data indicates an increase in computation time, attributable to the multiple



(a) Training data set **(b)** Test data set

Figure 4-3: $N_{pred} = 30$ steps ahead long-term predictions of velocity trajectories based on the MSGP model. 250 epochs.

GP evaluations required during the training phase. Nevertheless, the number of epochs has decreased, indicating that fewer optimization steps are necessary to identify the optimal kernel hyperparameters and inducing points. This suggests that under the proposed framework, the model can learn more efficiently and effectively from the same amount of data.

Table 4-2: Assessment of long-term predictions of the longitudinal v_x, lateral v_y and angular ω velocities by the MSGP model trained under the proposed framework.

Metrics	***Longitudinal***	***Lateral***	***Angular***
RMSE	0.1244	0.0293	0.2519
MAE	0.0880	0.0165	0.1656
Avg Var	4.0740e-5	1.1995e-5	4.6023e-4
Δ% RMSE* [%]	-28	-71	-44
Δ% MAE* [%]	-11	-36	-12
Δ% Avg Var* [%]	-94	-97	-80
Training time [min]	3:57		
Epochs	250		

Metrics used are root mean square error (RMSE), mean absolute error (MAE), average variance (Avg Var).
*The percentual change Δ% is computed w.r.t. the standard GP training method.

4-2 GP model training for controller simulations

This section uses the same training approaches as described in the previous section, but applies it to a larger data set consisting of 360 seconds of data split in a 80% training and 20% validation data set. The GP posterior is approximated using SVGP with $N_{ind} = 200$ inducing points. These trained models based on larger datasets are used for the controller simulations, which will be further elaborated on in Chapter 5.

Fig. 4-4 shows the velocity trajectory estimated using the predicted accelerations by the GP models trained under the standard framework as described in Section 3-1. It can be seen that the long-term predictions often become unstable and go out of distribution. Table 4-3 specifies the training time, number of training cycles and the metrics by which the long-term velocity trajectory predictions have been assessed.

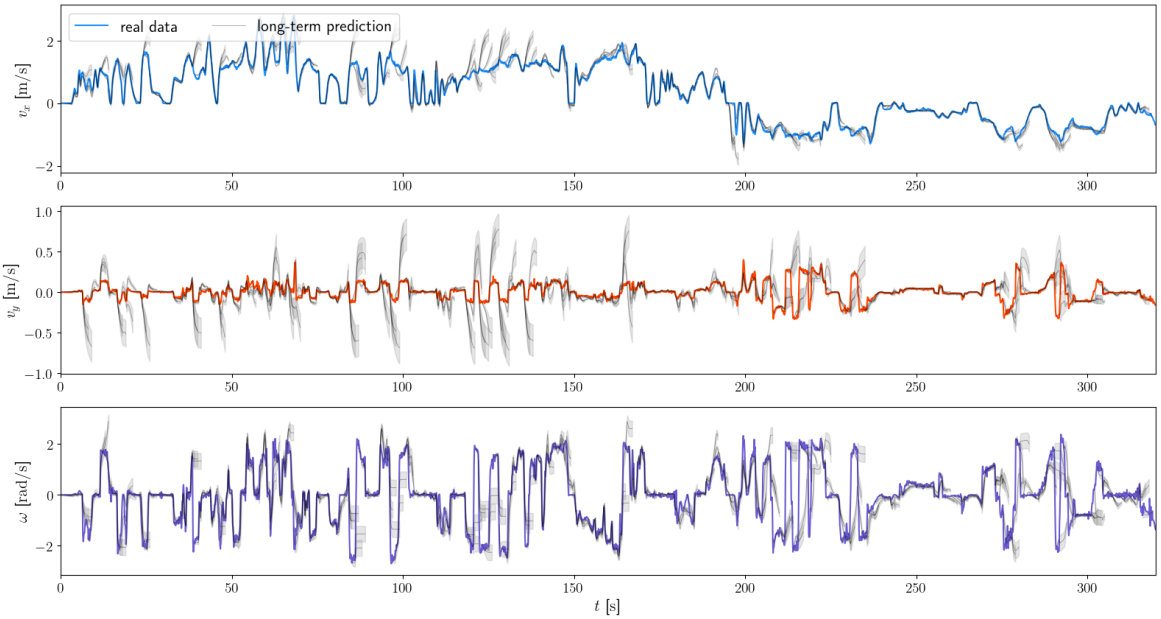


Figure 4-4: Long-term velocity trajectory prediction using standard GP model (tested on training data)

Table 4-3: Assessment of long-term predictions of the longitudinal v_x, lateral v_y and angular ω velocities by the Gaussian Process (GP) model trained under the standard framework. Computations are based on velocity trajectory predictions on the training dataset.

Metrics	***Longitudinal***	***Lateral***	***Angular***
RMSE	0.1613	0.1741	0.8924
MAE	0.0954	0.0833	0.4508
Avg Var	0.0003	0.0013	0.0261
Training time [min]	4:56		
Epochs	400		

Fig. 4-5 shows the predictions on new, unseen data points (the validation dataset), further highlighting that the long-term velocity predictions become unstable and go out of distribution.

