frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 12 August 2021
doi: 10.3389/fnbot.2021.723049

Gaussian Process Based Model Predictive Control for Overtaking in Autonomous Driving

Wenjun Liu[1], *Chang Liu*[1], *Guang Chen*[1,2]* *and Alois Knoll*[1]

[1] *Department of Informatics, Technical University of Munich, Munich, Germany,* [2] *School of Automotive Studies, Tongji University, Shanghai, China*

This paper proposes a novel framework for addressing the challenge of autonomous overtaking and obstacle avoidance, which incorporates the overtaking path planning into Gaussian Process-based model predictive control (GPMPC). Compared with conventional control strategies, this approach has two main advantages. Firstly, combining Gaussian Process (GP) regression with a nominal model allows for learning from model mismatch and unmodeled dynamics, which enhances a simple model and delivers significantly better results. Due to the approximation for propagating uncertainties, we can furthermore satisfy the constraints and thereby the safety of the vehicle is ensured. Secondly, we convert the geometric relationship between the ego vehicle and other obstacle vehicles into the constraints. Without relying on a higher-level path planner, this approach substantially reduces the computational burden. In addition, we transform the state constraints under the model predictive control (MPC) framework into a soft constraint and incorporate it as relaxed barrier function into the cost function, which makes the optimizer more efficient. Simulation results indicate that the proposed method can not only fulfill the overtaking tasks but also maintain safety at all times.

Keywords: autonomous driving, Gaussian process, model predictive control, overtaking, path planning

OPEN ACCESS

Edited by:
Xin Luo,
Chongqing Institute of Green and Intelligent Technology (CAS), China

Reviewed by:
Minzhi Chen,
China West Normal University, China
Lun Hu,
Chinese Academy of Sciences (CAS), China

****Correspondence:***
Guang Chen
guangchen@tongji.edu.cn

Received: *09 June 2021*
Accepted: *19 July 2021*
Published: *12 August 2021*

Citation:
Liu W, Liu C, Chen G and Knoll A (2021) Gaussian Process Based Model Predictive Control for Overtaking in Autonomous Driving. Front. Neurorobot. 15:723049. doi: 10.3389/fnbot.2021.723049

1. INTRODUCTION

Autonomous driving has attracted considerable attention because of its promising future (Chen et al., 2021; Kiran et al., 2021). A number of modern techniques have been employed for advanced driving assistant system, such as adaptive cruise control (Wu et al., 2019a), automatic parking (Ye et al., 2019), etc, which can be regarded as the low level autonomous driving. However, due to its demands of high reliability and real-time practicality of fully autonomous driving, performing overtaking maneuvers imposes a major challenge (Cha et al., 2018). Even for human beings, overtaking is also a dangerous task, therefore, reliable and safe autonomous overtaking systems are becoming more and more appealing (Lattarulo et al., 2018).

Model predictive control (MPC) has the ability to incorporate constraints into the online optimizations in a multivariable control framework and also provides a method to weigh the competing goals by carefully designing the cost function, so MPC is widely applied in control field. However, the control performance of MPC depends heavily on the accuracy of the acquired model. While vehicle dynamics are disreputable difficult to model in complex situations. Learning-based control method has been proposed and widely applied to solve this problem (Hewing et al., 2020; Xie et al., 2020). Gaussian process (GP) regression is the most commonly applied method in learning-based control. GP is also a non-parametric machine learning approach and has shown success in combining with model predictive control, i.e., Gaussian process model predictive control (GPMPC) method. Hewing et al. (2018) and Hewing et al. (2019) designed a GPMPC structure based on this conception to improve traditional MPC control performance for a race car, by making use of a relatively simple nominal model and an additive learned term, which solves the problems of computing demanding due to complex model and poor control performance due to inaccurate model. GPMPC has also applied in mobile robots to achieve path tracking control (McKinnon and Schoellig, 2019), where the control input can be modulated in time in the face of rapidly changing dynamics. Rezvani Arany (2019) use GPMPC to realize vehicle safety control in variable friction road conditions. In Langåker (2018), GPMPC is applied for vehicle obstacle avoidance.

In order to avoid collisions, there is one view stating that the tracking control strategy needs a higher-level path planner. This method was used in Gao et al. (2010), but the vehicle was modeled as a simple point mass model which neglects the vehicle kinematics and dynamics. This will strictly limit the performance when the speed of vehicle increases. A more complex dynamical model is adopted in the planner in Frazzoli et al. (2005) to generate the reference trajectories for the low level tracking controller. However, it is too complex to be solved due to the mixed-integer program optimization problem. Consequently, this approach is not well-suited for real-time overtaking task. Instead, one-level approach have been investigated recently. Liniger et al. (2015) combined the path planning and path tracking into one non-linear optimization problem, the path planner was based on dynamic programming and merged into model predictive contour control. However, they only take the stationary obstacles rather than moving vehicles into account. The complexity of obstacle avoidance would increase when the obstacle is moving, this approach may not work in this situation. A short-term path planning in Franco and Santos (2019) considered both static obstacles and moving vehicles, proposing a flexible overtaking paradigm based on adaptive MPC. Since the bicycle model is only concerned with kinematics, the lateral control with regard to tire model is simplified, the generated trajectories were a bit infeasible.

In this paper, we investigate the autonomous overtaking problems with GPMPC approach. There are three main contributions. Firstly, the overtaking problem is cleverly converted into constraint control, eliminating the need for path planning. Secondly, a single track model considering the non-linear wheel dynamics is adopted as the nominal model. GP is used to learn the unknown deviation between the nominal model and the true plant dynamics. Through the learning of GP, we can use a relatively simple vehicle model but get better control effect. By employing the Taylor approximation we can propagate the uncertainties and evaluate the residual uncertainties in the MPC prediction time domain, which increases the accurateness and cautiousness of the controller. Thirdly, we transform the state constraints under the model predictive control (MPC) framework into a soft constraint and incorporate it as relaxed barrier function into the cost function, which makes the optimizer more efficient.

The rest of this paper is constructed as follows. The vehicle model is introduced in section 2. In section 3, Gaussian process regression is introduced. In this section, we first present the preliminaries of GP, then give the approach of how to obtain training data, and give an approximate approach for the propagation of uncertainty in multi-step-ahead prediction. In section 4, we design GPMPC controller for vehicle overtaking problem. In section 5, simulations are conducted to verify the effectiveness of the proposed controller. Finally, we conclude in section 6.

2. VEHICLE MODEL

Establishing a reasonable vehicle model is not only a prerequisite for designing a model predictive controller, but also a basis for realizing vehicle overtaking control. Therefore, it is necessary to select control variables according to the driving conditions of the vehicle to establish a kinematics and dynamics model that can accurately describe the vehicle. However, if the model is too complex, it will affect the real-time performance of the control algorithm.

In this chapter, a simplified vehicle model is introduced to trade off computational performance and vehicle characteristics. A single track model is adopted in this paper as shown in **Figure 1**, where each side wheels are merged into one wheel. We assume that only the front wheel can steer. Only the longitudinal and lateral as well as yaw motion will be considered, the pitch and roll dynamics are neglected (Langåker, 2018). The vehicle is typically assumed to be a mass point with the global position coordinates (X, Y) and the yaw angle φ, while v_x and v_y represent the longitudinal and lateral velocities, respectively. ω refers to the yaw rate. The other parameters are vehicle mass M, yaw moment of inertia I_z, the steering angle δ, the distance between the center of gravity (c.g.) of the vehicle and the front and rear wheel are L_f and L_r, respectively. The forces which act on the front and rear

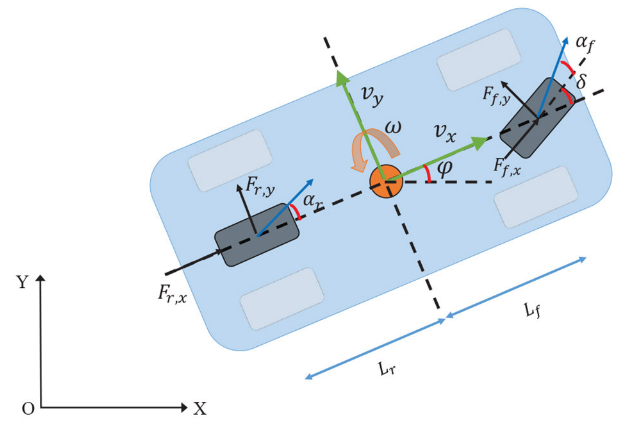

FIGURE 1 | A schematic drawing of the bicycle model.

wheel in longitudinal and lateral direction are defined by $F_{f,x}$, $F_{f,y}$, $F_{r,x}$, $F_{r,y}$. Finally, the front- and rear-slip angle are α_f and α_r, respectively. Then the vehicle model is given by

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) = \begin{bmatrix} v_x\cos(\varphi) - v_y\sin(\varphi) \\ v_x\sin(\varphi) + v_y\cos(\varphi) \\ \omega \\ \frac{1}{M}\left(F_{r,x} + F_{f,x}\cos(\delta) - F_{f,y}\sin(\delta) + M\omega v_y\right) \\ \frac{1}{M}\left(F_{r,y} + F_{f,x}\sin(\delta) + F_{f,y}\cos(\delta) - M\omega v_x\right) \\ \frac{1}{I_z}\left(F_{f,y}L_f\cos(\delta) + F_{f,x}L_f\sin(\delta) - F_{r,y}L_r\right) \end{bmatrix} \tag{1}$$

where $\boldsymbol{x} = [X; Y; \varphi; v_x; v_y; \omega]$ are the states of the system, while the input vector $\boldsymbol{u} = [\delta; T]$ to the system are the steering angle δ and the acceleration/brake pedal T ($T \in [-1, 1]$). The vehicle's velocity is controlled increasing or decreasing by the throttle, when it is set to $T > 0$ or $T < 0$.

The longitudinal wheel forces $F_{f/r,x}$ in vehicle coordinates are modeled simply as proportional to the acceleration/brake pedal T and the torque distribution ζ by

$$\begin{aligned} F_W &= T\left((T > 0)F_a + (T < 0)F_b\mathrm{sign}\left(V_x\right)\right) \\ F_{f,x} &= (1-\zeta)F_W \\ F_{r,x} &= \zeta F_W \end{aligned} \tag{2}$$

where F_a and F_b are acceleration force and brake force, respectively.

According to Pacejka and Bakker (1992), the lateral forces $F_{f,y}$ and $F_{r,y}$ are given by the full MAGIC formulas.

$$\begin{aligned} F_{f,y} &= D_f\sin\left[C_f\arctan\left(B_f\alpha_f - E_f\left(B_f\alpha_f - \arctan\left(B_f\alpha_f\right)\right)\right)\right] \\ F_{r,y} &= D_r\sin\left[C_r\arctan\left(B_r\alpha_r - E_r\left(B_r\alpha_r - \arctan\left(B_r\alpha_r\right)\right)\right)\right] \end{aligned} \tag{3}$$

where B_* is stiffness factor, D_* is peak factor, C_* and E_* are shape factors, α_* represents the front wheel slip angle and rear wheel slip angle, respectively.

