Sparse Gaussian Process-Based Strategies for Two-Layer Model Predictive Control in Autonomous Vehicle Drifting

Cheng Hu[*], Yangyang Xie[*], Lei Xie [*], Haokun Xiong[*], Hongye Su[*], and Michele Magno[†]
[*] State Key Laboratory of Industrial Control Technology
Zhejiang University, Hangzhou, China
[†]Center for Project-Based Learning, D-ITET, ETH Zurich, Switzerland
Email: {22032081, 22332009} @*zju.edu.cn*, *leix* @*iipc.zju.edu.cn*
haokunxiong @*zju.edu.cn*, *hysu* @*iipc.zju.edu.cn*
michele.magno @*pbl.ee.ethz.ch*

***Abstract*—Vehicle safety is paramount in autonomous driving, particularly when managing vehicles at extreme side-slip angles—a challenge often overlooked by conventional controllers. Recent studies have focused on vehicle drift control under such extreme conditions. However, tracking complex trajectories while drifting is still challenging, especially when a model mismatch exists. This paper proposes a novel two-layer model predictive controller based on sparse variational Gaussian processes. The first layer is responsible for computing the optimal drift equilibrium points, while the second layer is tasked with tracking these points. A variational free energy-based Gaussian process is utilized to compensate for errors in the upper-layer drift equilibrium point calculations as well as mismatches in the lower-layer controller model. To enhance prediction accuracy, vehicle error models are established separately for the transit drift and deep drift phases. The effectiveness of the controller is demonstrated through joint simulations on MATLAB and CarSim platforms. The proposed two-layer model predictive controller is compared with three state-of-the-art drift controllers, demonstrating at least a 43% reduction in average lateral error when tracking trajectories with varying curvature. In the scenario with a 2% friction coefficient mismatch, the experimental results show that the proposed controller, with model error learning, reduces the average lateral error by 72% under model mismatch conditions, which is even 13% lower than the lateral error without model mismatch. Additionally, it shows twice the tracking performance of fully independent training conditional-based model predictive controllers and is ten times faster in computation time compared to fully Gaussian process-based model predictive controllers.**

I. Introduction

Autonomous driving has garnered increasing popularity in recent years [1]. These systems aim to replace human drivers and enhance road safety. However, they encounter significant challenges in extreme driving conditions, such as very slippery surfaces, where traditional control systems may struggle due to tire skidding and model mismatches [2]. The behavior of vehicles during high sideslip situations, such as drifting, significantly differs from normal driving. Drifting involves vehicles achieving a turning equilibrium through controlled oversteer, with the front wheels turning opposite to the direction of normal steering [3]. The pursuit of reaching the limit for autonomous driving has urged a growing research

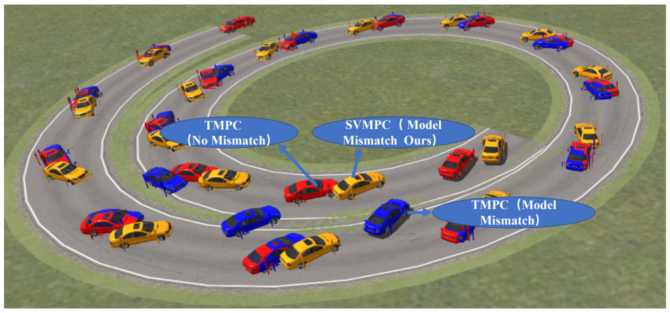


Figure 1: Autonomous drift in CarSim: red vehicle (TMPC without model mismatch), blue vehicle (TMPC with model mismatch, showing significant lateral error), yellow vehicle (SVMPC after model error learning, with superior tracking performance).

interest. Drifting represents another turning equilibrium [4, 5]. One critical distinction is that the drift equilibrium point is a saddle point, making it more susceptible to destabilization and exiting the steady state. Velenis et al. [6] proposed a method for calculating drift equilibrium points by specifying a fixed curvature and a desired sideslip angle, enabling the determination of these points through the vehicle dynamics equations. Subsequent work has primarily focused on how vehicles can track equilibrium points to maintain steady-state drifting [7, 8]. Tracking trajectories while a vehicle drifts poses a more challenging task due to the conflicting objectives of drifting and trajectory tracking [9]. The design philosophy behind both nonlinear controllers [9–11] and the Linear Quadratic Regulator (LQR) controller [12] is to transform the task of trajectory tracking into tracking drift equilibrium points. Nevertheless, they do not adequately consider constraints. Model Predictive Controller (MPC) can account for constraints, but its performance depends on the accuracy of the model. When the vehicle is in a drifting state, model mismatches occur, significantly diminishing its ability to track trajectories. The integration of Gaussian Process (GP) and

Model Predictive Controllers holds considerable potential in addressing these challenges [13, 14]. However, as the data size increases, the Gaussian model must be sparsely processed to ensure the controller's real-time performance.

This paper addresses the issue of simultaneously tracking trajectories and drifting while accounting for mismatches in the vehicle model. We propose a two-layer model predictive controller based on sparse Gaussian strategies with the following contributions:

- A Two-Layer Model Predictive Controller (TMPC) has been proposed to address the issue of tracking trajectories while drifting. The upper layer determines the optimal curvature by considering the lateral deviation from the trajectory to calculate the equilibrium points for drifting. The lower layer linearizes the model at the equilibrium points and employs a linear MPC to track the steady-state values.
- The sparse variational Gaussian process is introduced for the first time in the drifting domain to solve the problem of model mismatch. Initially, based on whether the rear tire forces are saturated and the steering angle exceeds the critical value, it is divided into two local sparse variational Gaussian processes, enhancing the accuracy of predictions. Subsequently, the sparse model compensates for both the model inaccuracies in calculating equilibrium points by the upper layer and the model errors of the lower-layer controller.
- TMPC shows at least a 43% improvement in tracking performance compared to three state-of-the-art drift controllers when tracking trajectories with varying curvature. Additionally, we demonstrate that our controller outperforms the well-known sparsed Fully Independent Training Conditional-based MPC (FCMPC) and Full GP based MPC (FGMPC) regarding tracking performance, model error learning, and computation time.

II. Related Work

A. Classical drift controllers

Some works have addressed the issues of tracking trajectories while drifting. Park et al. [12] proposed a separate lateral and longitudinal control framework. Lateral control is used to track the trajectory by adjusting the steering angle, while longitudinal control ensures the vehicle's state is stabilized towards the drift equilibrium point. If the lateral error is too large, the separate control will fail to maintain the vehicle in a drifting state. Goh et al. [9] formulated an error dynamics model to derive the desired state derivatives, subsequently translating them into control inputs utilizing a nonlinear modeling strategy. This controller successfully tracks trajectories with varying curvature during drifting and has been validated on actual vehicles. Similarly, an MPC approach for trajectory tracking proposed by Chen et al. [15] aims to enhance the input modeling of error dynamics. Nevertheless, a notable limitation of such methods is the neglect of constraints, potentially leading to the derivatives of desired states exceeding the limits imposed by the nonlinear mapping. Some MPC controllers [16–19] that simultaneously optimize steady-state drifting and trajectory tracking have addressed the issue of constraint neglect. Despite this progress, the conflicting objectives make adjusting the weights particularly challenging. To tackle these issues, We propose a two-layer MPC controller to decouple the control issues. The upper layer focuses solely on trajectory tracking, while the lower layer deals exclusively with steady-state drifting. However, model mismatches occurring during the drifting process result in inaccurate calculations of the drift equilibrium points and compromised low-level control, thereby limiting the tracking performance of the controller.