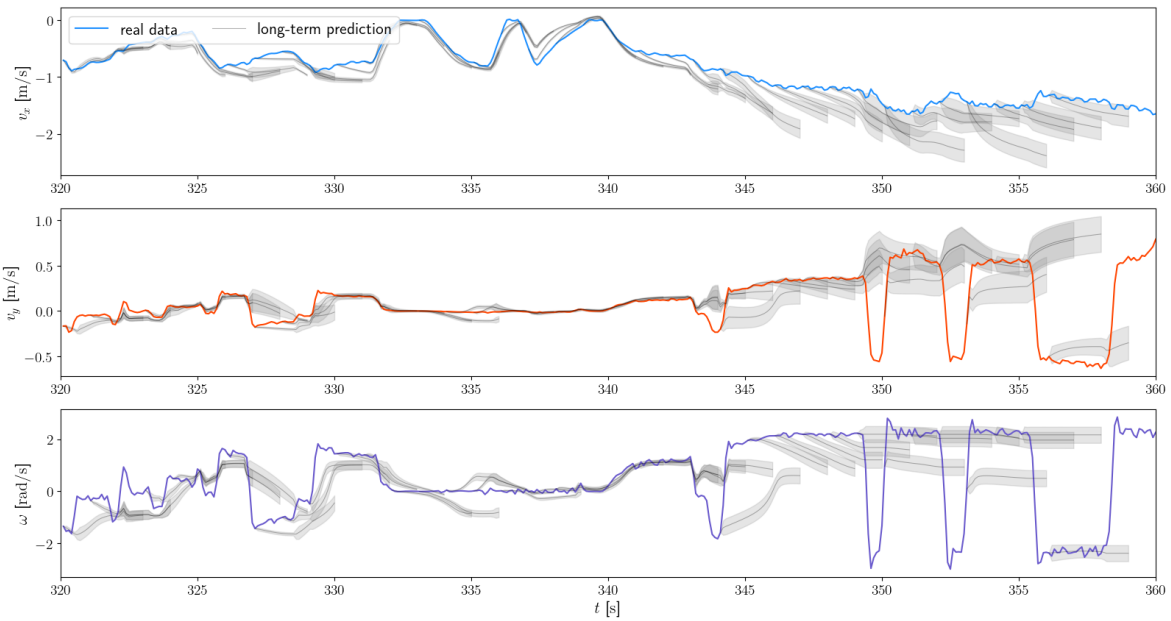


Figure 4-5: Long-term velocity trajectory prediction using standard GP model (tested on validation data)

Fig. 4-6 illustrates the velocity trajectory estimations derived from the predicted accelerations by the MSGP models, which were developed under the training framework outlined in Section 3-2-2. In contrast to the standard GP models, which often drift away from the actual data, the predictions from the MSGP models closely follow the true data, showing a more consistent prediction of dynamics over time with reduced uncertainty. The generalization ability of the MSGP models was tested on a validation dataset, confirming that their capability to predict stable long-term velocity trajectories also applies to unseen data (Fig. 4-7). Table 4-4 highlights the advantages of MSGP models over standard GP models in terms of specified metrics. Even though, the training time has tripled (an observation consistent with the shorter data excerpts) the number of training cycles required has lowered from 400 to 250 epochs. This indicates that the models trained under the proposed framework learn the underlying dynamics more quickly. It is important to note that despite the increase in training time, both the GP and MSGP models maintain the same level of complexity, ensuring no additional complexity at controller runtime.