However, the full MAGIC formulas are too complicated in practical applications. In order to ease the computational burden, the following simplified Pacejka Tire Model (Elbanhawi et al., 2018) is used, which is a linear approximation of (3).

$$\begin{aligned} F_{f,y} &= C_{l,f}\alpha_f \\ F_{r,y} &= C_{l,r}\alpha_r \end{aligned} \tag{4}$$

where $C_{l,f}$ and $C_{l,r}$ are the front and rear cornering stiffness. For both equations, the wheel slip angle α_* is defined as the angle between the orientation of the tire and the orientation of the velocity vector of the wheel (Rajamani, 2011)

$$\begin{aligned} \alpha_f &= \arctan\left(\frac{v_y + L_f\dot{\varphi}}{v_x}\right) - \delta \\ \alpha_r &= \arctan\left(\frac{v_y - L_r\dot{\varphi}}{v_x}\right) \end{aligned} \tag{5}$$

In this paper, the model with full MAGIC formulas serves as the true vehicle model, the model with simplified Pacejka Tire Model serves as the nominal model.

3. GAUSSIAN PROCESS REGRESSION

3.1. Preliminaries of Gaussian Process Regression

As defined in Rasmussen (2003), a Gaussian process is a collection of random variables, any finite number of which has a joint Gaussian distribution. For easy identification, the notation of the training data set of the GP is defined as

$$\begin{aligned} \mathcal{D} = \{\mathbf{Z} &= [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N] \in \mathbb{R}^{n_z\times N} \\ \mathbf{Y} &= \left[y_1, \ldots, y_N\right] \in \mathbb{R}^{1\times N}\} \end{aligned}$$

where n_z stands for the dimension of the input vector $\boldsymbol{z}$, N is the number of the input and output pairs $(\boldsymbol{z}_k, y_k)$. With input vector $\boldsymbol{z}_k$, each output y_k can be represented by $y_k = \boldsymbol{d}(\boldsymbol{z}_k) + \varepsilon_k$, where $\boldsymbol{d}: \mathbb{R}^{n_z} \rightarrow \mathbb{R}$ and $\varepsilon_k \sim \mathcal{N}\left(0, \sigma_\varepsilon^2\right)$ denotes Gaussian measurement noise.

Just like a Gaussian distribution is specified by its mean and variance, a Gaussian process is completely defined by mean function $\boldsymbol{m}(\boldsymbol{z})$ and a covariance function $\boldsymbol{k}(\boldsymbol{z}, \boldsymbol{z}')$.

$$\begin{aligned} \boldsymbol{m}(\boldsymbol{z}) &= \mathbb{E}[\boldsymbol{d}(\boldsymbol{z})] \\ \boldsymbol{k}\left(\boldsymbol{z}, \boldsymbol{z}'\right) &= \mathbb{E}\left[(\boldsymbol{d}(\boldsymbol{z}) - m(\boldsymbol{z}))\left(\boldsymbol{d}\left(\boldsymbol{z}'\right) - m\left(\boldsymbol{z}'\right)\right)\right] \end{aligned} \tag{6}$$

Thus Gaussian process is written as

$$\boldsymbol{d}(\boldsymbol{z}) \sim \mathcal{GP}\left(\boldsymbol{m}(\boldsymbol{z}), \boldsymbol{k}\left(\boldsymbol{z}, \boldsymbol{z}'\right)\right) \tag{7}$$

The covariance function $\boldsymbol{k}(\boldsymbol{z}, \boldsymbol{z}')$ is also known as kernel function. A squared exponential kernel is adopted in this paper.

$$\boldsymbol{k}(\boldsymbol{z}, \boldsymbol{z}') = \sigma_f^2 exp(-\frac{1}{2}(\boldsymbol{z} - \boldsymbol{z}')^{\mathrm{T}}\mathbf{M}^{-1}(\boldsymbol{z} - \boldsymbol{z}')) \tag{8}$$

where σ_f^2 and $\mathbf{M}$ are the signal variance and the diagonal matrix of squared characteristic length-scales, respectively. $\mathbf{M} = \mathrm{diag}\left(\left[\ell_1, \ldots, \ell_{n_z}\right]\right)$. Moreover, the noise variance σ_n^2 is usually to be considered, which can be added directly behind (8). These three parameters are called hyper-parameters, which are collected by parameter vector $\boldsymbol{\theta} = [\ell_1, ..., \ell_{n_z}, \sigma_f^2, \sigma_n^2]$. With predefined kernel function, we can get the prior distribution of samples. Hyper-parameters have a great influence on the performance of GP. In this paper, the Maximum Likelihood approach is adopted to obtain the optimal hyper-parameters (Rasmussen, 2003).

The posterior distribution at the test point $\boldsymbol{z}_*$ is also a Gaussian distribution with mean and variance (Williams and Rasmussen, 2006).

$$\mu^d(\boldsymbol{z}_*) = \mathbf{K}_*^\top\left[\mathbf{K} + \sigma_n^2\mathbf{I}\right]^{-1}\mathbf{Y} \tag{9}$$

$$\Sigma^d(\boldsymbol{z}_*) = \mathbf{K}_{*,*} - \mathbf{K}_*^\top\left[\mathbf{K} + \sigma_n^2\mathbf{I}\right]^{-1}\mathbf{K}_* \tag{10}$$

where d denotes the d-th dimension of the output. $\mathbf{K}$, $\mathbf{K}_*$ and $\mathbf{K}_{*,*}$ are short for $\mathbf{K}(\mathbf{Z}, \mathbf{Z})$, $\mathbf{K}\left(\mathbf{Z}, \boldsymbol{z}_*\right)$ and $\mathbf{K}\left(\boldsymbol{z}_*, \boldsymbol{z}_*\right)$, respectively. And we have $[\mathbf{K}(\mathbf{Z}, \mathbf{Z})]_{ij} = \boldsymbol{k}\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)$, $[\mathbf{K}(\mathbf{Z}, \boldsymbol{z}_*)]_j = \boldsymbol{k}\left(\boldsymbol{z}_j, \boldsymbol{z}_*\right)$ and $\mathbf{K}(\boldsymbol{z}_*, \boldsymbol{z}_*) = \boldsymbol{k}\left(\boldsymbol{z}_*, \boldsymbol{z}_*\right)$.

As discussed above, a GP regression for one dimensional output has been presented. In our paper, the output vector has n_d dimensions. The multivariate GP approximation is given by

$$\boldsymbol{d}(\boldsymbol{z}_*) \sim \mathcal{N}\left(\boldsymbol{\mu}^d(\boldsymbol{z}_*), \boldsymbol{\Sigma}^d(\boldsymbol{z}_*)\right) \tag{11}$$

where

$$\boldsymbol{\mu}^d(\boldsymbol{z}_*) = \left[\mu^1(\boldsymbol{z}_*); \ldots; \mu^{n_d}(\boldsymbol{z}_*)\right]$$

$$\boldsymbol{\Sigma}^d(\boldsymbol{z}_*) = diag(\left[\Sigma^1(\boldsymbol{z}_*); \ldots; \Sigma^{n_d}(\boldsymbol{z}_*)\right])$$

3.2. Training Data Acquisition for GP

The true vehicle model is presented as follows:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_n(\boldsymbol{x}_k, \boldsymbol{u}_k) + \mathbf{B}_d(\boldsymbol{d}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{w}_k) \tag{12}$$

where $\boldsymbol{f}_n(\boldsymbol{x}_k, \boldsymbol{u}_k)$ is a nominal function, which is the discrete model of (1). $\boldsymbol{x}_k \in \mathbb{R}^n$ denotes the state variables and $\boldsymbol{u}_k \in \mathbb{R}^m$ is the control inputs. The matrix $\mathbf{B}_d$ picks the states of system which are affected by the model error. $\boldsymbol{d}$ is the GP to capture the model mismatch and unmodeled dynamics. In our paper, we assume that the model mismatch and unmodeled dynamics, as well as the process noise $\boldsymbol{w}_k$ only affect the longitudinal velocity v_x, the lateral velocity v_y and the yaw rate w, i.e., $\mathbf{B}_d = [0; \mathbf{I}_3]$. $\boldsymbol{w}_k$ is i.i.d normally distributed process noise with $\boldsymbol{w}_k \sim \mathcal{N}(0, \boldsymbol{\Sigma}^w)$, $\boldsymbol{\Sigma}^w = \text{diag}\left[\sigma^2_{v_x}, \sigma^2_{v_y}, \sigma^2_\omega\right]$.

Since Gaussian process is a non-parametric method, the measurement data of states and inputs should be collected to infer the GP model. For a specific input data point $\boldsymbol{z}_k = [\boldsymbol{x}_k; \boldsymbol{u}_k]$, we have the training output as follows:

$$y_k = \boldsymbol{d}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{w}_k = \mathbf{B}_d^\dagger\left(\boldsymbol{x}_{k+1} - \boldsymbol{f}_n(\boldsymbol{x}_k, \boldsymbol{u}_k)\right) \tag{13}$$

where $\mathbf{B}_d^\dagger$ is the Moore-Penrose pseudo-inverse.

Then the training input and output pairs $(\boldsymbol{z}_k, y_k)$ will be used to train GP. The performance of GP relies on the training data which adds to the system. The more data we add to the model, the more precise result we can obtain. However, the increase of the size of training data will be a heavy burden to the solver, resulting in computational infeasibility over time. In order to make more use of the information of the data and take into account the computational burden, a novel matrix factorization model (Song et al., 2019) and a deep latent factor model (Wu et al., 2019b) are proposed and are proved to improve the estimation accuracy for the missing data at a little expense of the computation and storage burden. The deterministic training conditional approximation and the fully independent training conditional approximation are also applied for Gaussian process regression to overcome the computational limitations. To keep the training data size at an acceptable level, we restrict the number of actively used data points to N_{max} in our paper. Once the training data size reaches the maximum size N_{max}, some data need to be replaced. The data selection method is based on a distance measure Θ_* which has been introduced in Kabzan et al. (2019). It is defined as the posterior variance at the data point location $\boldsymbol{z}_*$, given all other data points currently in the dictionary $\mathbf{Z}_{\setminus *}$, which is shown as:

$$\Theta_* = \mathbf{K}_{\boldsymbol{z}_*,\boldsymbol{z}_*} - \mathbf{K}_{\boldsymbol{z}_*,\mathbf{Z}_{\setminus *}}(\mathbf{K}_{\mathbf{Z}_{\setminus *},\mathbf{Z}_{\setminus *}} + \sigma\mathbf{I})^{-1}\mathbf{K}_{\mathbf{Z}_{\setminus *},\boldsymbol{z}_*} \tag{14}$$

where σ is a tuning parameter. The data with the lowest distance measure should be dropped.