B. Learning based drift controllers

There are also methods employed by machine learning techniques to address model mismatches in drift control systems. Model-free reinforcement learning approaches [20, 21] have been utilized to learn drift control policies, while neural networks [22, 23] have been employed to approximate vehicle models. Nevertheless, these methods do not fully leverage existing mechanistic models, leading to extended training periods to ensure control performance convergence. Djeumou et al. [24] proposed a data-driven tire model to enhance the accuracy of mechanistic models. However, mechanistic models often involve approximations to meet real-time requirements and manage model complexity, resulting in discrepancies with actual vehicle dynamics. In the context of autonomous racing, Hewing et al. [13, 14] proposed a model predictive control approach based on Gaussian Processes. They leveraged a sparse Gaussian Process based on Fully Independent Training Con- ditional (FITC) to fit the model error and implemented an information gain-based selection method to maintain the dataset at 300 data points. This approach enhanced model accuracy while meeting the controller's real-time requirements. Similar to their approach, we also used Gaussian Processes to fit the model error in the proposed drift controller, but with a sparse Gaussian Process based on Variational Free Energy (VFE), which provides better uncertainty estimation and prediction accuracy than FITC [25, 26]. Additionally, to address the problem of fitting the drift data, we constructed two sparse Gaussian Processes based on tire saturation and steering angle value to improve the accuracy of the model error prediction. Moreover, the constructed sparse Gaussian Processes compensate for the error in calculating the equilibrium point, thereby improving the trajectory tracking performance of the controller.

III. Preliminary

A. Gaussian Process Regression

In general, a GP is used to identify an unknown function $f(z) : \mathbb{R}^{n_z} \to \mathbb{R}$ with the following noise model:

$$y = f(z) + \omega \tag{1}$$

where $z \in \mathbb{R}^{n_z}$ is the input, $y \in \mathbb{R}$ is the output, and $\omega \sim \mathcal{N}(0, \Sigma^{\omega})$ is noise term. Consider a training dataset D=$\{z_i, y_i | i = 1, 2, ..N\}$ arising from the unknown function (1), $\mathbf{Z} = [z_1, z_2, ..., z_N]^T \in R^{N\times n_z}$ and $\mathbf{Y} = [y_1, y_2, ..., y_N]^T \in R^N$ are defined. The Prior distribution over the function can be expressed as:

$$f(z) \sim \mathcal{N}(m(z), K_{\mathbf{ZZ}}) \tag{2}$$

where $m(z)$ is the zero mean function. $K_{\mathbf{ZZ}}$ represents the gram matrix, i.e. $[K_{\mathbf{ZZ}}]_{ij} = k(z_i, z_j)$. $k(z_i, z_j)$ is the covariance function. The joint probability distribution of $\mathbf{Y}$ and $f(z^*)$ at the test point z^* can be formulated as:

$$\begin{bmatrix} \mathbf{Y} \\ f(z^*) \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \begin{bmatrix} K_{\mathbf{ZZ}} + \mathbf{I}\Sigma^\omega & K_{\mathbf{Z}z^*} \\ K_{z^*\mathbf{Z}} & K_{z^*z^*} \end{bmatrix}) \tag{3}$$

Then, the Bayesian formula is used to derive the posterior probability of the output at the test point, which includes both the mean and variance of the prediction.

$$\mu^s(z^*) = K_{z^*\mathbf{Z}}(K_{\mathbf{ZZ}} + \mathbf{I}\Sigma^\omega)^{-1}\mathbf{Y} \tag{4}$$

$$\Sigma^s(z^*) = K_{z^*z^*} - K_{z^*\mathbf{Z}}(K_{\mathbf{ZZ}} + \mathbf{I}\Sigma^\omega)^{-1}K_{\mathbf{Z}z^*} \tag{5}$$

In our work, the squared exponential covariance function $k(z, z') = \sigma^2 exp(\frac{-1}{2}(z - z')^\top \Lambda^{-1}(z - z'))$ is applied to define the GP, with hyperparameters σ^2, Σ^ω, and $\Lambda^{-1} = \text{diag}(\lambda_1^{-1}, \lambda_2^{-1}, ..., \lambda_{n_z}^{-1})$. For multidimensional outputs, each dimension is assumed to be independent. The GP approximation of the unknown function with multidimensional outputs can be denoted by

$$d(z) \sim \mathcal{N}(\mu(z^*), \Sigma(z^*)) \tag{6}$$

with $\mu = [\mu_1^s, \ldots, \mu_{n_d}^s]^\top$ and $\Sigma = \text{diag}([\Sigma_1^s, \ldots, \Sigma_{n_d}^s])$. n_d is the dimension of the output.

B. Sparse Gaussian Process

The disadvantage of combining Gaussian processes with controllers is that the computational complexity for calculating the mean and variance increases as data accumulates. Fortunately, many studies focus on the sparse approximation of Gaussian processes. Our work concentrates on approximating the posterior distribution using M inducing points ($M \leq N$). The main principle is to approximate the low-dimensional representation of kernel matrix $K_{\mathbf{ZZ}}$ through global distillation [27]. The computational complexities for the mean and variance of a full GP are $\mathcal{O}(n_d n_z N)$ and $\mathcal{O}(n_d n_z N^2)$, respectively. However, with sparse approximation, the complexities for computing the mean and variance are reduced to $\mathcal{O}(n_d n_z M)$ and $\mathcal{O}(n_d n_z M^2)$ [28].

In the context of extreme driving conditions, a FITC based Gaussian process is integrated into the MPC [13]. This sparse method imposes the full independence assumption to remove the dependency among training data such that given $p(\mathbf{f}|\mathbf{f}_z)$

$$: \prod_{i=1}^{N} p(f_i|\mathbf{f}_z) = \mathcal{N}(K_{\mathbf{ZZ}_m}K_{\mathbf{Z}_m\mathbf{Z}_m}^{-1}\mathbf{f}_z, \text{diag}\,(K_{\mathbf{ZZ}} - Q_{\mathbf{ZZ}})) \tag{7}$$

where $\mathbf{f}$ represents the function values of the training data. $\mathbf{f}_z$ is the latent function of the inducing point with the input of $\mathbf{Z}_m$ similar to $\mathbf{f}$. $Q_{\mathbf{ZZ}}$ is a Nyström notation $\boldsymbol{Q}_{ab} = \boldsymbol{K}_{a\mathbf{Z}_m}\boldsymbol{K}_{\mathbf{Z}_m\mathbf{Z}_m}^{-1}\boldsymbol{K}_{\mathbf{Z}_m b}$. The utilization of the diagonal terms in FITC for variance approximation leads to underestimating the predictive variance. Therefore, we employ the VFE-based Gaussian process into the controller, which offers better predictive performance[25, 27]. The VFE method will be introduced in Section V.

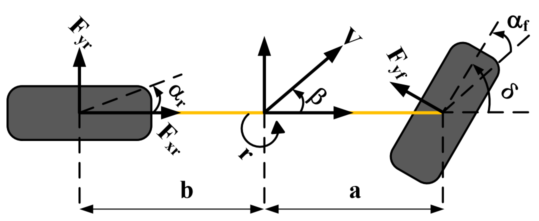


Figure 2: Single-track vehicle model.