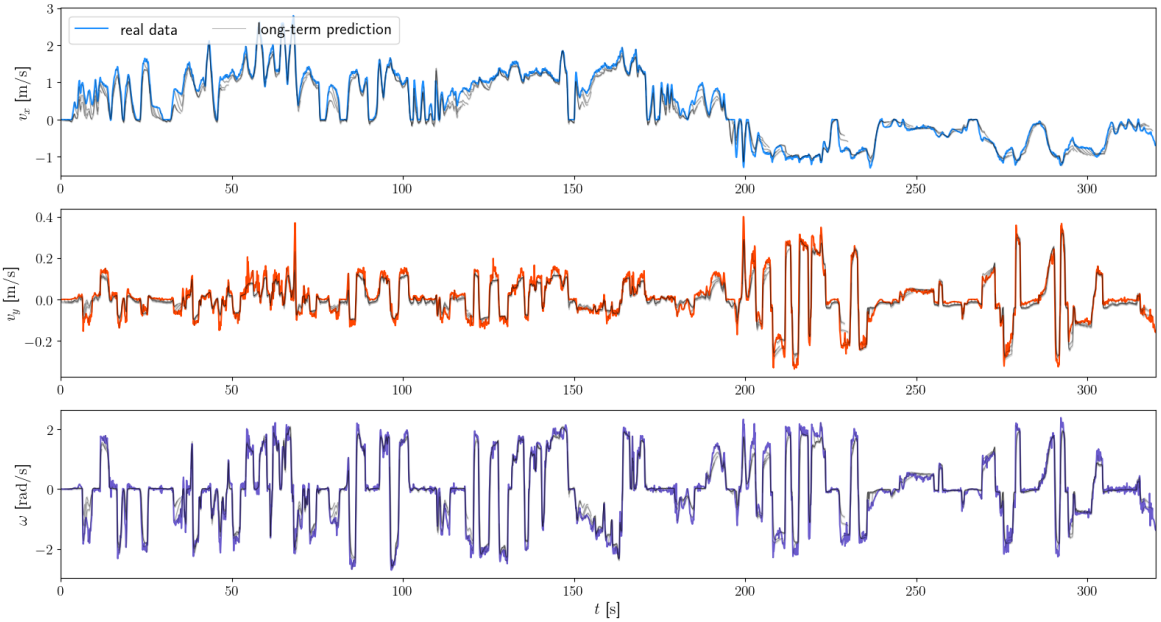


Figure 4-6: Long-term velocity trajectory prediction using MSGP model (tested on training data)

Table 4-4: Assessment of long-term predictions of the longitudinal v_x, lateral v_y and angular ω velocities by the MSGP model trained under the proposed framework.

Metrics	***Longitudinal***	***Lateral***	***Angular***
RMSE	0.1284	0.0349	0.2724
MAE	0.0903	0.0225	0.1694
Avg Var	4.5820e-5	7.4750e-6	0.0003
Δ% RMSE* [%]	−20	−79	−69
Δ% MAE* [%]	−3	−35	−31
Δ% Avg Var* [%]	−85	−99	−98
Training time [min]	15:39		
Epochs	250		

Metrics used are root mean square error (RMSE), mean absolute error (MAE), average variance (Avg Var).
*The percentual change Δ% is computed w.r.t. the standard GP training method.

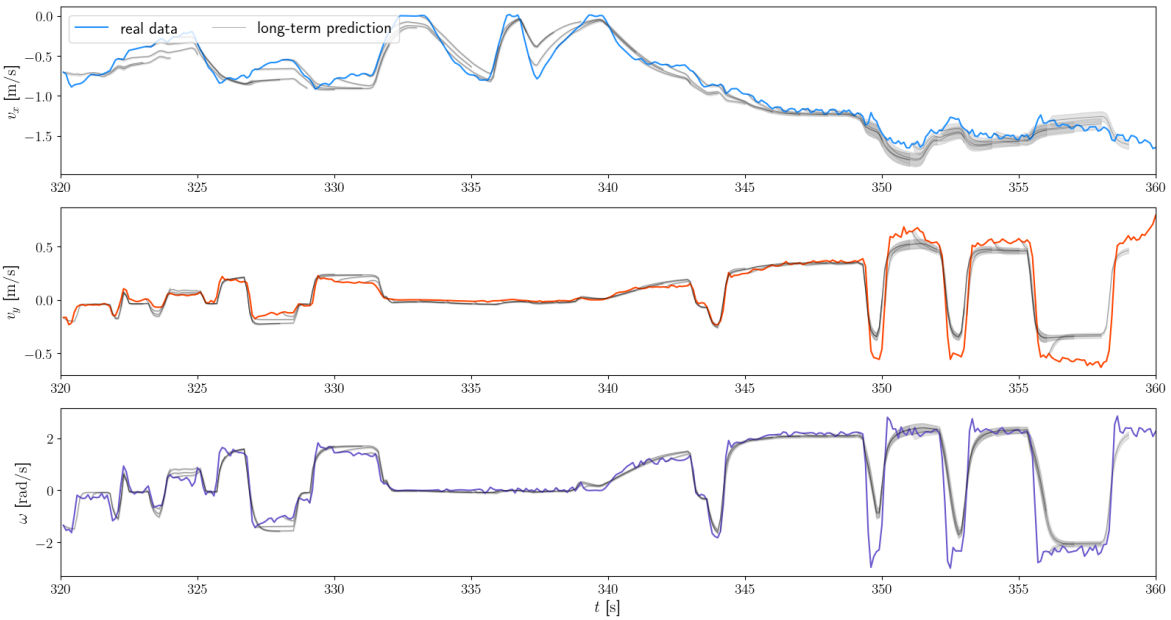


Figure 4-7: Long-term velocity trajectory prediction using MSGP model (tested on validation data)

Table 4-5: Assessment of long-term predictions of the longitudinal v_x, lateral v_y and angular ω velocities by the MSGP model trained under the standard framework. (test data)

Metrics	***Longitudinal***	***Lateral***	***Angular***
Δ% RMSE* [%]	-26	-68	-61
Δ% MAE* [%]	-22	-28	-33
Δ% Avg Var* [%]	-40	-90	-95

Metrics used are root mean square error (RMSE), mean absolute error (MAE), average variance (Avg Var).
*The percentual change Δ% is computed w.r.t. the standard GP training method.

Chapter 5

Supporting Information on Results: Control

This chapter explores the capabilities and limitations of the Gaussian Process (GP) and Multi-Step Gaussian Process (MSGP) model in an Model Predictive Contouring Control (MPCC) controller.