3.3. Approximate Uncertainty Propagation

The states and GP disturbances are approximated as jointly Gaussian distribution at each time step.

$$\begin{aligned}\left[\, \boldsymbol{x}_k^{\mathrm{T}} \ (\boldsymbol{d}_k + \boldsymbol{w}_k)^{\mathrm{T}} \,\right]^{\mathrm{T}} &\sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\ &= \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_k^x \\ \boldsymbol{\mu}_k^d \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_k^x & (\boldsymbol{\Sigma}_k^{dx})^{\mathrm{T}} \\ \boldsymbol{\Sigma}_k^{dx} & \boldsymbol{\Sigma}_k^d + \boldsymbol{\Sigma}^w \end{bmatrix}\right)\end{aligned} \tag{15}$$

where $\boldsymbol{d}_k$ represents model mismatch and unmodeled dynamics learned by GP, $\boldsymbol{\Sigma}_k^{dx}$ denotes the covariances between states and GP. To approximate the distribution of predicted states over the prediction horizon, linearization techniques related to extended Kalman filter are then derived, which allows simple update for the state mean and variance.

$$\boldsymbol{\mu}_{k+1}^x = \boldsymbol{f}_n\left(\boldsymbol{\mu}_k^x, \boldsymbol{u}_k\right) + \mathbf{B}_d\boldsymbol{\mu}_k^d \tag{16}$$

$$\boldsymbol{\Sigma}_{k+1}^x = \left[\nabla_x \boldsymbol{f}_n\left(\boldsymbol{\mu}_k^x, \boldsymbol{u}_k\right) \quad \mathbf{B}_d\right] \boldsymbol{\Sigma}_k \left[\nabla_x \boldsymbol{f}_n\left(\boldsymbol{\mu}_k^x, \boldsymbol{u}_k\right) \quad \mathbf{B}_d\right]^{\mathrm{T}} \tag{17}$$

In real scenario, the input will not always be deterministic, e.g., in the context of multi-step-ahead prediction, the last time's prediction is the input for the next iteration, which has a probability distribution. The challenge is how to propagate the resulting stochastic state distributions over the prediction horizon (Hewing et al., 2020). Assume the input itself is Gaussian, $\boldsymbol{z}_k \sim \mathcal{N}(\boldsymbol{\mu}_k^z, \boldsymbol{\Sigma}_k^z)$, the predictive distribution is stated as

$$p\left(\boldsymbol{d}(\boldsymbol{z}_k) \mid \boldsymbol{\mu}_k^z, \boldsymbol{\Sigma}_k^z\right) = \int p\left(\boldsymbol{d}(\boldsymbol{z}_k) \mid \boldsymbol{z}_k\right) p\left(\boldsymbol{z}_k \mid \boldsymbol{\mu}_k^z, \boldsymbol{\Sigma}_k^z\right) \mathrm{d}\boldsymbol{z}_k \tag{18}$$

In general, (18) is not Gaussian since a Gaussian distributions mapped through a non-linear function leads to a non-Gaussian predictive distribution (Deisenroth, 2010). Therefore, it can not be computed analytically. This issue is typically solved by approximation: Approximate (18) as a Gaussian distribution which has the same mean and variance function. Based on the criteria of computationally cheap and practical, a first order Taylor approximation method is adopted in this paper (Girard et al., 2003).

$$\boldsymbol{\mu}_k^d = \boldsymbol{u}^d\left(\boldsymbol{\mu}_k^z\right), \boldsymbol{\Sigma}_k^d = \boldsymbol{\Sigma}^d\left(\boldsymbol{\mu}_k^z\right), \boldsymbol{\Sigma}_k^{dx} = \nabla_x \boldsymbol{\mu}^d\left(\boldsymbol{\mu}_k^z\right)\boldsymbol{\Sigma}_k^x \tag{19}$$

4. GPMPC FOR VEHICLE OBSTACLE AVOIDANCE AND OVERTAKING MANEUVERS

4.1. Obstacle Avoidance and Overtaking Problems

Obstacle avoidance is one of the most difficult maneuvers for an autonomous vehicle. It combines lateral and longitudinal motion of vehicles while avoiding collisions with obstacles. In addition, other types of maneuvers such as lane-changing, lane-keeping and merging in a sequential manner should be considered (Dixit et al., 2018). Overtaking can be treated as a moving-obstacle avoidance problem. The overtaking vehicle with faster speed is called ego vehicle while the vehicle to be overtaken with lower speed called lead vehicle. **Figures 2A,B** illustrate the typical scenario of overtaking a static object and a dynamic object, respectively.

The essence of obstacle avoidance or overtaking problems are trajectory planning and trajectory tracking. The definitions of two terms have subtle differences that trajectory planning concerns more about how to generate a state trajectory, while tracking focuses on how to follow a planned trajectory. Basically, these two aspects are often studied together. In the literature, a variety of approaches have been developed for collision avoidance and planning safe trajectories to overtake the obstacles. These methods can be grouped in four categories: graph-search based methods like rapidly exploring random trees (Kuwata et al., 2008), artificial potential field based methods (Tang et al., 2010), meta-heuristic based methods (Hussein et al., 2012) and mathematical optimization based methods (Gao et al., 2010). Potential field based techniques are commonly used approaches since they have shown success in generating collision-free trajectories for overtaking (Kitazawa and Kaneko, 2016). However, they do not take the vehicle dynamics into account and hence can not ensure the reliability of the trajectories especially when the vehicle operates at high speed. Dixit et al. (2018) proposed a method that combines the potential field with MPC to overcome the absence of the vehicle model. This will turn the trajectory planning to several constraints that require to be satisfied. A new problem arises because collision avoidance constraints are typically non-convex which will lead to the local minimum instead of global minimum of optimal problems. Bengtsson (2020) introduces learning model predictive control to approximate the issue but the new approach suffers from high computational complexity.

Another method proposed by Franco and Santos (2019) is verified to be feasible to cope with obstacles avoidance. It also combines the adaptive MPC with collision avoidance methods. However, the bicycle model is only concerned with kinematics, the lateral control with regard to tire model is simplified. The proposed method of this paper is based on Franco and Santos (2019) and extended to the more accurate vehicle models combined with data-driven control strategy, making it closer to the real scenarios.

4.2. Overtaking Scenario and Overtaking Constraints

First and foremost, we need to build scenarios for overtaking problems. In this paper, we consider the case of double lane change where overtaking maneuvers are involved in it and it is also the most common cases in daily life, as shown in **Figure 2**. The road width is set to 7.5 m according to general highway standard. The solid black lines on both sides represent the road boundaries, the dashed line is the center line of the road. The ego vehicle drives from the left side to the right side and stays in the same lane all the time, unless there is an obstacle ahead that needs to overtake. There are a few lead vehicles or obstacles setting in front of the ego vehicle with constant velocity. On the contrary, the ego vehicle is given a greater degree of freedom and can adjust its speed in time according to the situation, such as accelerating when overtaking or braking when it needs to maintain a safe distance from the lead vehicles.

The objective of overtaking problems is to maximize progress on the center line of the track and avoid collisions at the same time, which is quite suitable for MPC controller. MPC controller can incorporate tracking constraints and overtaking constraints in a systematic way and make the controlled vehicle react in advance due to its long prediction horizon.

In order to avoid collisions, there are some approaches relying on a higher-level path planner (Frazzoli et al., 2005; Gray et al., 2012). However, this will lead to a rapid increase in computational complexity, which is not well-suited for real-time

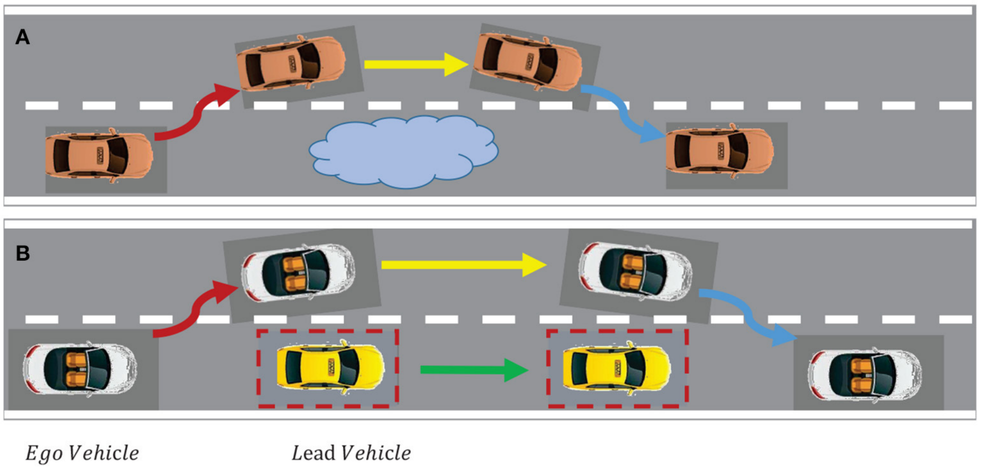

FIGURE 2 | Typical scenario of overtaking. **(A)** Overtaking a static object; **(B)** Overtaking a dynamic object.

overtaking. In our paper, we incorporate the path planning into the tracking controller by using additional constraints formed by the geometric relationship between the ego vehicle and other obstacle vehicles.

For safety overtaking maneuvers, we define an area called "safe zone" of the lead vehicle, which is a rectangle area around the lead vehicle. The safety zone is twice the length and width of the vehicle in length and width, respectively. The ego vehicle should not enter this area during the overtaking. At the next control interval the area is refreshed based on the new position of the lead vehicle. To avoid entering the area, the following state constraints are used:

$$\mathbf{A}\boldsymbol{x} \leq \mathbf{B}$$

where $\boldsymbol{x} = [X; Y; \Phi; v_x; v_y; \omega]$ is the state vector, while **A** and **B** are the constraint matrices that can be updated when the controller is in operation. Since the overtaking maneuvers are mainly related to the longitudinal and lateral motion, the constraints have effect on the position of the ego vehicle (X, Y). The matrices **A** and **B** are defined as:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ k & -1 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} L_1 \\ L_2 \\ -b \end{bmatrix} \tag{20}$$

where k is the slope of the line formed from the c.g. of the ego vehicle to safe zone corner. Obviously, b is the intercept. $L_{1,2}$ represent the upper bound and lower bound on the Y coordinate. The coordinate system of the entire road is established with the origin point at the middle point of the left side of the road. **Figure 3** takes the left overtaking as an example, two vehicles are on the lower lane and the ego vehicle will detect the lead vehicle when the distance is less than 20 m. The dashed orange area is the accessible area when overtaking happens, while the dashed red rectangle boundary is the safe zone of the lead vehicle. To ensure safety, an extra safe lateral distance is added, which is set to half the width of the vehicle in this paper, as shown in **Figure 3**.

Remark: When the ego vehicle needs to overtake, the choice of left overtaking or right overtaking can be determined simply according to the position of the front vehicle in the coordinate system of the ego vehicle. If the Y coordinate of the front vehicle is negative, then the ego vehicle will choose left overtaking, otherwise choose right overtaking.