IV. Vehicle Model and analysis

A. Drift Dynamics

Table I: Vehicle and Controller Parameters

Parameters	Value	Parameters	Value
m	1830 kg	N	40
I_z	3234 kg·m^2	q_x, q_y	$10, 10$
a	1.40 m	Λ^u	diag$(20, 20)$
b	1.65 m	$\mathcal{U}^u$	$[15, 55] \times [0, 10]$
μ	1	Q	diag$(1, 1, 1)$
$B_{f/r}$	8.32	Λ^l	diag$(1, 0.001)$
$C_{f/r}$	1.63	$\mathcal{U}^l$	$[-0.4, 0.4] \times [-7e3, 7e3]$
T	0.1 s	$\overline{x}^l$	$[20, 1.5, 1.5]^\top$
ϵ	0.05		

The single-track vehicle model depicted in Fig. 2 is widely used in drift controllers due to its appropriate level of complexity and accuracy [2]. The model incorporates the states of velocity V, sideslip angle β, and yaw rate r. The vehicle is assumed to be rear-wheel drive, with the inputs to the model being the steering angle δ and the longitudinal force of the rear tire F_{xr}. The equations are as follows:

$$\dot{V} = \frac{-F_{yf}\sin(\delta - \beta) + F_{yr}\sin(\beta) + F_{xr}\cos(\beta)}{m} \tag{8}$$

$$\dot{\beta} = \frac{F_{yf}\cos(\delta - \beta) + F_{yr}\cos(\beta) - F_{xr}\sin(\beta)}{mV} - r \tag{9}$$

$$\dot{r} = \frac{aF_{yf}\cos(\delta) - bF_{yr}}{I_z} \tag{10}$$

where m is the vehicle's mass and I_z represents the moment of inertia in the vertical direction. The distances from the center of gravity (CoG) to the front and rear tires are indicated by a and b, respectively. Additionally, the lateral forces on the front and rear tires are denoted as F_{yf} and F_{yr}.

The lateral forces are modeled using the simplified Pacejka Formula [29]:

$$F_{yi} = \begin{cases} -\mu F_{zi}\sin(C\arctan(B\alpha_i)) & \text{if } |\alpha_i| < \alpha_{sl} \\ -\mu F_{zi}\text{sgn}(\alpha_i) & \text{if } |\alpha_i| \geq \alpha_{sl} \end{cases} \tag{11}$$

with

$$\alpha_{sl} = \tan\left(\arcsin(1)/C\right)/B \tag{12}$$

where μ is the coefficient of friction, F_{zi} is the normal load on the tire ($i \in (f, r)$). B and C are parameters that that necessitate identification. α_{sl} signifies the critical value of tire slip angle. The slip angle α_i for the front and rear tires can be expressed as follows:

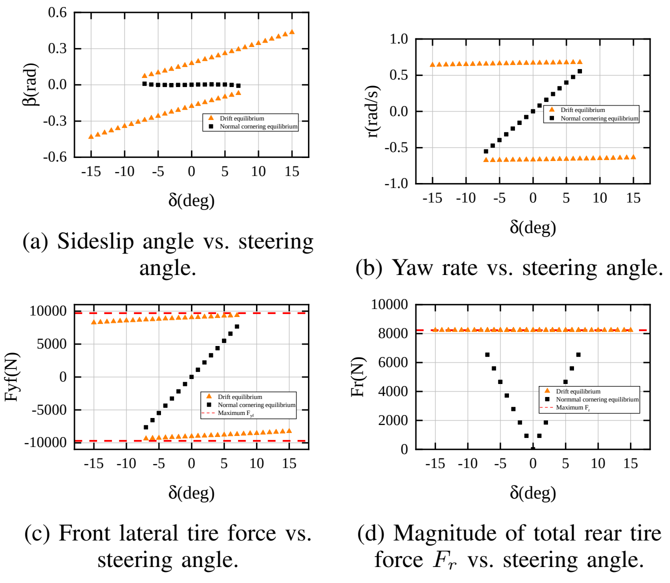
$$\alpha_f = \arctan\left(\frac{V\sin(\beta) + ar}{V\cos(\beta)}\right) - \delta \tag{13}$$

$$\alpha_r = \arctan\left(\frac{V\sin(\beta) - br}{V\cos(\beta)}\right) \tag{14}$$

B. Drift Steady Analysis

Drifting is another manifestation of turning equilibrium [3]. Given the vehicle's velocity and steering angle, the equilibrium points can be determined by setting the derivatives of Eqs. (8) to (10) to zero. In this study, the velocity is set to 50 km/h, while the steering angle varies between -15° and 15°. Multiple equilibrium points are then obtained, as illustrated in Fig. 3.

The black markers represent normal turning equilibrium points characterized by a small sideslip angle, while the yellow markers correspond to equilibrium points for drifting with a large sideslip angle and nearly saturated front tire forces. This makes it very challenging to simultaneously control the front wheel to follow the trajectory while maintaining the drift. The figure also demonstrates that when the steering angle δ remains within $\pm 7°$, the vehicle can transition between normal turning and drifting. Beyond this threshold, exceeding $-7°$ initiates a deep drift state (left-handed drift). Therefore, this angle is identified as the critical point for deep drifting, denoted as $\delta_{sl} = -7°$.



(a) Sideslip angle vs. steering angle.
(b) Yaw rate vs. steering angle.
(c) Front lateral tire force vs. steering angle.
(d) Magnitude of total rear tire force F_r vs. steering angle.

Figure 3: Equilibrium points of drifting and normal turning.

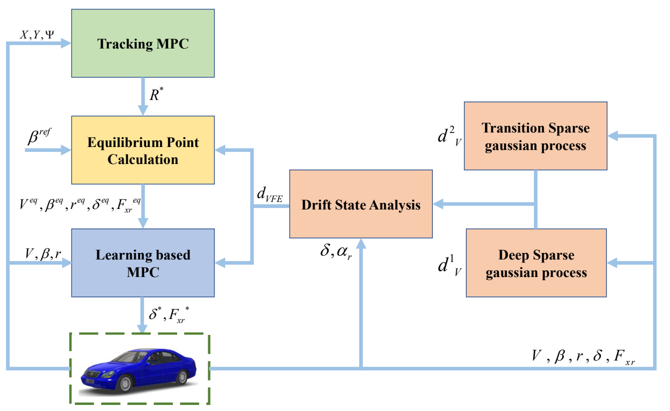

Tracking MPC
X, Y, Ψ
R^*
β^{ref}
Equilibrium Point Calculation
$V^{eq}, \beta^{eq}, r^{eq}, \delta^{eq}, F_{xr}^{eq}$
V, β, r
Learning based MPC
δ^*, F_{xr}^*
d_{VFE}
Drift State Analysis
δ, α_r
d^2_V
Transition Sparse gaussian process
d^1_V
Deep Sparse gaussian process
$V, \beta, r, \delta, F_{xr}$

Figure 4: Two-layer model predictive controller structure: the upper layer calculates the drift equilibrium point for trajectory tracking, while the lower layer tracks the setpoint to maintain drift. Based on whether the vehicle has entered a deep drift state, different local Gaussians are selected for error compensations.

Furthermore, Fig. 3d reveals that the vehicle's rear wheels in the drift steady state satisfy the following dynamic equation:

$$\sqrt{F_{xr}^2 + F_{yr}^2} = \mu F_{zr} \tag{15}$$

References [9, 16] also demonstrate that when a vehicle is drifting, and the rear tire slip angle exceeds the critical value at the first time, the resultant force on the rear tire F_r must satisfy Eq. (15). Consequently, the critical rear tire slip angle and steering angle serve as indicators to ascertain whether the vehicle has transitioned into a deep drift state. In subsequent controller design, distinct sparse Gaussian models can be established based on the vehicle's drift status.