5-1 Simulation setup

The controller and simulation setup used are based on the works of [15]. The robot in the simulation environment is represented by a kinematic bicycle model,

$$\begin{aligned} \dot{v}_x &= \frac{F_x}{m} \\ \dot{x} &= v_x \cos(\theta) \\ \dot{y} &= v_x \sin(\theta) \\ \dot{\theta} &= \frac{v_x \tan(\delta)}{L} \ , \end{aligned} \tag{5-1}$$

in which the variables (x, y, θ) represent the pose of the robot, m denotes its mass, and F_x the force exerted by the motor. The term v_x corresponds to the robot's longitudinal velocity, while δ represents the steering angle of the robot's front wheels. Lastly, the wheelbase L denotes the distance between the front and rear axles of the robot. [15].

The MPCC cost function is defined as (5-2). In this MPCC formulation, s is the path along which the robot progresses, $\dot{s}$ is the robot's velocity along this path, and $\dot{s}_{ref}$ is the reference velocity in m/s. $\mathcal{U}$ is the set that forms the constraints on the control inputs. Lane boundaries are defined as soft constraints on the normalized contouring error e^c using a slack variable ε. q_1, q_2 are weights on the velocity and contouring error, respectively. $R_u = \text{diag}(r_1, r_2)$ is a

diagonal weighting matrix on the control inputs. p_1 is the weight on the slack variable ε.

$$\begin{aligned}
\min_{\boldsymbol{u}_{t+k}\in\mathcal{U}} \quad & \sum_{k=0}^{N-1} q_1 \cdot (\dot{s}_{t+k} - \dot{s}_{ref})^2 + q_2 \cdot (e^c_{t+k})^2 + \boldsymbol{u}^T_{t+k} R_u \boldsymbol{u}_{t+k} + p_1 \cdot \varepsilon^2 \\
\text{s.t.} \quad & \boldsymbol{x}_0 = \boldsymbol{x}(0) \\
& s_0 = s(0) \\
& \boldsymbol{x}_{t+k+1} = \boldsymbol{f}(\boldsymbol{x}_{t+k}, \boldsymbol{u}_{t+k}) \\
& s_{t+k+1} = f(s_{t+k}) \\
& \boldsymbol{u}_{t+k} \in \mathcal{U} \\
& 0 \leq s_{t+k} \leq L \\
& -(1+\varepsilon) \leq \frac{e^c_{t+k}}{\frac{l_{width}}{2}} \leq 1 + \varepsilon \\
& \forall k \in [0, N-1] \quad .
\end{aligned} \tag{5-2}$$

The cost function is adaptable to different control objectives, with varying priorities based on predefined weights. This flexibility allows for tuning specific dynamic behaviors, such as precise velocity tracking and minimizing lateral error for conservative driving, as well as more aggressive maneuvers where the same objectives would be less prioritized. This study aims to evaluate the models' performance under highly aggressive conditions. To achieve this objective, an emphasis is placed on promoting high speeds. This is accomplished by intentionally keeping the weight assigned to the throttle input low. This strategy encourages the controller to explore its limits and observe the response of each model in such demanding situations. The prediction horizon of the MPCCs is set to $N_{pred} = 30$. The sampling rate is set to $T_s = 0.1$s, enabling the controller to predict three seconds ahead. ForcesPro [6] is employed for solving the nonlinear optimization problem. Control commands are transmitted at a frequency of 10 Hz via a Robot Operating System (ROS) network to control the robot within an RViz simulation environment from [15]. The robot odometry data is returned through the same ROS network.

To test the simulation setup's functionality, tests were conducted using a MPCC based on a kinematic bicycle model. The performance of the controller in velocity tracking and maintaining a close distance to the path's center was assessed by assigning higher penalties to velocity tracking errors and deviations from the path. Table 5-1 shows the controller settings. Since there is no discrepancy between the simulation model and the MPCC prediction model, we anticipate precise control and tracking.

Table 5-1: Controller settings for reference tracking, target speed 1 m/s

Weights				**Input constraints**		**Prediction horizon**	**Target velocity**
q_1	q_2	r_1	r_2	$\underline{\delta}, \bar{\delta}$	$\underline{\tau}, \bar{\tau}$	N_{pred}	$\dot{s}_{ref}$ [m/s]
10	5	1	0.1	[-1, 1]	[0, 0.6]	30	1.0

5-2 GP vs MSGP target velocity 1 m/s

To assess the model's behavior under aggressive conditions, a focus is placed on achieving high speeds by setting the throttle input's weight low (Table 5-2). This approach pushes the controller to its limits, allowing the observation of each model's response in these challenging