4.3. Contouring Control and Resulting Cost Function

GPMPC controller takes advantage of a contouring control framework, which follows a similar strategy as used in Lam et al. (2010). We modify the specific formulation of the model predictive contouring control to maximize the traveled length on the reference path. Therefore, the center line of a certain lane is chosen as the reference tracking path, but it is employed solely as a measure of progress. The reference path is parameterized by its arc length ξ using third order spline polynomials. Then given an exemplary ξ, the centerline position $[X_c(\xi), Y_c(\xi)]$ and orientation $\Phi_c(\xi)$ and the track radius $R_c(\xi)$ of vehicle can be evaluated by interpolation. As a result, the cost function is defined by the so-called lag error e_l, contour error e_c, orientation error e_o and offset error e_{off}, as illustrated in **Figure 4**. The definition of

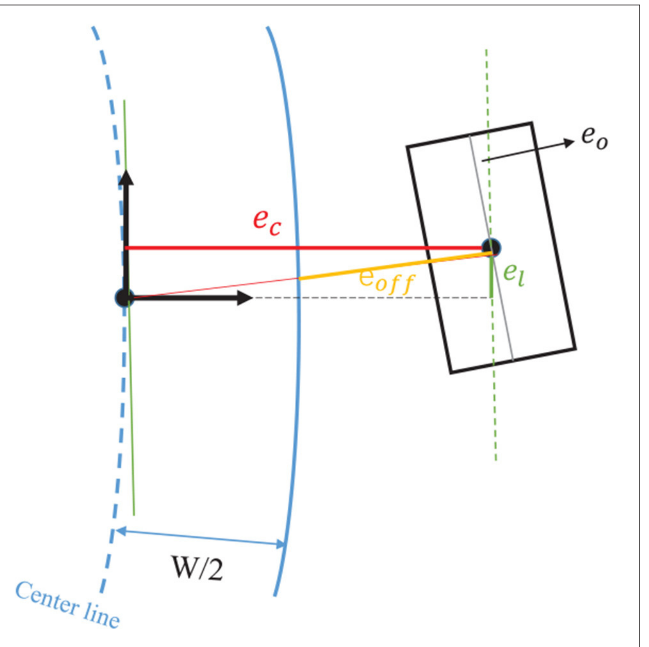

FIGURE 4 | Lag-, contour-, orientation-, and offset error. The vehicle model is intentionally plotted outside the road boundary to show these errors clearly.

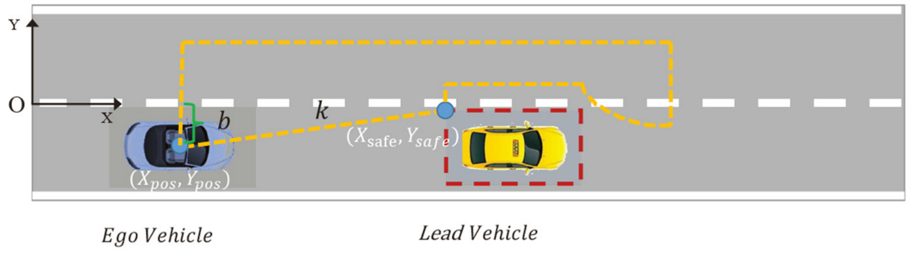

FIGURE 3 | Schematic of accessible driving area in the case of left overtaking.

these parameters can refer to Liniger et al. (2015).

$$
\begin{aligned}
e_l\left(\boldsymbol{u}_k^x, \xi_k\right) &= \cos\left(\Phi\left(\xi_k\right)\right)\left(X_c\left(\xi_k\right) - X_k\right) \\
&\quad + \sin\left(\Phi\left(\xi_k\right)\right)\left(Y_c\left(\xi_k\right) - Y_k\right) \\
e_c\left(\boldsymbol{u}_k^x, \xi_k\right) &= -\sin\left(\Phi\left(\xi_k\right)\right)\left(X_c\left(\xi_k\right) - X_k\right) \\
&\quad + \cos\left(\Phi\left(\xi_k\right)\right)\left(Y_c\left(\xi_k\right) - Y_k\right) \\
e_o\left(\boldsymbol{u}_k^x, \xi_k\right) &= 1 - \left|\cos\left(\Phi\left(\xi_k\right)\right)\cos(\varphi) + \sin\left(\Phi\left(\xi_k\right)\right)\sin(\varphi)\right| \\
e_{off}\left(\boldsymbol{u}_k^x, \xi_k\right) &= \frac{1}{R_c\left(\xi_k\right)}\sqrt{e_l\left(\boldsymbol{u}_k^x, \xi_k\right)^2 + e_c\left(\boldsymbol{u}_k^x, \xi_k\right)^2} - 1
\end{aligned}
\tag{21}
$$

The MPC formulation can be made more efficient by removing constraints. However, to keep the vehicle staying inside the boundary of road, there must be vehicle state constraint. In this paper, we transform the traditional hard state constraint into a soft constraint and incorporate it as relaxed barrier function $\mathcal{R}_b(e_{off})$ into the cost function, which will improve the optimizer performance. The relaxed barrier function is defined as:

$$
\mathcal{R}_b(e_{off}) = \beta\left(\sqrt{\frac{\left(c + \gamma(\lambda - e_{off})^2\right)}{\gamma}} - (\lambda - e_{off})\right) \tag{22}
$$

where β, γ, λ, and c are constant parameters. The stage cost function is then written as:

$$
\begin{aligned}
l\left(\boldsymbol{u}_k^x, \xi_k\right) &= \left\|e_c\left(\boldsymbol{u}_k^x, \xi_k\right)\right\|_{q_c}^2 + \left\|e_l\left(\boldsymbol{u}_k^x, \xi_k\right)\right\|_{q_l}^2 \\
&\quad + \left\|e_o\left(\boldsymbol{u}_k^x, \xi_k\right)\right\|_{q_o}^2 + \left\|\mathcal{R}_b(e_{off}\left(\boldsymbol{u}_k^x, \xi_k\right))\right\|_{q_{off}}^2
\end{aligned}
\tag{23}
$$

where q_c, q_l, q_o and q_{off} are the corresponding weights.

4.4. Input Constraints and Resulting Formulation

The input vector constraints $\mathcal{U}$ are limited as below:

$$
\begin{bmatrix} -\delta_{\max} \\ -T_{\max} \end{bmatrix} \leq \begin{bmatrix} \delta_k \\ T_k \end{bmatrix} \leq \begin{bmatrix} \delta_{\max} \\ T_{\max} \end{bmatrix} \tag{24}
$$

Based on this contouring formulation, we integrate a stochastic GPMPC model which results in minimizing the cost function (23) over a finite horizon of length N_p. The corresponding GPMPC formulation with tractable approximation is defined as follows:

$$
\min_{\boldsymbol{u}_k} \quad J\left(\boldsymbol{\mu}_k^x, \xi_k\right) = \sum_{k=0}^{N-1} l\left(\boldsymbol{\mu}_k^x, \xi_k\right) \tag{25a}
$$

$$
s.t. \quad \boldsymbol{u}_{k+1}^x = \boldsymbol{f}_n\left(\boldsymbol{u}_k^x, \boldsymbol{u}_k\right) + \mathbf{B}_d\left(\boldsymbol{d}\left(\boldsymbol{u}_k^x, \boldsymbol{u}_k\right) + \boldsymbol{w}_k\right) \tag{25b}
$$

$$
\mathbf{A}\boldsymbol{\mu}_{k+1}^x \leq \mathbf{B} \tag{25c}
$$

$$
\boldsymbol{u}_k \in \mathcal{U} \tag{25d}
$$

$$
\boldsymbol{\mu}_0^x = \boldsymbol{x}(k), \boldsymbol{\Sigma}_0^x = 0, \xi_0 = \xi(k) \tag{25e}
$$

5. SIMULATION AND ANALYSIS

In order to verify the effectiveness of the proposed approach, two overtaking scenarios on a two-lane road are constructed. In the first scenario, we require the ego vehicle to drive in the right lane, unless there is an obstacle ahead that needs to overtake. We call this scenario as left overtaking. The initial position of the ego vehicle is set to $(0, -1.875)$ with an initial speed 20 m/s. Lead vehicle 1 (in red) starts from $(25, -1.875)$ with a constant velocity 12 m/s, while lead vehicle 2 (in blue) is at point $(60, -1.875)$ with a constant velocity 10 m/s. In the second scenario, we require the ego vehicle to drive in the left lane, unless there is an obstacle ahead that needs to overtake. We call this scenario as right overtaking. The initial position of the ego vehicle is set to $(2, 1.875)$ with an initial speed 20 m/s. There are three vehicles in front of the ego vehicle. The lead vehicle 1 brokes down at point (25, 1.875) and stops here (in red). The lead vehicle 2 is driving forward at (45, 1.875) at a constant velocity 10 m/s (in green). The lead vehicle 3 is driving forward at (75, 1.875) with a constant velocity 8 m/s (in blue). Please note that during the experiment time, lead vehicles will not collide.

The NMPC algorithm is used for comparison. The details of NMPC algorithm can refer to Liniger et al. (2015). The GPMPC problem in (25) is implemented with a prediction horizon of $N_p = 10$. The sampling time is $T_s = 50$ ms, resulting in a 0.5 s look-ahead. The maximum number of iterations is limited to 30 to ensure consistent maximum solve times. Considering the reality, we limit the ego vehicle speed between 10 to 35 m/s. The vehicles and obstacles are abstracted to small blocks with 4 m long and 1.6 m wide. For easy distinction, vehicles are colored. The ego vehicle is black, the obstacles and lead vehicles are depicted in red, green, or blue. The parameters of the ego vehicle is shown in **Table 1**. For the full MAGIC formulas, B_f is 0.4, C_f is 8, D_f is 4560.4, E_f is -0.5, B_r is 0.45, C_r is 8, D_r is 4,000, E_r is -0.5.

Nominal vehicle model $\boldsymbol{f}_n\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right)$ and true vehicle model (12) are prepared for calculating the deviations to be learned by GP model. At first, the ego vehicle starts at the initial point with NMPC controller, meaning that all GP-dependent variables are set to zero. The corresponding parameters are tuned slightly to prevent crashes and driving off the road. Since the nominal model does not consider the model mismatch and unmodeled dynamics, it is allowed that its driving behavior is somewhat erratic and there are small collisions with road boundaries. Throughout the first run, the data are collected to fill the training dictionary and train the GP error model $\boldsymbol{d}$. In the next run, GP is activated, using the knowledge from the last run and also accumulating new data from the current run. The GP model was first generated with fixed hyperparameters, but we can infer the hyperparameters by using maximum likelihood optimization according these collected data

TABLE 1 | Parameters of the ego vehicle.

$M(kg)$	$I_z(kg \cdot m^2)$	$L_f(m)$	$L_r(m)$	$C_{l,f}(N/rad)$	$C_{l,r}(N/rad)$
500	600	0.9	1.5	1,400	1,400

(Rasmussen, 2003). After that we activate the GPMPC with loaded data and optimized hyperparameters. The new training data will be added into the GP model after each iteration. When the maximal dictionary size is reached, some data points will be discarded by using method mentioned in (14).

The parameters of the NPMC controller is shown in **Table 2**. We manually adjust these parameters again and again based on a large number of simulation experiments. For GPMPC controller, its parameters in the MPC part are the same as those of NPMC. The hyper-parameters of GPMPC controller are shown in **Table 3**, where $\mathbf{M}_1$, $\sigma^2_{v_x} = 7.1304e-4$ and $\sigma_{f1} = 2.8052e-11$ are the hyper-parameters for v_x dimension, $\mathbf{M}_2$, $\sigma^2_{v_y} = 1.0358e-10$, and $\sigma_{f2} = 0.0236$ are the hyper-parameters for v_y dimension, $\mathbf{M}_3$, $\sigma^2_{\omega} = 1.0059e-10$ and $\sigma_{f3} = 0.0117$ are the hyper-parameters for ω dimension.

To quantify the performance of the GPMPC control scheme and the improvement due to the learning, we compare the predicted model error in v_x, v_y, and ω, calculated by nominal model with NMPC controller and estimated model with GPMPC controller, respectively.

TABLE 2 | Parameters of the NPMC controller.

q_c	q_l	q_o	q_{off}	β	c	γ	λ	u_{max}
20	50	20	180	5	4	1,000	−0.1	$[0.3419 \quad 1]^T$

TABLE 3 | Hyper-parameters for GPMPC controller.