V. Methodology

In this section, we will introduce the proposed method called Sparse Variational Gaussian Processes with Two-Layer Model Predictive Control (SVMPC). The controller block diagram is shown in Fig. 4.

A. Variational Free Energy Gaussian process

The well-known posterior approximation of the Gaussian process is the VFE method [26], which is introduced in Appendix VIII-A.

The VFE based sparse GP is used to learn the error dynamics of the vehicle model[14]. Based on the previous analysis of drift states, it is observed that the data distribution differs before and after entering the deep drift. In order to enhance the precision of predictions, two local Gaussians can be constructed by assessing the saturation of tire force and reverse degree of steering angle:

$$d_{\text{VFE}} = \begin{cases} d_V^1 & \text{if } |\alpha_r| \geq \alpha_{sl} \quad \text{and } \delta \leq \delta_{sl} \\ d_V^2 & \text{else} \end{cases} \tag{16}$$

The first GP d_V^1 is for the deep drift state, and the second d_V^2 is for the transition state.

B. Trajectory Tracking layer

In our previous work [18], the vehicle's future trajectory was considered to be an arc with a fixed drifting radius, which was very effective when tracking circular paths. However, when extended to complex trajectories, this assumption does not hold true, and the future path should instead be an arc with varying curvature. Therefore, we developed a discrete model for predicting varying curvature trajectories, using the longitudinal position X, lateral position Y, and heading angle Ψ as the model states x^u, and the radius R and arc length L as control inputs u^u. The model is expressed as follows:

$$X_{i+1} = X_i + R_i \left(\sin(\Psi_{i+1}) - \sin(\Psi_i)\right) \tag{17}$$

$$Y_{i+1} = Y_i + R_i \left(\cos(\Psi_{i+1}) - \cos(\Psi_i)\right) \tag{18}$$

$$\Psi_{i+1} = \Psi_i + \frac{R_i}{L_i} \tag{19}$$

Then, the formulation of the optimal curvature controller using the model is

$$\begin{aligned} \min_{u^u} \quad & \sum_{i=1}^{N-1} q_x(X_{k+i} - X_{ref})^2 + q_y(Y_{k+i} - Y_{ref})^2 \\ & + \|u^u_{k+i} - u^u_{k+i-1}\|^2_{\Lambda^u} + q_x(X_{k+N} - X_{ref})^2 \\ & + q_y(Y_{k+N} - Y_{ref})^2 && (20) \\ \text{s.t.} \quad & x^u_k = x^u_0 && (21) \\ & Eqs.\ (17)\ to\ (19) && (22) \\ & u^u_{k+i} \in \mathcal{U}^u && (23) \\ & \forall i = 0, 1, \cdots, N-1 \end{aligned}$$

where $x^u = \begin{bmatrix} X & Y & \Psi \end{bmatrix}^\top$ and $u^u = \begin{bmatrix} R & L \end{bmatrix}$. The parameters q_x, q_y, and Λ^u are weights, and N is the prediction horizon.

Only the first stage of the input from the MPC solution is used to calculate the optimal drift equilibrium point:

$$u^*_{k|k} = \begin{bmatrix} R^* & L^* \end{bmatrix}^\top \tag{24}$$

The state of the drift model defined in Section IV-A is $x^l = \begin{bmatrix} V & \beta & r \end{bmatrix}^\top$, and the control input is $u^l = \begin{bmatrix} \delta & F_{xr} \end{bmatrix}^\top$. The drift dynamics can be denoted as

$$\dot{x}^l = f_v(x^l, u^l) \tag{25}$$

Drift equilibrium is a form of steady-state turning, which also satisfies the formula for calculating the steady-state radius [6] $R = \frac{V}{r}$. Thus, the velocity can be expressed in terms of the yaw rate and the optimal radius R^*. By fixing the reference sideslip angle β^{ref}, the equilibrium point can be determined by solving $f_v(x^l, u^l) = 0$. To address the issue of inaccurate equilibrium point calculation due to model mismatch, the sparse Gaussian model d_{VFE} is used to fit the model error, and the following compensation is applied when calculating the equilibrium point:

$$0 = f_v(x^l, u^l) + d_{\text{VFE}}(x^l, u^l)/T \tag{26}$$

where T is the control period. The vectors $x^{eq} = \begin{bmatrix} V^{eq} & \beta^{eq} & r^{eq} \end{bmatrix}^\top$ and $u^{eq} = \begin{bmatrix} \delta^{eq} & F^{eq}_{xr} \end{bmatrix}^\top$ represent the optimal equilibrium point of the drift dynamics.

Remark. *Consider the true dynamics of the nonlinear system*

$$\dot{x}^l = f_{v,true}(x^l, u^l) = f_v(x^l, u^l) + \varepsilon \tag{27}$$

where ε is the model error of continuous time system. Let (x^l_{eq}, u^l_{eq}) be the equilibrium point for which $0 = f_v(x^l_{eq}, u^l_{eq}) + \varepsilon$. The forward Euler's discrete appproximation of Eq. (27) *is*

$$\begin{aligned} x^l_{k+1} =& x^l_k + T[f_v(x^l_k, u^l_k) + \varepsilon] \\ =& f_d(x^l_k, u^l_k) + T\varepsilon \end{aligned}$$

Thus we have $\varepsilon \approx d_{VFE}/T$.

C. Learning-based Control Layer

By linearizing the drift model at the optimal equilibrium point and discretizing it using the forward Euler method, we derive the following nominal model:

$$x^l_{k+1} = Ax^l_k + Bu^l_k + d^l \tag{28}$$

$$d^l = x^{eq} - Ax^{eq} - Bu^{eq} \tag{29}$$

which is defined as

$$x^l_{k+1} = f_d(x^l_k, u^l_k) \tag{30}$$

Due to its simplifications, the single-track model cannot fully capture the vehicle's drifting dynamics. The spare Gaussian process is employed to learn the model error dynamics.

$$x^l_{k+1} = f_d(x^l_k, u^l_k) + d_{\text{VFE}}(x^l_k, u^l_k) \tag{31}$$

At each time step, the predicted state, owing to the Gaussian error term, becomes a stochastic distribution after linear superposition. Therefore, the state constraints can be expressed as chance constraints, representing the maximum probability of constraint satisfaction. The learning-based MPC can be formulated as follows:

$$\begin{aligned} \min_{\nu_{k+i}} \quad & \mathbb{E}\left(\sum_{i=1}^{N-1} l(\mathbf{x}^l_{k+i}, u^l_{k+i}) + Q(\mathbf{x}^l_{k+N}, u^l_{k+N})\right) && (32) \\ \text{s.t.} \quad & x^l_{k+i+1} = f_d(x^l_{k+i}, u^l_{k+i}) + d_{\text{VFE}}(x^l_{k+i}, u^l_{k+i}) && (33) \\ & u^l_{k+i} = Kx^l_{k+i} - Kx^{eq} + u^{eq} + \nu_{k+i} && (34) \\ & \mathcal{P}(-\overline{x}^l \le x^l_{k+i+1} \le \overline{x}^l) \ge 1 - \epsilon && (35) \\ & u^l_{k+i} \in \mathcal{U}^l && (36) \\ & x^l_k = x^l_0 && (37) \\ & \forall i = 0, 1, \cdots, N-1 \end{aligned}$$

where K is the local feedback gain, which can be calculated by the LQR [30, 31]. ϵ represents the significance level of constraint violation. The optimization problem of MPC cannot be solved directly due to the chance constraint (35). Then, the chance constraint is reformulated by the double half-space constraint [13].

x^l becomes a random variable that follows the Gaussian distribution. Its mean and the variance are denoted by μ^{x^l} and Σ^{x^l}, respectively. Given Eq. (31), the mean and variance of the predicted state can be expressed as follows:

$$\mu_{k+1}^{x^l} = f_d(\mu_k^{x^l}, u_k^l) + \mu^{d_v}(\mu_k^{x^l}, u_k^l) \tag{38}$$

$$\Sigma_{k+1}^{x^l} = \tilde{A}\Sigma_k^{x^l}\tilde{A}^\top + \Sigma^{d_v}(\mu_k^{x^l}, u_k^l) + \Sigma^w \tag{39}$$

where $\tilde{A} = \nabla(f_d(x_k^l, u_k^l,)) + \mu^{d_v}(x_k^l, u_k^l)|_{\mu_k^{x^l}}$.