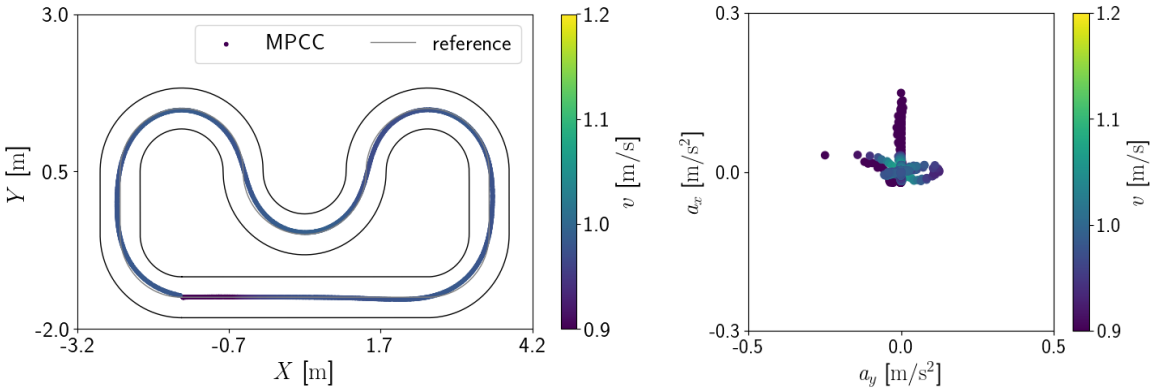


Figure 5-1: 1-lap tracking results (left). The grey center line represents the path reference. The inner and outer lines denote the track boundaries. The color gradient illustrates the velocity magnitude at each recorded vehicle position. The accelerations achieved by the MPCC over 1 lap (right). The acceleration values are presented in m/s^2 for clarity, without conversion to Gs.

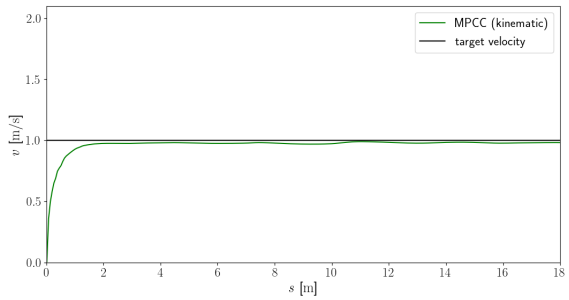


Figure 5-2: Velocity tracking results of the kinematic bicycle model-based MPCC

scenarios. Fig. 5-3 shows the velocity input map along with the target velocity. This map illustrates the longitudinal and lateral velocity data points that have been used to train the models. Here, the target velocity is set at 1 m/s, representing the scenario with the highest data density.

Table 5-2: GP-MPCC and MSGP-MPCC controller settings, target velocity 1 m/s

Weights				**Input constraints**		**Prediction horizon**	**Target velocity**
q_1	q_2	r_1	r_2	$\underline{\delta}, \bar{\delta}$	$\underline{\tau}, \bar{\tau}$	N_{pred}	$\dot{s}_{ref}$ [m/s]
1	1.2	1	0.1	[-1, 1]	[0, 0.6]	30	1.0

Figure 5-4a demonstrates that in the most data-dense scenario, the Gaussian Process-based Model Predictive Contouring Controller (GP-MPCC) already exhibits unstable driving behavior, illustrated by the wiggly paths. Conversely, Figure 5-4b reveals that the Multi-Step Gaussian Process-based Model Predictive Contouring Controller (MSGP-MPCC) operates confidently under identical conditions. This indicates that the model trained using the proposed framework has learned the dynamics better compared to the model trained with the

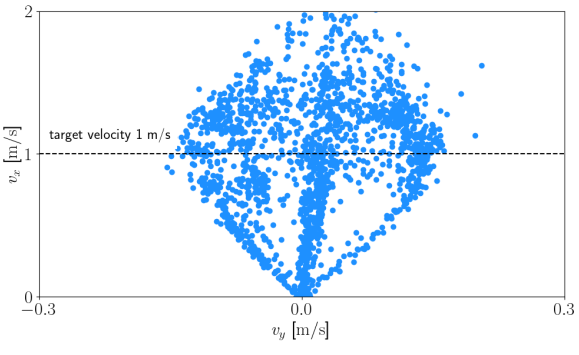
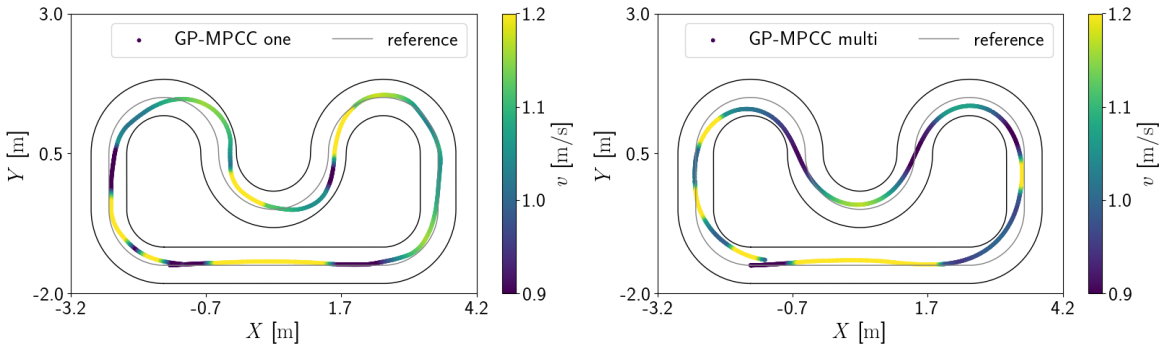


Figure 5-3: Velocity input map and respective target velocity of 1 m/s



(a) GP-MPCC **(b)** MSGP-MPCC

Figure 5-4: 1-lap tracking results of the GP-MPCC (left) and the MSGP-MPCC (right). Target speed 1 m/s.

standard framework.