Parameter	Value
M_1	diag(0.0346,0.0151,0.0148,0.0153,0.0163,0.0156,0.0148,0.016)
M_2	diag(9.9184e4,6.94995e4,731,1988,15,6.2355e4,0.12,1098)
M_3	diag(9.9829e4,9.6999e4,1.199e4,2131,77,12,0.51,982)

Figures 5, **6** illustrate that GPMPC performs much better than NMPC. In order to see the capability of GP learning model clearly, the mean squared error (MSE) of the tracking error in each dynamic state is shown in **Tables 4**, **5**.

Where $\|\boldsymbol{e}_{NMPC}\| = \left\|\boldsymbol{x}_{k+1} - \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)\right\|$, $\|\boldsymbol{e}_{GPMPC}\| = \left\|\boldsymbol{x}_{k+1} - \left(\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \mathbf{B}_d \boldsymbol{\mu}^d(\boldsymbol{z}_k)\right)\right\|$.

In addition, we investigate the controller performance by plotting the predicted values in one iteration. In each time step, both controllers will make predictions for 10 steps ahead as shown in **Figures 7–12** for left overtaking and in **Figures 13–18** for right overtaking. Please note that we did not give a fixed reference trajectory for NMPC and GPMPC controller. The optimized reference trajectory for each prediction is calculated online. Therefore, GPMPC and NMPC have different reference trajectories. We investigated the controller performance by plotting the predicted values in one iteration at the same starting time.

As evident in **Figures 7–18**, the NMPC controller performs visually suboptimally and is unable to predict the future evolution in some cases. On the contrary, we can see that the GPMPC controller matches the real values quite well in most cases. The uncertainty in form of a 2σ confidence interval is shown in light gray. With uncertainty propagation, we observe that the majority of predictive states during overtaking are still anticipated by the GP-uncertainty.

Evolution of control inputs throughout the whole simulation is another important index for controller performances. **Figures 19**, **20** show applied control inputs to left overtaking problem and right overtaking problem, respectively. The upper figures represent the change of the first control variable: steering angle δ. The green dashed line is GPMPC. During the time interval $1.5-2$ s and $3-3.5$ s in **Figure 19A**, δ changed more rapidly than in NMPC, which means GPMPC consumed a lot to achieve a steady state. But after oscillation, δ stabilized quickly. During the time interval $2.9-3.3$ s in **Figure 20A**, δ changed

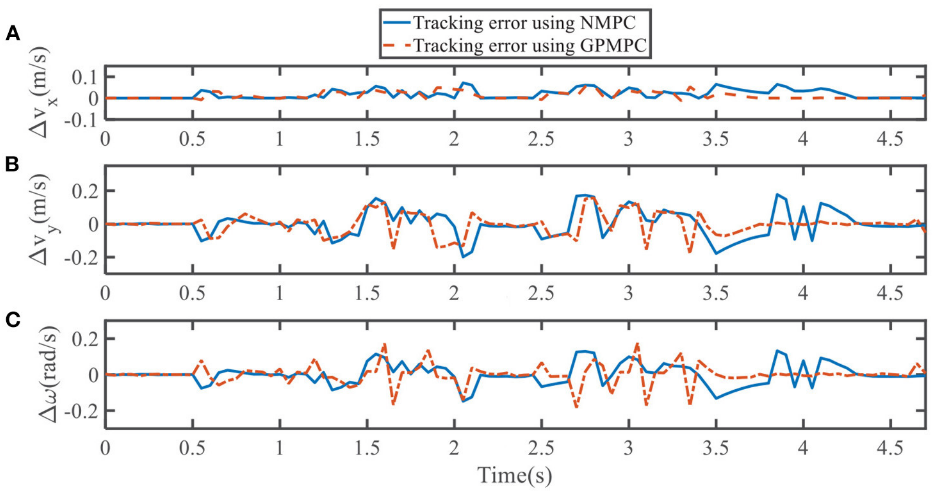

FIGURE 5 | Tracking error using NMPC and GPMPC in left overtaking. **(A)** Tracking error v_x; **(B)** Tracking error v_y; **(C)** Tracking error ω.

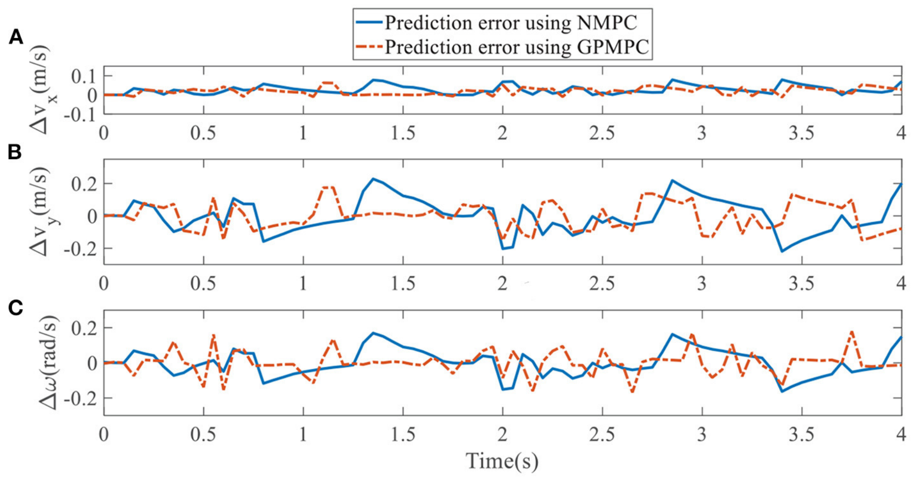

FIGURE 6 | Tracking error using NMPC and GPMPC in right overtaking. **(A)** Tracking error v_x; **(B)** Tracking error v_y; **(C)** Tracking error ω.

TABLE 4 | MSE comparison in left overtaking.

Model	$\overline{\|e_{v_x}\|}$	$\overline{\|e_{v_y}\|}$	$\overline{\|e_{\omega}\|}$	$\overline{\|e\|}$
NMPC	0.2700	0.7684	0.5693	0.9565
GPMPC	0.2025	0.6494	0.5659	0.8000

TABLE 5 | MSE comparison in right overtaking.

Model	$\overline{\|e_{v_x}\|}$	$\overline{\|e_{v_y}\|}$	$\overline{\|e_{\omega}\|}$	$\overline{\|e\|}$
NMPC	0.3042	0.8792	0.6501	1.0936
GPMPC	0.2136	0.6622	0.5260	0.7755

more rapidly than in NMPC, which is related to the ego vehicle to stay on the current lane and avoid overtaking early.

When it comes to the control variable T, GPMPC performs much better than NMPC. Since $T = 1$ represents full accelerating and $T = -1$ means full braking, **Figures 19B**, **20B** show that NMPC controller shifts extremely steep to avoid constraint violation. This limitation is not present in the GPMPC approach, where a more precisely prediction has made. Therefore, except a few points where the vehicle first detect the obstacle, GPMPC controls fairly smooth comparing to the NMPC controller, which gives a speed benefit and consumes less power.

For autonomous overtaking scenario, taking driven trajectories as a performance criterion is the most intuitive way. Therefore, we investigate two controller performances by comparing the overtaking maneuvers and the overall driven trajectories. **Figure 21** illustrates the cases in left overtaking while **Figure 22** is for right overtaking. **Figures 21A,C** are the driven trajectories with velocity profile generated by NMPC and GPMPC, respectively. The maneuvers where the ego vehicle is overtaking the first vehicle are shown in **Figures 21B,D**. In **Figure 22**, **Figures 22a,c,e,g** are for NMPC. **Figures 22b,d,f,h** are for GPMPC. The maneuvers where the ego vehicle is overtaking the first vehicle are shown in **Figures 22a,b**. The maneuvers where the ego vehicle is overtaking the second vehicle are shown in **Figures 22c,d**. The maneuvers where the ego vehicle is overtaking the third vehicle are shown in **Figures 22e,f**. **Figures 22g,h** are the driven trajectories with velocity profile generated by NMPC and GPMPC, respectively. Both control strategies are able to accomplish the overtaking mission without collisions, which proves that the parameters of MPC controller are valid. However, it is evident to see GPMPC outperforms NMPC, especially with regard to constraint satisfaction. This can be seen from the black dotted circle in **Figure 21B** and the blue dotted circle in **Figures 22c,e**. The ego vehicle hits the "safe zone." Although the "safe zone," depicted by dashed red lines, is virtual in real world, driving too close to the overtaken vehicles will indeed increase the risk of collision.

Furthermore, there is another phenomenon worth mentioning. It can be seen from NMPC's trajectories in left overtaking that there is a wave crest after overtaking the second vehicle, which is clearly visible from the blue dotted circle in **Figure 21A**. And there are two more wave crests in NMPC than in GPMPC for right overtaking, which is clearly visible from **Figure 22g**. The extra displacement of trajectories are generated due to the wrongly estimation of lateral force by NMPC, which leads the vehicle return to the original track prematurely when it not fully finishes overtaking. Then, since the lead vehicle is moving, at the next time step, situation gets not suitable for overtaking. The ego vehicle has to overtake the obstacle vehicle again. Comparing to NMPC, the resulting trajectories with GPMPC are displayed in **Figures 21C**, **22g**, generally showing a much more smooth and safe overtaking behavior. In particular, almost all of the problems in the trajectories of the nominal model and NMPC controller can be alleviated.

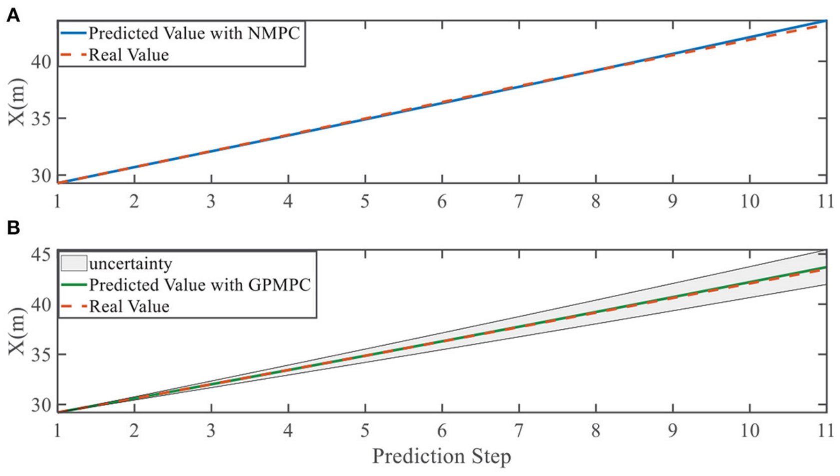

FIGURE 7 | Position X evolution in left overtaking. **(A)** NMPC; **(B)** GPMPC.