The error between the state and its mean is denoted by e^{x^l}, which can be transformed to the normal distribution $Ce^{x^l} \sim \mathcal{N}(0, C^T\Sigma^{x^l}C)$.

Definition 1 (Probabilistic Reachable Set)**.** *A set is a probabilistic reachable set of significance level ϵ if*

$$\mathcal{P}(e^{x^l} \in \mathcal{R}) \geq 1 - \epsilon \tag{40}$$

Then the constraint of the mean μ^{x^l} is defined as follows:

$$\mu^{x^l} \in \mathcal{X}^\mu = \mathcal{X}^{x^l} \ominus \mathcal{R} \tag{41}$$

where $\mathcal{X}^{x^l} = \{x^l | -\overline{x}^l \leq x^l \leq \overline{x}^l\}$. Satisfaction of the tightened constraint Eqs. (40) and (41) implies satisfaction of the original chance constraint Eq. (35) since $\mathcal{P}(x^l = \mu^{x^l} + e^{x^l} \in \mathcal{X}^{x^l}) \geq \mathcal{P}(e^{x^l} \in \mathcal{R}) \geq 1 - \epsilon$. The quantile function $z(\epsilon/2)$ is used to change the probabilistic reachable set to a deterministic constraint:

$$\mathcal{R}\left(\Sigma_k^{x^l}\right) = \left\{e^{x^l} \middle| e^{x^l} \leq \left| z(\epsilon/2)\sqrt{C^\top \Sigma_k^{x^l} C}\right| \right\} \tag{42}$$

Then the equivalent satisfaction of the chance constraint (35) is defined as:

$$\begin{aligned}\mathcal{X}^\mu\left(\Sigma_k^{x^l}\right) &= \mathcal{X}^{x^l} \ominus \mathcal{R}\left(\Sigma_k^{x^l}\right) \\ &= \left\{x_k^l \middle| \left|x_k^l - \overline{x}^l\right| \leq z(\epsilon/2)\sqrt{C^\top \Sigma_k^x C}\right\}\end{aligned} \tag{43}$$

To achieve computational efficiency, the estimation of the cost function utilizes the mean and variance of the state [13]. The optimization problem (32) can be transformed into a solvable MPC problem as follows:

$$\min_{\nu_{k+i}} \sum_{i=1}^{N-1} \left(\|\mu_{k+i}^{x^l} - x^{eq}\|_Q^2 + \|u_{k+i}^l - u^{eq}\|_{\Lambda^l}^2\right) + \|\mu_{k+N}^{x^l} - x^{eq}\|_Q^2 \tag{44}$$

$$\text{s.t.} \quad \mu_{k+1+i}^{x^l} = f_d(\mu_{k+i}^{x^l}, u_{k+i}^l) + d_{\text{VFE}}(\mu_{k+i}^{x^l}, u_{k+i}^l) \tag{45}$$

$$u_{k+i}^l = K\mu_{k+i}^{x^l} - Kx^{eq} + u^{eq} + \nu_{k+i} \tag{46}$$

$$\Sigma_{k+1+i}^{x^l} = \tilde{A}_{k+i}\Sigma_{k+i}^{x^l}\tilde{A}_{k+i}^\top + \Sigma^{d_v}(\mu_{k+i}^x, u_{k+i}^l) \tag{47}$$

$$\mu_{k+i}^{x^l} \in \mathcal{X}^\mu(\Sigma_{k+i}^{x^l}) \tag{48}$$

$$u_{k+i}^l \in \mathcal{U}^l \tag{49}$$

$$\mu_k^{x^l} = x_0^l \tag{50}$$

$$\forall i = 0, 1, \cdots, N-1$$

where Q and Λ^l are weight matrices. The first element of the solution $u_k^{l,*} = \begin{bmatrix}\delta^* & F_{xr}^*\end{bmatrix}^T$ is then implemented as the control command.

VI. Simulation Result

A. Simulation Setup

A clothoid reference trajectory is generated with the curvature of the trajectory ranging from $1/40$ to $1/21$ over a distance of 265 m [9]. The reference sideslip angle is -0.61 rad. The vehicle starts to run at the speed of 50 km/h.

Initially, without the friction coefficient mismatch, the controller's ability to track a varying curvature trajectory is verified without compensating the model error using GP. The comparison with the State-of-the-Art drift controllers can found in Appendix VIII-B. Subsequently, employing the same controller, the vehicle tracks the second lap with a 2% reduction in the friction coefficient of the vehicle model. After collecting data from the second lap, the learning-based controller is activated on the third lap. It continues learning for a total of five laps to test the controller's ability to track the trajectory under model mismatch conditions.

Additionally, the proposed controller SVMPC will be compared with four kinds of controllers to demonstrate the effectiveness of model error learning.

- **TMPC**: the same control structure as SVMPC but without GP compensation.
- **SVMPC-1**: the same control structure with a single VFE to fit the model error.
- **FCMPC**: the same control structure with a single FITC to fit the model error.
- **FGMPC**: the same control structure with a single full GP to fit the model error.

In the specific implementation of the algorithm, three states, including velocity, sideslip angle, and yaw rate, are selected for error compensation, indicating that the GP dimension is $n_d = 3$. The number of inducing points is set to M=15. To prevent the dataset from continuously expanding and becoming unsuitable for real-time requirements, outlier removal and information gain are used to limit the dataset size to N=200 [14]. d_V^1 and d_V^2 are optimized using 100 data points each, depending on whether the vehicle enters a deep drift state. In contrast, STMPC-1, FCMPC, and FGMPC each train only one GP model. The experiment is carried out on a MATLAB and CarSim joint simulation platform. The simulation is executed on a computer with Windows 10 OS equipped with an Intel i7 processor clocked at 1.8 GHz. FORCES Pro [32] is used to solve the MPC optimization problem. Similar to [13], the variance dynamics (47) are pre-evaluated based on the previous MPC solution, enabling the precomputation of state constraints (48) and meeting real-time requirements. The vehicle and controller parameters can be found in Table I.

B. Simulation Results

Following the simulation results of the first lap shown in Fig. 1, it can be observed that the red vehicle controlled by TMPC successfully followed the trajectory while drifting, demonstrating the capability in tracking complex trajectories. Before the start of the second lap, the ground friction coefficient was reduced by 2%, making the surface slipperier

Table II: Experimental Results of FGMPC with 2% Friction Reduction.