5-3 MSGP target velocity 1.3 m/s

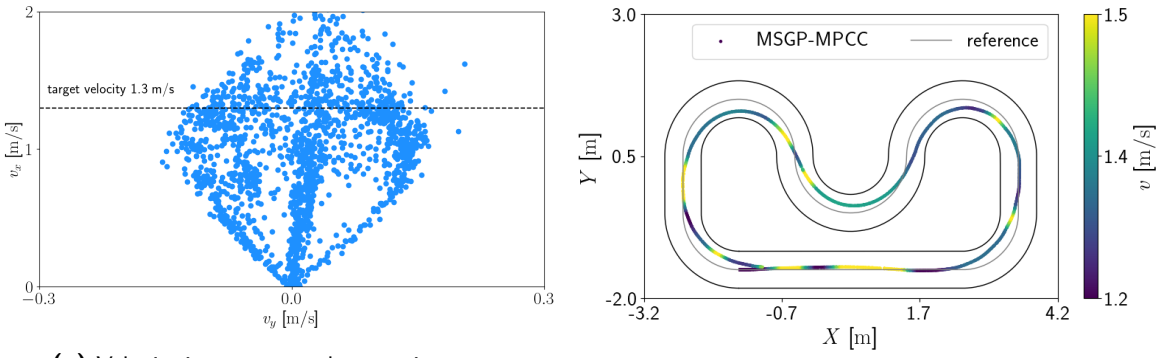
To evaluate how well the MSGP model has improved in predicting dynamics and to further explore its limitations, it is tested in an MPCC at progressively increasing target velocities. This pushes the model into scenarios with less available data to observe its performance. Only the MSGP model undergoes this testing, as it has been previously observed that the GP model fails at the target velocity of 1 m/s where the data is least scarce. Since the target speed has increased, the weights on the tracking errors are increased to ensure stable control. Table 5-4 shows the remainder of the MSGP-MPCC controller settings.

Table 5-3: MSGP-MPCC controller settings, target velocity 1.3 m/s

Weights				**Input constraints**		**Prediction horizon**	**Target velocity**
q_1	q_2	r_1	r_2	$\underline{\delta}, \bar{\delta}$	$\underline{\tau}, \bar{\tau}$	N_{pred}	$\dot{s}_{ref}$ [m/s]
10	10	2	0.1	[-1, 1]	[0, 0.6]	30	1.0

Despite the limited data available at higher speeds, the MSGP-MPCC model maintains sta-

bility and avoids venturing into out-of-distribution states during extreme maneuvers. Fig. 5-5a presents the velocity input map with a target velocity of 1.3 m/s, where the available velocity data is notably less abundant compared to the 1 m/s region. Fig. 5-5b demonstrates that even as the robot is pushed to its limits in situations with limited training data, the robot controlled by the MSGP-MPCC is still able to navigate the track safely. Notably, while the GP-MPCC model already struggled in the most data-rich scenario at 1 m/s, the MSGP-MPCC model performs effectively not only at this baseline speed but also at the more challenging speed of 1.3 m/s, where data is scarcer.



(a) Velocity input map and respective target velocity of 1.3 m/s

(b) 1 lap tracking results of MSGP-MPCC at target 1.3 m/s

Figure 5-5: 1 lap tracking results of the controller based on the MSGP model trained under the proposed cascaded framework. Target velocity 1.3 m/s.

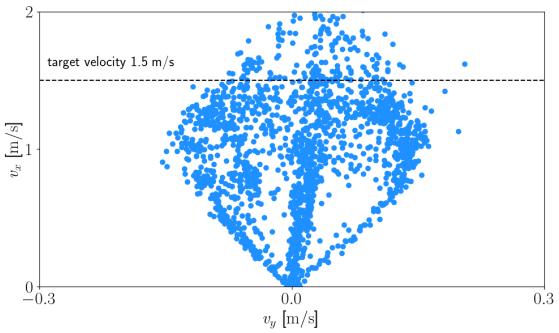
5-4 MSGP target velocity 1.5 m/s

While the GP-MPCC model exhibited instability at a moderate speed of 1 m/s, the MSGP-MPCC managed racetrack navigation confidently at speeds of both 1 m/s and 1.3 m/s. It only began to show erratic behavior when the speed was increased to 1.5 m/s, as depicted in Figure 5-6b. This illustrates that the MSGP model, developed under the proposed framework, more effectively utilizes the available data to develop a better understanding of the underlying dynamics. Consequently, it surpasses the GP-MPCC in more demanding scenarios by maintaining stable behavior at higher velocities.