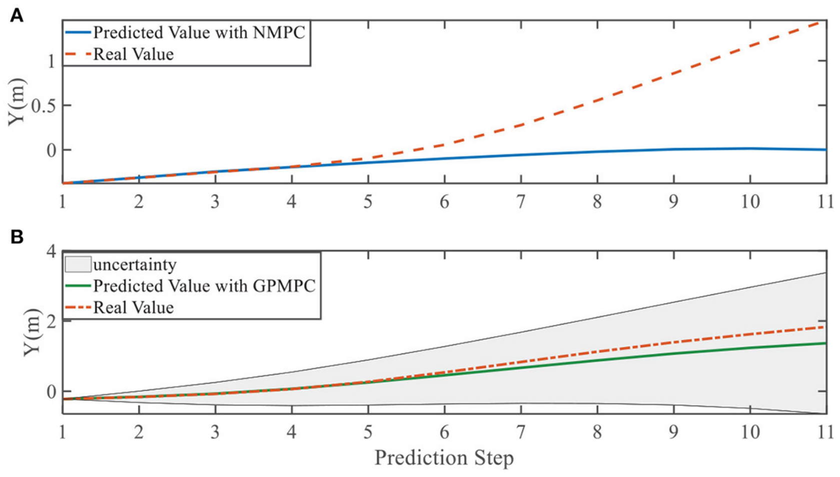

FIGURE 8 | Position Y evolution in left overtaking. **(A)** NMPC; **(B)** GPMPC.

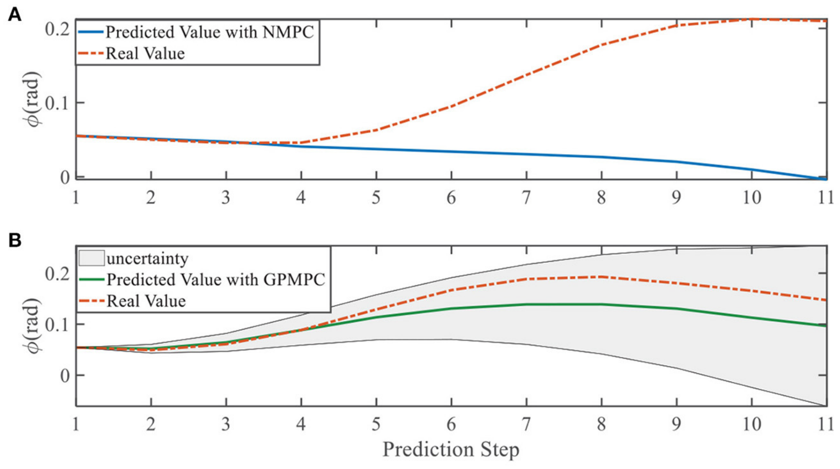

FIGURE 9 | Yaw angle evolution in left overtaking. **(A)** NMPC; **(B)** GPMPC.

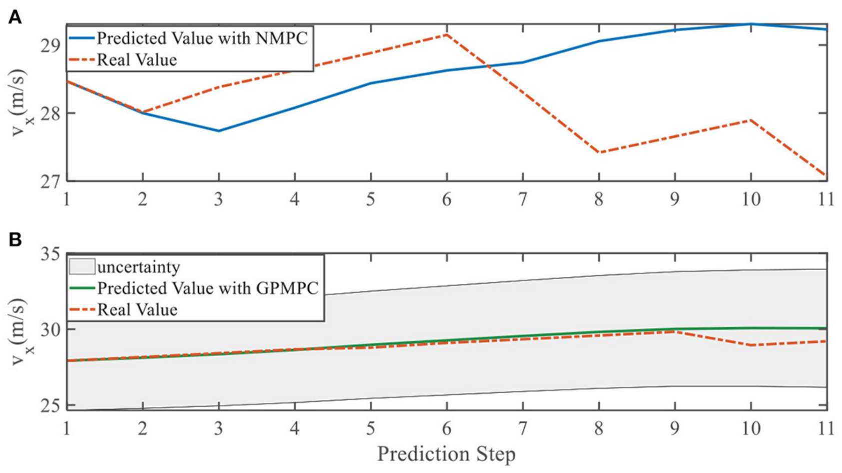

FIGURE 10 | Velocity v_x evolution in left overtaking. **(A)** NMPC; **(B)** GPMPC.

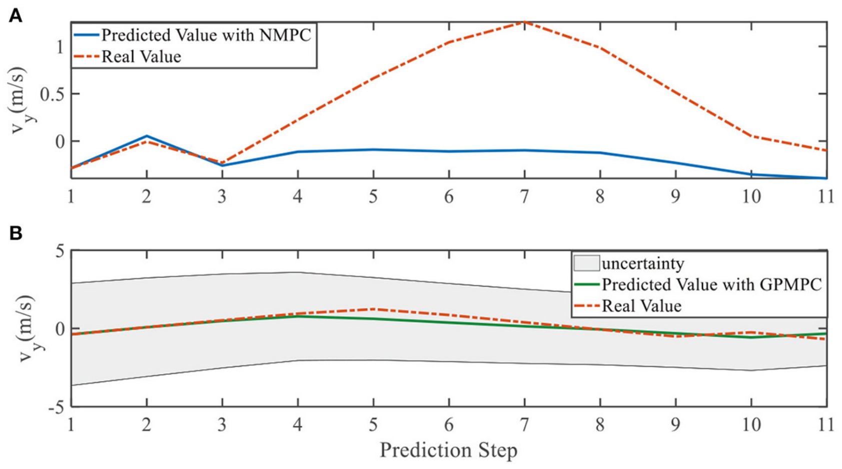

FIGURE 11 | Velocity v_y evolution in left overtaking. **(A)** NMPC; **(B)** GPMPC.

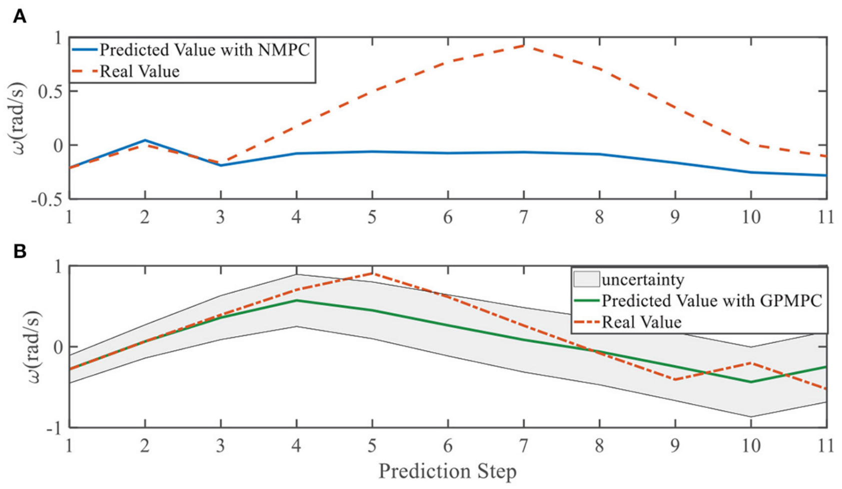

FIGURE 12 | Yaw rate evolution in left overtaking. **(A)** NMPC; **(B)** GPMPC.

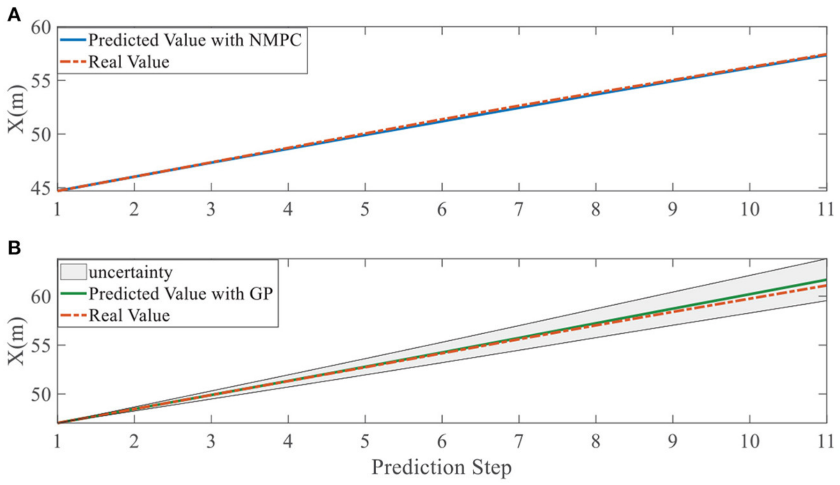

FIGURE 13 | Position X evolution in right overtaking. **(A)** NMPC; **(B)** GPMPC.

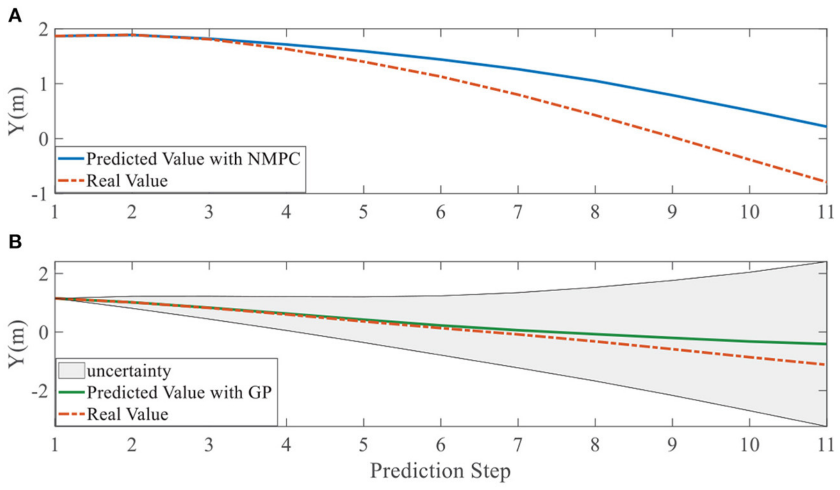

FIGURE 14 | Position Y evolution in right overtaking. **(A)** NMPC; **(B)** GPMPC.

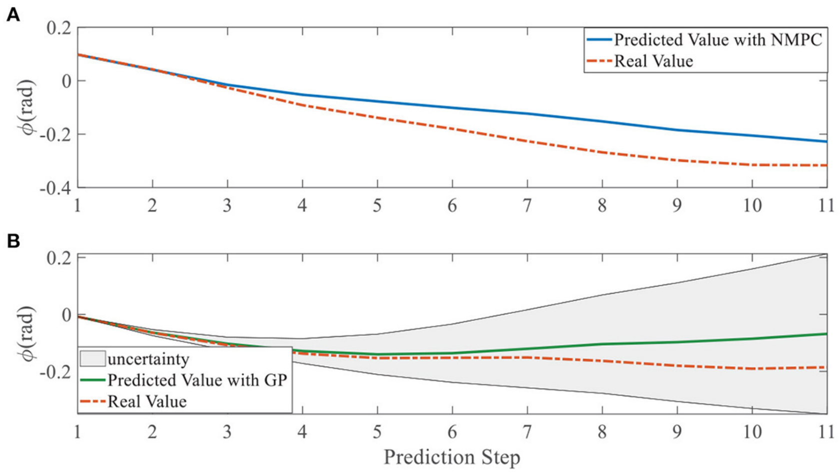

FIGURE 15 | Yaw angle evolution in right overtaking. **(A)** NMPC; **(B)** GPMPC.