Lap	Avg. e (m)	Max. e (m)	V		β		r	
			$\|e_{GP}\|^1$	$1-\epsilon(\%)$	$\|e_{GP}\|^1$	$1-\epsilon(\%)^2$	$\|e_{GP}\|^1$	$1-\epsilon(\%)^2$
1	0.4399	1.5593	-	-	-	-	-	-
2	1.3464	5.2684	-	-	-	-	-	-
3	0.4555	**0.9182**	0.0132	75.43	0.0035	95.43	0.0636	70.86
4	0.6445	1.6680	0.0188	64.41	**0.0038**	78.53	0.0576	53.11
5	0.3889	0.9398	0.0129	66.29	0.0030	91.43	0.0651	65.71
6	0.4244	0.9602	0.0173	48.57	**0.0028**	78.86	0.0642	50.86
7	0.4676	1.0129	0.0160	61.71	**0.0023**	86.86	0.0651	50.29

[1] e_{GP} represent the difference between the predicted error of the Full GP and the actual error.
[2] $1-\epsilon$ indicates the percentage of actual error data that falls within the 95% confidence interval of the Full GP.

Table III: Experimental Results of FCMPC with 2% Friction Reduction.

Lap	Avg. e (m)	Max. e (m)	V		β		r	
			$\|e_{GP}\|^1$	$1-\epsilon(\%)$	$\|e_{GP}\|^1$	$1-\epsilon(\%)^2$	$\|e_{GP}\|^1$	$1-\epsilon(\%)^2$
1	0.4399	1.5593	-	-	-	-	-	-
2	1.3464	5.2684	-	-	-	-	-	-
3	0.8165	3.1676	0.0200	**93.96**	0.0050	**98.35**	0.0485	**96.70**
4	0.5839	1.2607	0.0153	**94.32**	0.0051	**98.30**	0.0308	**96.59**
5	0.9818	3.0102	0.0193	**95.63**	0.0036	**97.81**	0.0401	**95.63**
6	0.7250	2.3478	0.0177	**94.54**	0.0050	**96.17**	0.0430	83.06
7	1.1279	2.3833	0.0263	**92.15**	0.0044	**96.34**	0.0295	**96.86**

[1] e_{GP} represent the difference between the predicted error of the FITC and the actual error.
[2] $1-\epsilon$ indicates the percentage of actual error data that falls within the 95% confidence interval of the FITC.

Table IV: Experimental Results of SVMPC-1 (**ours**) with 2% Friction Reduction.

Lap	Avg. e (m)	Max. e (m)	V		β		r	
			$\|e_{GP}\|^1$	$1-\epsilon(\%)$	$\|e_{GP}\|^1$	$1-\epsilon(\%)^2$	$\|e_{GP}\|^1$	$1-\epsilon(\%)^2$
1	0.4399	1.5593	-	-	-	-	-	-
2	1.3464	5.2684	-	-	-	-	-	-
3	**0.4507**	1.2921	**0.0109**	89.20	**0.0025**	96.02	**0.0247**	85.23
4	**0.4429**	**0.9797**	0.0122	81.36	0.0048	88.14	0.0508	78.53
5	0.4008	**0.8657**	0.0091	88.57	**0.0019**	93.14	0.0283	88.00
6	0.4160	0.9028	0.0137	82.86	0.0035	92.57	0.0352	**85.14**
7	0.4449	1.1749	0.0104	85.23	0.0024	97.16	0.0316	82.95

[1] e_{GP} represents the difference between the predicted error of VFE and the actual error.
[2] $1-\epsilon$ indicates the percentage of actual error data that falls within the 95% confidence interval of the VFE.

Table V: Experimental Results of SVMPC (**ours**) with 2% Friction Reduction.

Lap	Avg. e (m)	Max. e (m)	V		β		r	
			$\|e_{GP}\|^1$	$1-\epsilon(\%)$	$\|e_{GP}\|^1$	$1-\epsilon(\%)^2$	$\|e_{GP}\|^1$	$1-\epsilon(\%)^2$
1	0.4399	1.5593	-	-	-	-	-	-
2	1.3464	5.2684	-	-	-	-	-	-
3	0.5193	1.1631	0.0143	68.57	0.0070	81.14	0.0307	65.14
4	0.4830	1.1218	**0.0107**	72.57	0.0050	75.43	**0.0120**	70.29
5	**0.3765**	0.9077	**0.0081**	89.66	0.0025	90.80	**0.0176**	85.06
6	**0.3864**	**0.9013**	**0.0136**	86.21	0.0029	91.38	**0.0185**	75.86
7	**0.3807**	**0.9056**	**0.0079**	86.78	0.0024	91.38	**0.0150**	83.33

[1] e_{GP} represents the difference between the predicted error of VFE and the actual error.
[2] $1-\epsilon$ indicates the percentage of actual error data that falls within the 95% confidence interval of the VFE.

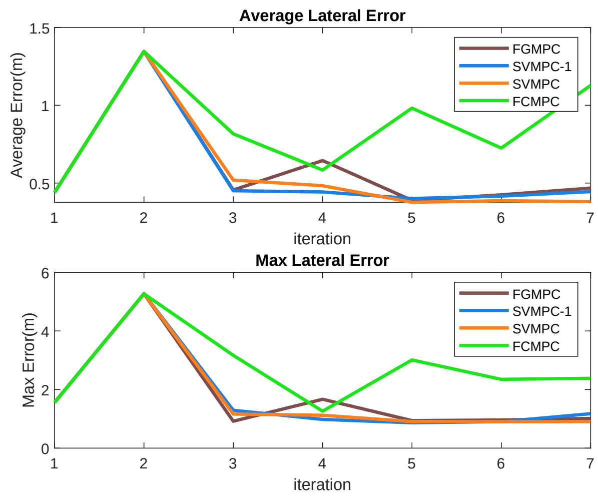


Figure 5: Average lateral errors for the controllers.

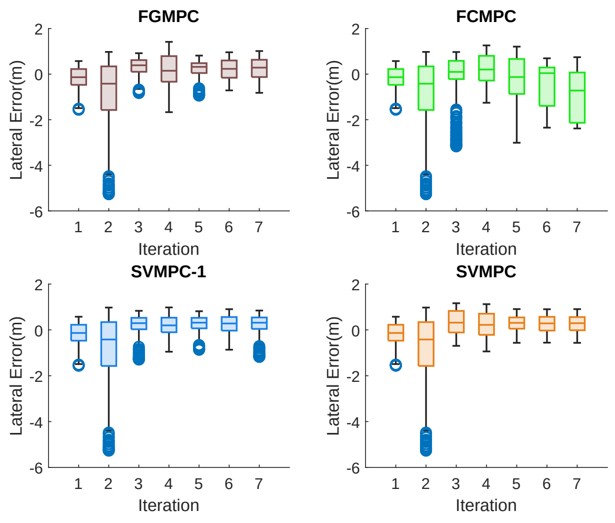


Figure 6: Lateral error boxplots for controllers.

and increasing the difficulty for the blue vehicle controlled by TMPC to track the trajectory. Fig. 5 shows that the maximum lateral error of the TMPC reached 5.27 m in the second lap, which is three times the maximum error of the first lap. This indicates the risk associated with the vehicle drifting on unknown surfaces.