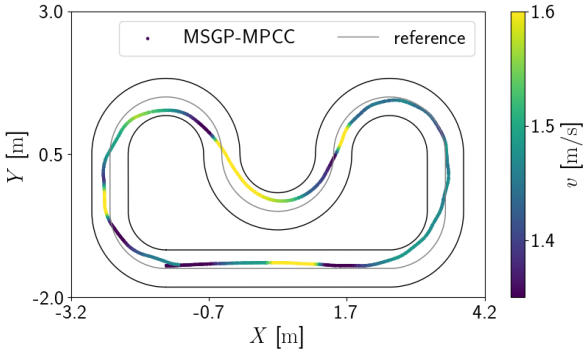
Table 5-4: MSGP-MPCC controller settings, target velocity 1.5 m/s

Weights				**Input constraints**		**Prediction horizon**	**Target velocity**
q_1	q_2	r_1	r_2	$\underline{\delta}, \bar{\delta}$	$\underline{\tau}, \bar{\tau}$	N_{pred}	$\dot{s}_{ref}$ [m/s]
5	10	2	0.1	[-1, 1]	[0, 0.6]	30	1.0

The observation that MSGP-MPCC performance begins to decline at 1.5 m/s emphasizes that although the proposed MSGP training framework extracts more information from data (or we can learn the same amount from less data), data availability is still crucial for optimal model performance, even with an advanced training framework.



(a) Velocity input map and respective target velocity of 1.5 m/s



(b) 1 lap tracking results of MSGP-MPCC at target 1.5 m/s

Figure 5-6: 1 lap tracking results of the controller based on the MSGP model trained under the proposed cascaded framework. Target velocity 1.5 m/s.

Chapter 6

Discussion

This thesis introduced a novel method for training Gaussian Process (GP) models to accurately predict long-term stable vehicle dynamics, in this study: the dynamics of a 1:10 scale robot car. This chapter discusses the interpretation of the results and limitations, their relevance to control applications, and suggests potential directions for future research.

6-1 Gaussian Process Modeling

This thesis presented a training approach for GP models, focusing on strategically handling out-of-distribution scenarios and stabilizing long-term dynamics predictions. The hypothesis was that refining the parameters of the kernel function would enhance the model's capability to capture long-term dynamics and provide accurate predictions of their respective uncertainties. To address this, a Multi-Step Gaussian Process (MSGP) model training framework was developed. The framework relies on in-training simulation of future dynamics, where the GP acceleration output is used to derive new velocity state input estimates. These estimates, along with the recorded control inputs, are used to advance the dynamics several steps ahead, minimizing the discrepancy between the prediction and the actual target values at the final time step. This methodology equips the GP with predictive foresight and uncertainty quantification capabilities, making them better suited for integration with Model Predictive Control (MPC). By employing three distinct Gaussian Processes (GPs) to model the complete dynamics of the vehicle — longitudinal, lateral, and angular accelerations — the MSGP model achieved an average reduction of 19% in long-term velocity mean prediction error and a 90% decrease in long-term prediction variance compared to the standard GPs. Notably, these enhancements were realized using the same dataset that trained the standard Gaussian Process models, and the models have identical complexity. This indicates that this training method facilitates a more profound comprehension of the underlying relationships in vehicle dynamics.

The enhanced performance can be attributed to the retuning of the kernel hyperparameters, which yielded the following improvements:

- **Recalibrated sensitivity** By retuning the kernel parameters, MSGP models have adapted their sensitivity to specific input features. Longer length scales yielded smoother functions and a slightly more conservative response for immediate dynamics, ensuring long-term stability and minimizing the risk of shooting out-of-distribution.
- **Rectified coupling** The MSGP has acquired an improved understanding of the level of coupling between longitudinal and lateral dynamics. By elongating the kernel parameter of the lateral input features (steering, lateral, and angular velocity states) within the longitudinal acceleration model, the framework ensures the appropriate consideration of lateral dynamics, without disproportionately affecting longitudinal acceleration predictions.
- **Reduced uncertainty** Now that the long-term predictions are stable and the predicted future dynamics consistently correlate with the training data, the uncertainty reduction term in the GP's predictive equations for the covariance is activated, leading to a decrease in variance associated with the mean predictions.

This work addressed shortcomings in the current literature about the application of GPs in control systems. Traditional GP applications typically aim to align nominal physics-based models with actual system behavior by addressing residual errors. However, as identified in research by [2], these methods may fail to yield stable long-term predictions, particularly when GPs face out-of-distribution data points. While the literature has not directly tackled out-of-distribution behavior in GP-based control, there is research focused on managing uncertainties that partly arise from unstable GP predictions. These studies often view large uncertainties as an isolated issue, proposing specific solutions for the control phase, such as the ancillary controllers mentioned by [13] or the truncated uncertainty propagation in [12]. In contrast, this thesis adopts a fundamentally different approach by concentrating on the modeling stage to enhance the long-term predictive reliability of GPs through a recalibration of kernel parameters that prevent GPs from going into data areas poorly correlated with the training data set. This approach not only aligns predicted dynamics more closely with actual dynamics but also substantially reduces the uncertainties associated with GP predictions.