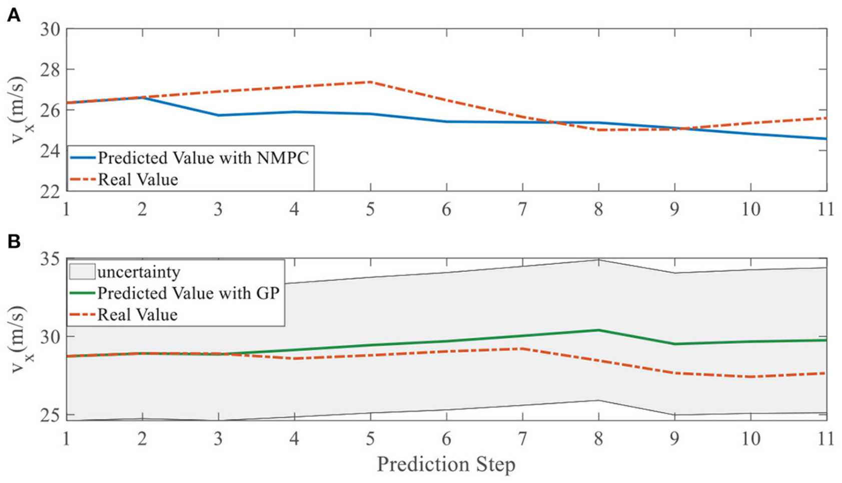

FIGURE 16 | Velocity v_x evolution in right overtaking. **(A)** NMPC; **(B)** GPMPC.

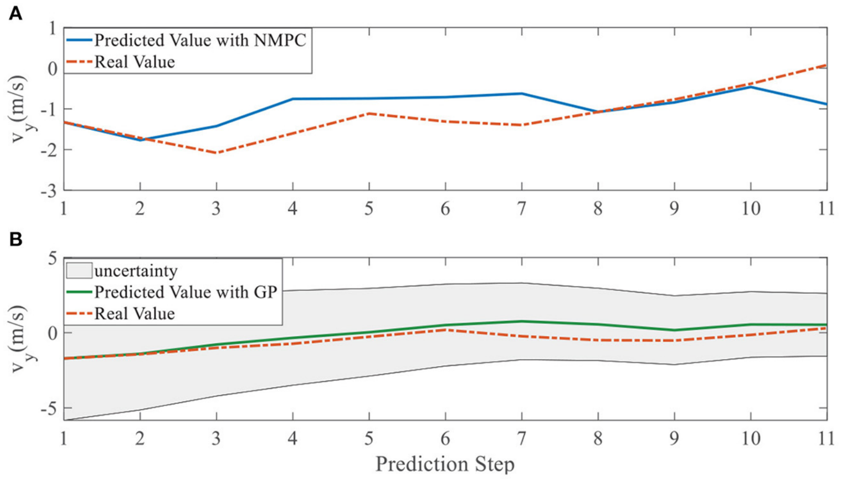

FIGURE 17 | Velocity v_y evolution in right overtaking. **(A)** NMPC; **(B)** GPMPC.

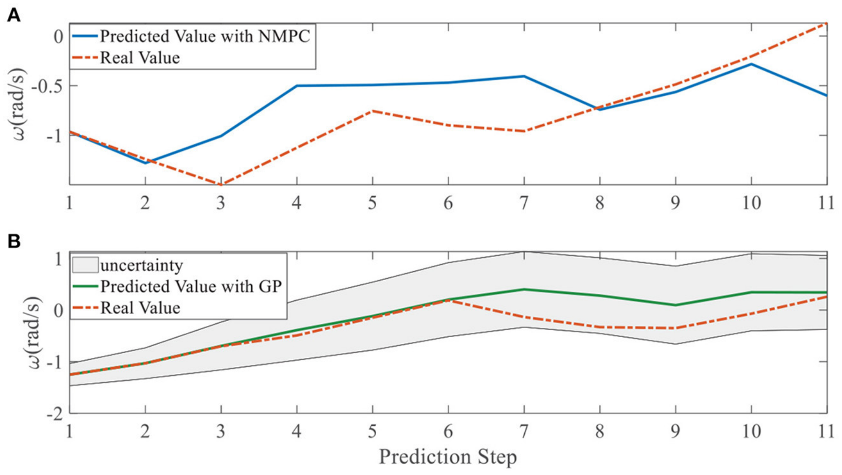

FIGURE 18 | Yaw rate evolution in right overtaking. **(A)** NMPC; **(B)** GPMPC.

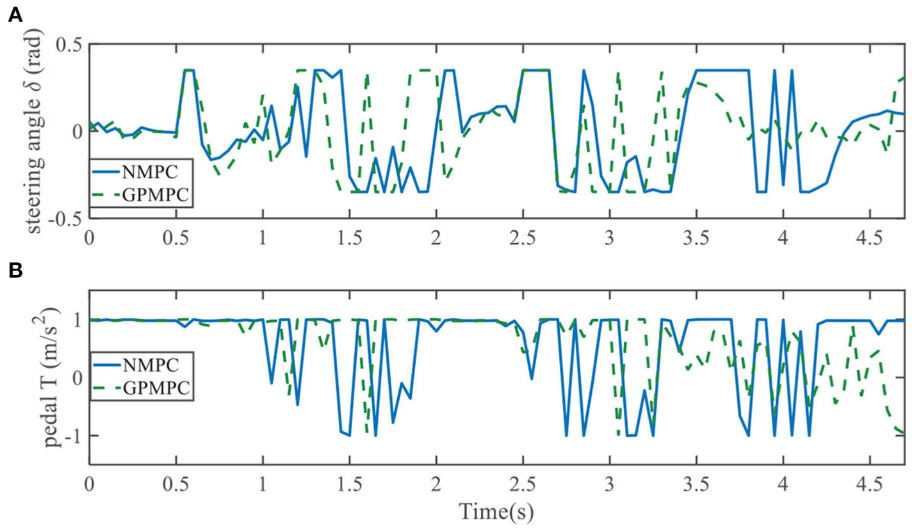

FIGURE 19 | Control inputs using NMPC and GPMPC for left overtaking. **(A)** Steering angle; **(B)** Pedal.

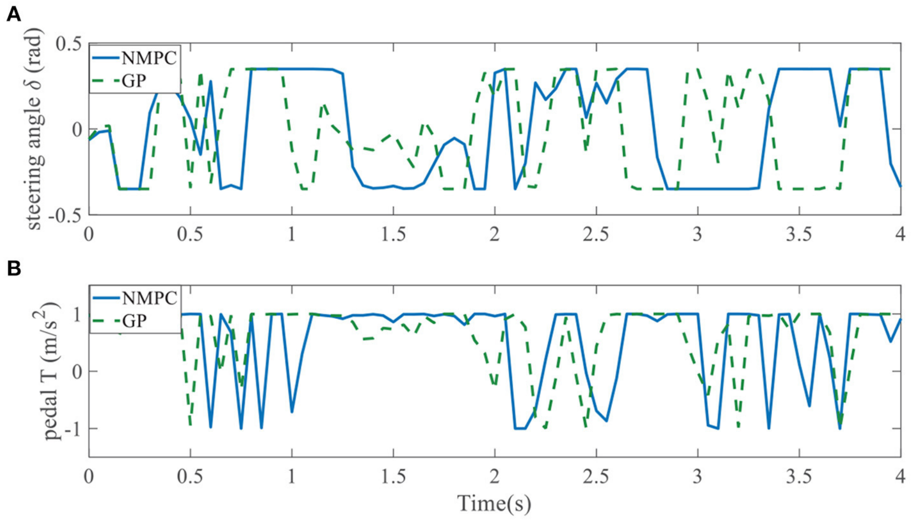

FIGURE 20 | Control inputs using NMPC and GPMPC for right overtaking. **(A)** Steering angle; **(B)** Pedal.

6. CONCLUSION

We have investigated overtaking problems in autonomous driving and dedicate to build a GP-based control framework which is able to complete vehicle control, trajectory tracking and obstacles avoidance. Since the vehicle model is a extremely complicated system and the road condition is time-varying, it is intractable to derive a precise model. Thus, the learning based method is introduced and the core concept of this method is only using a nominal model to represent the vehicle while the rest uncertainties, disturbances and mismatch can be learned by GP model. However, one issue raised during the combination of GP regression and traditional NMPC controller: the MPC became a stochastic formulation because of the GP approximation. By employing the Taylor approximation we can propagate the uncertainties and evaluate the residual uncertainties, which increases the accurateness and the controller. The implemented Taylor approximation depends directly on the dimension of training data. As the data points constantly adding into the model, it becomes expensive to evaluate in high dimensional spaces. We limit the upper bound of the number of data points with a dictionary and set a selection mechanism, thus the computational complexity will be sustained on a medium level. In addition, we modify the constraints and cost function

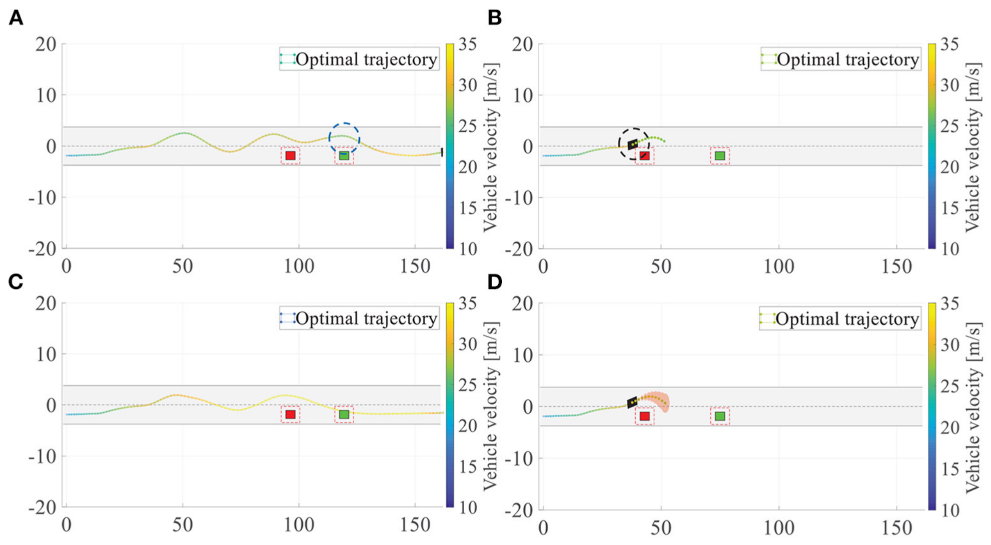

FIGURE 21 | Overtaking maneuvers and the overall driven trajectories for left overtaking. **(A)** Overall driven trajectories of NMPC; **(B)** Overtaking maneuver of NMPC; **(C)** Overall driven trajectories of GPMPC; **(D)** Overtaking maneuver of GPMPC.