After collecting data from the second lap, we conducted offline optimization to compute the hyperparameters and the inducing points. Subsequently, we carried out five-lap learning tests on SVMPC, SVMPC-1, FCMPC, and FGMPC. As shown in Table V, our proposed method achieved an average lateral error of 0.38 m after learning for five laps, which is 72% smaller than the lateral error (1.35 m) in the second lap. Moreover, its average lateral error and maximum lateral error are even 13% and 42% smaller, respectively, compared to the

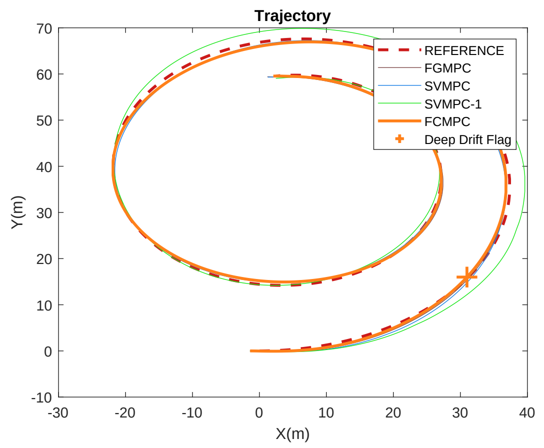


Figure 7: Comparison of track performance in different controllers in the final lap.

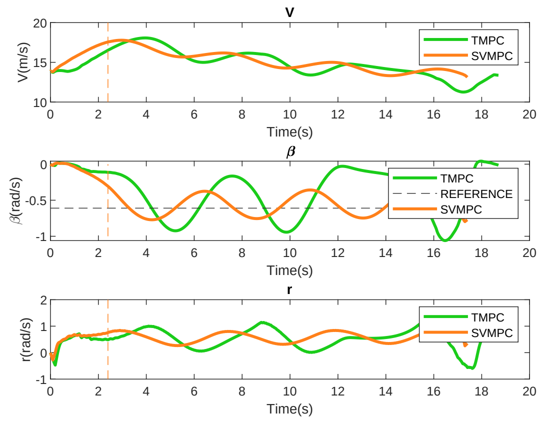


Figure 8: Comparison of states between the first lap without GP compensation in TMPC and the last lap in SVMPC under 2% friction coefficient loss.

controller without model mismatch in the first lap. This controller demonstrates the ability to improve trajectory tracking performance under model mismatch conditions by learning the model errors. As for maintaining the drift state, after model error learning, the SVMPC shows better tracking of the reference sideslip angle compared TMPC, shown in Fig. 8. At 2.4 s, the vehicle's slip angle and steering angle reach -0.37 rad and -0.41 rad, respectively, exceeding the critical values. At this point, the vehicle enters a deep drift state. Table II, Table III, and Table IV illustrate the complete model error learning process for FGMPC, FCMPC, and SVMPC-1, respectively. The lateral error comparison between Table IV and Table V demonstrates that by establishing sparse variational Gaussian processes before and after entering deep drift, SVMPC can track the trajectory better than SVMPC-1. Additionally, the experimental results of the lateral error and trajectories in Table III and Fig. 7 indicate that VFE-based MPC outperforms FITC-based MPC in trajectory tracking. Table II shows that the Full GP-based MPC achieved the smallest maximum lateral error of 0.92 m in the third lap, although its overall tracking performance was weaker than that of SVMPC. The model error learning comparsion can be found in Appendix VIII-C.

VII. Conclusion

This paper proposes a method that integrates variational sparse Gaussian processes with a two-layer model predictive controller to tackle model mismatch challenges when a drifting vehicle is tracking complex trajectories. Compared to three state-of-the-art drift controllers, simulation results demonstrate that the proposed TMPC achieves at least a 43% improvement in tracking performance on the trajectory with varying curvature. In scenery where the model's friction coefficient loss is 2%, experimental results show that the tracking performance of the VFE-based MPC controller is superior to that of the FITC-based MPC controller. Additionally, splitting the sparse model into two local Gaussian processes based on whether the vehicle enters a drift state significantly improves the accuracy of GP predictions. Compared to FGMPC, the proposed method is ten times faster regarding solution time and achieves better tracking performance.

Future work will focus on enriching the GP dataset to enhance the efficiency of learning while ensuring the stability of the controller.

Acknowledgments

Cheng Hu and Yangyang Xie are equally contributed. Lei Xie is the corresponding author. Thanks to Xiaoling Zhou and Ran Duo for their valuable suggestions. Thanks to PBL staff for the support and help.

References

[1] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020.

[2] Fang Zhang, Jon Gonzales, Shengbo Eben Li, Francesco Borrelli, and Keqiang Li. Drift control for cornering maneuver of autonomous vehicles. *Mechatronics*, 54: 167–174, 2018.

[3] Rami Y Hindiyeh and J Christian Gerdes. A controller framework for autonomous drifting: Design, stability, and experimental validation. *Journal of Dynamic Systems, Measurement, and Control*, 136(5):051015, 2014.

[4] Sina Milani, Hormoz Marzbani, and Reza N Jazar. Vehicle drifting dynamics: discovery of new equilibria. *Vehicle system dynamics*, 60(6):1933–1958, 2022.

[5] Haotian Dong, Huilong Yu, and Junqiang Xi. Phase portrait analysis and drifting control of unmanned tracked vehicles. *IEEE Transactions on Intelligent Vehicles*, 2024.

[6] Efstathios Velenis, Emilio Frazzoli, and Panagiotis Tsiotras. Steady-state cornering equilibria and stabilisation for a vehicle during extreme operating conditions. *International Journal of Vehicle Autonomous Systems*, 8(2-4): 217–241, 2010.

[7] Eunhyek Joa, Hyunsoo Cha, Youngjin Hyun, Youngil Koh, Kyongsu Yi, and Jaeyong Park. A new control approach for automated drifting in consideration of the driving characteristics of an expert human driver. *Control Engineering Practice*, 96:104293, 2020.

[8] Efstathios Velenis, Diomidis Katzourakis, Emilio Frazzoli, Panagiotis Tsiotras, and Riender Happee. Steady-state drifting stabilization of rwd vehicles. *Control Engineering Practice*, 19(11):1363–1376, 2011.

[9] Jonathan Y Goh, Tushar Goel, and J Christian Gerdes. Toward automated vehicle control beyond the stability limits: drifting along a general path. *Journal of Dynamic Systems, Measurement, and Control*, 142(2): 021004, 2020.

[10] Jonathan Y Goh and J Christian Gerdes. Simultaneous stabilization and tracking of basic automobile drifting trajectories. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 597–602. IEEE, 2016.

[11] Fengjiao Jia, Houhua Jing, and Zhiyuan Liu. A novel nonlinear drift control for sharp turn of autonomous vehicles. *Vehicle System Dynamics*, 62(2):490–510, 2024.

[12] Mincheol Park and Yeonsik Kang. Experimental verification of a drift controller for autonomous vehicle tracking: A circular trajectory using lqr method. *International Journal of Control, Automation and Systems*, 19(1):404–416, 2021.

[13] Lukas Hewing, Juraj Kabzan, and Melanie N Zeilinger. Cautious model predictive control using gaussian process regression. *IEEE Transactions on Control Systems Technology*, 28(6):2736–2743, 2019.

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

[15] Guoying Chen, Xuanming Zhao, Zhenhai Gao, and Min Hua. Dynamic drifting control for general path tracking of autonomous vehicles. *IEEE Transactions on Intelligent Vehicles*, 2023.

[16] Jonathan YM Goh, Michael Thompson, James Dallas, and Avinash Balachandran. Beyond the stable handling limits: nonlinear model predictive control for highly transient autonomous drifting. *Vehicle System Dynamics*, pages 1–24, 2024.

[17] Haotian Dong, Huilong Yu, and Junqiang Xi. Real-time model predictive control for simultaneous drift and trajectory tracking of autonomous vehicles. In *2022 6th CAA International Conference on Vehicular Control and Intelligence (CVCI)*, pages 1–6. IEEE, 2022.