6-2 Gaussian Process-based Control

The comparative analysis between the Multi-Step Gaussian Process-based Model Predictive Contouring Controller (MSGP-MPCC) controller and Gaussian Process-based Model Predictive Contouring Controller (GP-MPCC) highlighted the superior performance of the MSGP-MPCC. Notably, the MSGP-MPCC excelled in reducing lap times and executing more aggressive and coupled accelerations, while also demonstrating a high level of reliability. Through ten consecutive simulations of the controller, the MSGP-MPCC maintained a 100% control success rate, outperforming the GP-MPCC, which had a 20% success rate due to failures in navigating sharp turns in 80% of the simulations. Additionally, when tested at increasingly higher speeds, the MSGP model proved its robustness at higher speeds (1.3 m/s) where the data becomes more scarce. The MSGP model only starts to exhibit erratic behavior at 1.5 m/s, whereas the GP model staggered already at 1 m/s (where the data is least scarce).

6-3 Conclusion and Future Work

In conclusion, the proposed MSGP learning framework demonstrates significant improvements in long-term prediction accuracy and uncertainty quantification, attributed to recalibrated sensitivity, rebalanced coupling, and reduced uncertainty through tuning of the kernel parameters.

This research opens up several directions for future studies. Having established a method for training GP models to deliver accurate long-term predictions, a logical next step would be to conduct real-life tests using the MSGP model on the actual robot with dedicated hardware to get practical validation and insights of the proposed methodology. Additionally, the calibrated variances could be integrated into the Model Predictive Contouring Control (MPCC) long with the mean as a safety mechanism, such as dynamically adjusting lane boundaries or modifying obstacle occupancy areas based on the given variance value. Furthermore, future research could investigate the advantages of combining physics-based models with GPs, using the proposed framework to enhance stability in error dynamics and potentially leading to more robust modeling solutions.

Bibliography

[1] National Highway Traffic Safety Administration. National motor vehicle crash causation survey, 2008.

[2] Roushan R. Arany. Gaussian process model predictive control for autonomous driving in safety-critical scenarios. Master's thesis, Linköping University, 2019.

[3] Manuel Blum and Martin Riedmiller. Optimization of gaussian process hyperparameters using rprop. *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2013.

[4] Bruno Brito, Boaz Floor, Laura Ferranti, and Javier Alonso-Mora. Model predictive contouring control for collision avoidance in unstructured dynamic environments. *IEEE Robotics and Automation Letters*, PP:1–1, 07 2019.

[5] Bindell D. Numerical methods of data science: Kernels. *Cornell University*, 2018.

[6] A Domahidi and J. Jerez. Forces professional, embotech gmbh (http://embotech.com/forces-pro). 2014.

[7] D Duvenaud. Automatic model construction with gaussian processes, phd thesis. Master's thesis, University of Cambridge, 2014.

[8] Daniel J. Fagnant and Kara Kockelman. Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, 77:167–181, 2015.

[9] Rolf Findeisen and Frank Allgöwer. An introduction to nonlinear model predictive control. 01 2002.

[10] Giovanni Franzese. Introduction to gaussian processes for robotics and control lecture notes. *Delft University of Technology*, 2024.

[11] David González, Joshué Pérez, Vicente Milanés, and Fawzi Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1135–1145, 2016.

[12] Lukas Hewing, Juraj Kabzan, and Melanie N. Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4:3363–3370, 2019.

[13] Lukas Hewing, Juraj Kabzan, and Melanie N. Zeilinger. Cautious model predictive control using gaussian process regression. *IEEE Transactions on Control Systems Technology*, 28:2736–2743, 2020.

[14] Wenjun Liu, Chang Liu, Guang Chen, and Alois Knoll. Gaussian process based model predictive control for overtaking in autonomous driving. *frontiers in Neurorobotics*, 2021.

[15] Lorenzo Lyons, Giovanni Franzese, Riccardo Chiesa, Jens Kober, and Laura Ferranti. Promoting safety under localized data: A prudent model predictive control for autonomous ground vehicles. *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[16] World Health Organization. Road traffic injuries, 2022.

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

[18] Gonçalo Collares Pereira. Adaptive lateral model predictive control for autonomous driving of heavy-duty vehicles, phd thesis. Master's thesis, KTH Royal Institute of Technology, 2023.

[19] C.E. Rasmussen and C.K. Williams. *Gaussian Processes for Machine Learning*. MIT press Cambridge, MA, 2006.

[20] Scikit-learn. Gaussian processes, 2024. [Online; accessed April 15, 2024].

[21] Lars Svensson, Monimoy Bujarbaruah, Arpit Karsolia, Christian Berger, and Martin Törngren. Traction adaptive motion planning and control at the limits of handling. *IEEE Transactions on Control Systems Technology*, pages 1–17, 2021.

[22] Jie Wang. An intuitive tutorial to gaussian processes regression, 2022.

Glossary

List of Acronyms

Avg Var	Average variance
GP	Gaussian Process
GPs	Gaussian Processes
GP-MPCC	Gaussian Process-based Model Predictive Contouring Controller
MPC	Model Predictive Control
MPCC	Model Predictive Contouring Control
MSGP	Multi-Step Gaussian Process
MSGP-MPCC	Multi-Step Gaussian Process-based Model Predictive Contouring Controller
MVN	Multivariate Normal Distribution
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
SVGP	Stochastic Variational Gaussian Process