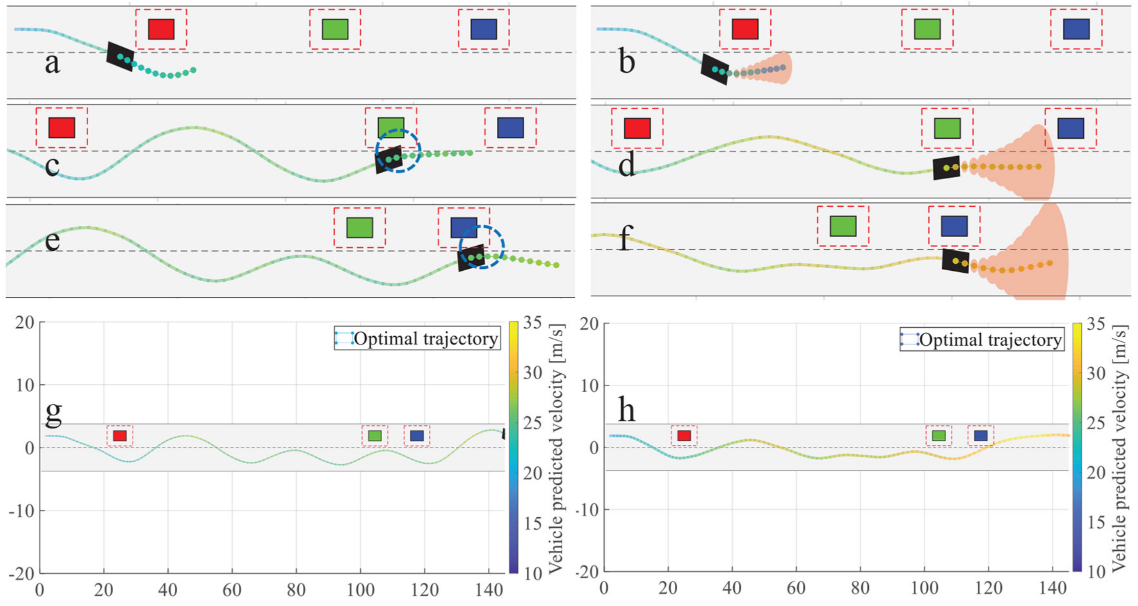

FIGURE 22 | Overtaking maneuvers and the overall driven trajectories. **(a)** Overtaking the first vehicle by NMPC; **(b)** Overtaking the first vehicle by GPMPC; **(c)** Overtaking the second vehicle by NMPC; **(d)** Overtaking the second vehicle by GPMPC; **(e)** Overtaking the third vehicle by NMPC; **(f)** Overtaking the third vehicle by GPMPC **(g)** Overall driven trajectories of NMPC; **(h)** Overall driven trajectories of GPMPC.

to reduce the computation need for optimization. Collectively, simulation results demonstrate that both performance and safety in overtaking can be improved by using GPMPC.

There are two suitable directions for future work on this topic. Firstly, the GP model was trained online but the hyperparameters were selected offline, which means the trained GP remains constant throughout the prediction online progress. However, the velocity of vehicle is constantly changing, making the optimized parameters not suitable for the GP model. Train the hyperparameters online and generalize the model to various overtaking scenarios deserve further investigation. Secondly, overtaking at a corner is a special case in autonomous driving which deserves deep investigation. The proposed overtaking method is able to handle nearly all situations on the straight road but when it comes to the corner, things get beyond its competence.

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

AUTHOR CONTRIBUTIONS

WL: methodology, software, data curation, and writing original draft. CL: software and data curation. GC: methodology and data curation. AK: conceptualization, supervision, and writing—review. All authors contributed to the article and approved the submitted version.

FUNDING

This work was financially supported by the German Research Foundation (DFG) and the Technical University of Munich (TUM) in the framework of the Open Access Publishing Program.

REFERENCES

Bengtsson, I. (2020). *Optimization and systems theory* (Master thesis).

Cha, E. S., Kim, K.-E., Longo, S., and Mehta, A. (2018). "OP-CAS: collision avoidance with overtaking maneuvers," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (Maui, HI: IEEE), 1715–1720. doi: 10.1109/ITSC.2018.8569740

Chen, T., Cai, Y., Chen, L., Xu, X., and Sun, X. (2021). Trajectory tracking control of steer-by-wire autonomous ground vehicle considering the complete failure of vehicle steering motor. *Simul. Modell. Pract. Theory* 109:102235. doi: 10.1016/j.simpat.2020.102235

Deisenroth, M. P. (2010). *Efficient Reinforcement Learning Using Gaussian Processes*, Vol. 9. karlsruhe: KIT Scientific Publishing.

Dixit, S., Montanaro, U., Fallah, S., Dianati, M., Oxtoby, D., Mizutani, T., et al. (2018). "Trajectory planning for autonomous high-speed overtaking using mpc with terminal set constraints," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (Maui, HI: IEEE), 1061–1068. doi: 10.1109/ITSC.2018.8569529

Elbanhawi, M., Simic, M., and Jazar, R. (2018). Receding horizon lateral vehicle control for pure pursuit path tracking. *J. Vibrat. Control* 24, 619–642. doi: 10.1177/1077546316646906

Franco, A., and Santos, V. (2019). "Short-term path planning with multiple moving obstacle avoidance based on adaptive MPC," in *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)* (Porto: IEEE), 1–7. doi: 10.1109/ICARSC.2019.8733653

Frazzoli, E., Dahleh, M. A., and Feron, E. (2005). Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Trans. Robot.* 21, 1077–1091. doi: 10.1109/TRO.2005.852260

Gao, Y., Lin, T., Borrelli, F., Tseng, E., and Hrovat, D. (2010). Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads. *Dyn. Syst. Control Conf.* 44175, 265–272. doi: 10.1115/DSCC2010-4263

Girard, A., Rasmussen, C. E., Candela, J. Q., and Murray-Smith, R. (2003). "Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting," in *Advances in Neural Information Processing Systems* (Vancouver, BC: MIT Press), 545–552.

Gray, A., Gao, Y., Lin, T., Hedrick, J. K., Tseng, H. E., and Borrelli, F. (2012). "Predictive control for agile semi-autonomous ground vehicles using motion primitives," in *2012 American Control Conference (ACC)* (Montreal, QC: IEEE), 4239–4244. doi: 10.1109/ACC.2012.6315303

Hewing, L., Kabzan, J., and Zeilinger, M. N. (2019). Cautious model predictive control using gaussian process regression. *IEEE Trans. Control Syst. Technol.* 2736–2743. doi: 10.1109/TCST.2019.2949757

Hewing, L., Liniger, A., and Zeilinger, M. N. (2018). "Cautious NMPC with gaussian process dynamics for autonomous miniature race cars," in *2018 European Control Conference (ECC)* (Limassol: IEEE), 1341–1348. doi: 10.23919/ECC.2018.8550162

Hewing, L., Wabersich, K. P., Menner, M., and Zeilinger, M. N. (2020). Learning-based model predictive control: toward safe learning in control. *Annu. Rev. Control Robot. Auton. Syst.* 3, 269–296. doi: 10.1146/annurev-control-090419-075625

Hussein, A., Mostafa, H., Badrel-din, M., Sultan, O., and Khamis, A. (2012). "Metaheuristic optimization approach to mobile robot path planning," in *2012 International Conference on Engineering and Technology (ICET)* (Cairo: IEEE), 1–6. doi: 10.1109/ICEngTechnol.2012.6396150

Kabzan, J., Hewing, L., Liniger, A., and Zeilinger, M. N. (2019). Learning-based model predictive control for autonomous racing. *IEEE Robot. Automat. Lett.* 4, 3363–3370. doi: 10.1109/LRA.2019.2926677

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., et al. (2021). Deep reinforcement learning for autonomous driving: a survey. *IEEE Trans. Intell. Transport. Syst.* 1–18. doi: 10.1109/TITS.2021.3054625

Kitazawa, S., and Kaneko, T. (2016). "Control target algorithm for direction control of autonomous vehicles in consideration of mutual accordance in mixed traffic conditions," in *Proc. 13th Int. Symp. Adv. Vehicle Control* (London), 151. doi: 10.1201/9781315265285-25

Kuwata, Y., Fiore, G. A., Teo, J., Frazzoli, E., and How, J. P. (2008). "Motion planning for urban driving using RRT," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Nice: IEEE), 1681–1686. doi: 10.1109/IROS.2008.4651075

Lam, D., Manzie, C., and Good, M. (2010). "Model predictive contouring control," in *49th IEEE Conference on Decision and Control (CDC)* (Atlanta, GA: IEEE), 6137–6142. doi: 10.1109/CDC.2010.5717042

Langåker, H.-A. (2018). *Cautious MPC-based control with machine learning* (Master's thesis). NTNU, Trondheim, Norway.

Lattarulo, R., He, D., and Perez, J. (2018). "A linear model predictive planning approach for overtaking manoeuvres under possible collision circumstances," in *2018 IEEE Intelligent Vehicles Symposium (IV)* (Changshu: IEEE), 1340–1345. doi: 10.1109/IVS.2018.8500542

Liniger, A., Domahidi, A., and Morari, M. (2015). Optimization-based autonomous racing of 1: 43 scale RC cars. *Optimal Control Appl. Methods* 36, 628–647. doi: 10.1002/oca.2123

McKinnon, C. D., and Schoellig, A. P. (2019). Learn fast, forget slow: safe predictive learning control for systems with unknown and changing dynamics performing repetitive tasks. *IEEE Robot. Automat. Lett.* 4, 2180–2187. doi: 10.1109/LRA.2019.2901638

Pacejka, H. B., and Bakker, E. (1992). The magic formula tyre model. *Vehicle Syst. Dyn.* 21, 1–18. doi: 10.1080/00423119208969994

Rajamani, R. (2011). *Vehicle Dynamics and Control*. Minneapolis, MN: Springer Science & Business Media. doi: 10.1007/978-1-4614-1433-9_2

Rasmussen, C. E. (2003). "Gaussian processes in machine learning," in *Summer School on Machine Learning* (Tübingen: Springer), 63–71. doi: 10.1007/978-3-540-28650-9_4

Rezvani Arany, R. (2019). *Automatic Control* (Master thesis).

Song, Y., Li, M., Luo, X., Yang, G., and Wang, C. (2019). Improved symmetric and nonnegative matrix factorization models for undirected, sparse and large-scaled networks: a triple factorization-based approach. *IEEE Trans. Indus. Inform.* 16, 3006–3017. doi: 10.1109/TII.2019.2908958

Tang, L., Dian, S., Gu, G., Zhou, K., Wang, S., and Feng, X. (2010). "A novel potential field method for obstacle avoidance and path planning of mobile robot," in *2010 3rd International Conference on Computer Science and Information Technology*, Vol. 9 (Chengdu: IEEE), 633–637. doi: 10.1109/ICCSIT.2010.5565069

Williams, C. K., and Rasmussen, C. E. (2006). *Gaussian Processes for Machine Learning*, Vol. 2. Cambridge, MA: MIT Press. doi: 10.7551/mitpress/3206.001.0001

Wu, C., Lin, Y., and Eskandarian, A. (2019a). Cooperative adaptive cruise control with adaptive Kalman filter subject to temporary communication loss. *IEEE Access* 7, 93558–93568. doi: 10.1109/ACCESS.2019.2928004

Wu, D., Luo, X., Shang, M., He, Y., Wang, G., and Zhou, M. (2019b). A deep latent factor model for high-dimensional and sparse matrices in recommender systems. *IEEE Trans. Syst. Man Cybernet. Syst*. 51, 4285–4296. doi: 10.1109/TSMC.2019.2931393

Xie, Z., Jin, L., Luo, X., Li, S., and Xiao, X. (2020). A data-driven cyclic-motion generation scheme for kinematic control of redundant manipulators. *IEEE Trans. Control Syst. Technol*. 29, 53–63. doi: 10.1109/TCST.2019.2963017

Ye, H., Jiang, H., Ma, S., Tang, B., and Wahab, L. (2019). Linear model predictive control of automatic parking path tracking with soft constraints. *Int. J. Adv. Robot. Syst*. 16:1729881419852201. doi: 10.1177/1729881419852201

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.