[18] Yu Qi, Zhiming Zhang, Cheng Hu, Xiaoling Zhou, Lei Xie, and Hongye Su. An mpc-based controller framework for agile maneuvering of autonomous vehicles. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 1228–1234. IEEE, 2021.

[19] Cheng Hu, Xiaoling Zhou, Ran Duo, Haokun Xiong, Yu Qi, Zhiming Zhang, and Lei Xie. Combined fast control of drifting state and trajectory tracking for autonomous vehicles based on mpc controller. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1373–1379. IEEE, 2022.

[20] Mark Cutler and Jonathan P How. Autonomous drifting using simulation-aided reinforcement learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5442–5448. IEEE, 2016.

[21] Peide Cai, Xiaodong Mei, Lei Tai, Yuxiang Sun, and Ming Liu. High-speed autonomous drifting with deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):1247–1254, 2020.

[22] Manuel Acosta and Stratis Kanarachos. Teaching a vehicle to autonomously drift: A data-based approach using neural networks. *Knowledge-Based Systems*, 153: 12–28, 2018.

[23] Xiaoling Zhou, Cheng Hu, Ran Duo, Haokun Xiong, Yu Qi, Zhiming Zhang, Hongye Su, and Lei Xie. Learning-based mpc controller for drift control of autonomous vehicles. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 322–328. IEEE, 2022.

[24] Franck Djeumou, Jonathan YM Goh, Ufuk Topcu, and Avinash Balachandran. Autonomous drifting with 3 minutes of data via learned tire models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 968–974. IEEE, 2023.

[25] Matthias Bauer, Mark Van der Wilk, and Carl Edward Rasmussen. Understanding probabilistic sparse gaussian process approximations. *Advances in neural information processing systems*, 29, 2016.

[26] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pages 567–574. PMLR, 2009.

[27] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When gaussian process meets big data: A review of scalable gps. *IEEE transactions on neural networks and learning systems*, 31(11):4405–4423, 2020.

[28] Joaquin Quinonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

[29] Hans B Pacejka and Egbert Bakker. The magic formula tyre model. *Vehicle system dynamics*, 21(S1):1–18, 1992.

[30] Anil Aswani, Humberto Gonzalez, S Shankar Sastry, and Claire Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013.

[31] Ben Tearle, Kim P Wabersich, Andrea Carron, and Melanie N Zeilinger. A predictive safety filter for learning-based racing control. *IEEE Robotics and Automation Letters*, 6(4):7635–7642, 2021.

[32] Andrea Zanelli, Alexander Domahidi, Juan Jerez, and Manfred Morari. Forces nlp: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, 93(1):13–29, 2020.

VIII. Appendix

A. Variational Free Energy Gaussian process

VFE method involves jointly inferring the inducing points and the hyperparameters by maximizing a lower bound of the marginal likelihood. Specifically, the approximate posterior is assumed to be $q(\mathbf{f},\mathbf{f}_z|\mathbf{Y}) = p(\mathbf{f}|\mathbf{f}_z)\phi(\mathbf{f}_z)$, where $\phi(\mathbf{f}_z)$ is a variational distribution over $\mathbf{f}_z$. This assumption enables a critical cancellation that results in a computationally tractable lower bound. The bound can be obtained by utilizing Jensen's inequality:

$$\begin{aligned}\log p(\mathbf{Y}) &= \log \int p\left(\mathbf{f}|\mathbf{f}_z\right)\phi\left(\mathbf{f}_z\right)\frac{p(\mathbf{Y}|\mathbf{f})p\left(\mathbf{f}|\mathbf{f}_z\right)p\left(\mathbf{f}_z\right)}{p\left(\mathbf{f}|\mathbf{f}_z\right)\phi\left(\mathbf{f}_z\right)}d\mathbf{f}d\mathbf{f}_z\\ &\geq \int p(\mathbf{f}|\mathbf{f}_z)\phi(\mathbf{f}_z)\log\frac{p(\mathbf{Y}|\mathbf{f})\cancel{p(\mathbf{f}|\mathbf{f}_z)}p(\mathbf{f}_z)}{\cancel{p(\mathbf{f}|\mathbf{f}_z)}\phi(\mathbf{f}_z)}d\mathbf{f}d\mathbf{f}_z\\ &= \mathcal{F}_V(\phi(\mathbf{f}_z),\sigma^2,\Lambda^{-1},\Sigma^\omega) \qquad (51)\end{aligned}$$

where $\mathcal{F}_V$ is the lower bound. The closed-form expressions for the optimal variational bound can be derived as follows:

$$\begin{aligned}\mathcal{F}_V &= \log\mathcal{N}(\mathbf{Y};\mathbf{0},Q_{\mathbf{ZZ}}+\mathbf{I}\Sigma^\omega) - \frac{1}{2\Sigma^\omega}Tr(K_{\mathbf{ZZ}}-Q_{\mathbf{ZZ}})\\ &= \frac{N}{2}\log 2\pi + \frac{1}{2}\log|Q_{\mathbf{ZZ}}+\mathbf{I}\Sigma^\omega|\\ &+ \frac{1}{2}\mathbf{Y}^\top(Q_{\mathbf{ZZ}}+\mathbf{I}\Sigma^\omega)^{-1}\mathbf{Y} - \frac{1}{2\Sigma^\omega}Tr(K_{\mathbf{ZZ}}-Q_{\mathbf{ZZ}}) \qquad (52)\end{aligned}$$

with the optimal variation distribution $\phi^*(\mathbf{f}_z)$.

Then we can infer the approximation of the predictive Gaussian $p(y^*|\mathbf{Y},z^*)$ at the test point z^*:

$$\begin{aligned}q(y^*|\mathbf{Y},z^*) &= \int p(y^*|\mathbf{f}_z)p(\mathbf{f}|\mathbf{f}_z)\phi^*(\mathbf{f}_z)d\mathbf{f}d\mathbf{f}_z\\ &= \int p(y^*|\mathbf{f}_z)\phi^*(\mathbf{f}_z)d\mathbf{f}_z \qquad (53)\end{aligned}$$

with the assumption that y^* and $\mathbf{f}$ are independent, given $\mathbf{f}_z$.

The mean and covariance function of the approximate posterior GP are as follows:

$$\mu^v(z^*) = (\Sigma^\omega)^{-1}K_{z^*Z_m}(\mathcal{W})^{-1}K_{Z_mZ}\mathbf{Y} \qquad (54)$$

$$\Sigma^v(z^*) = K_{z^*z^*} - Q_{z^*z^*} + K_{z^*Z_m}(\mathcal{W})^{-1}k_{Z_mz^*} \qquad (55)$$

with $\mathcal{W} = K_{Z_mZ_m} + (\Sigma^\omega)^{-1}K_{Z_mZ}K_{ZZ_m}$. For n^d output dimensions, the VFE based Gaussian process can be denoted by

$$d_V \sim \mathcal{N}(\mu^V(z^*),\Sigma^V(z^*)) \qquad (56)$$

with $\mu^V = [\mu_1^v,\ldots,\mu_{n_d}^v]^\top$ and $\Sigma^V = \text{diag}([\Sigma_1^v,\ldots,\Sigma_{n_d}^v])$.

B. Comparison with State-of-the-Art Drift Control

To validate the drift and tracking ability of the proposed Two-layer Model Predictive Controller (TMPC) without sparse Gaussian error compensation, we compared it against three state-of-the-art drift controllers. These include the unconstrained nonlinear controller MARTY [9], the MPC controller DDTC [17], which simultaneously manages both drifting and trajectory tracking, and the MPC-HDDC controller [15], which combines nonlinear mapping with MPC based on the MARTY framework.

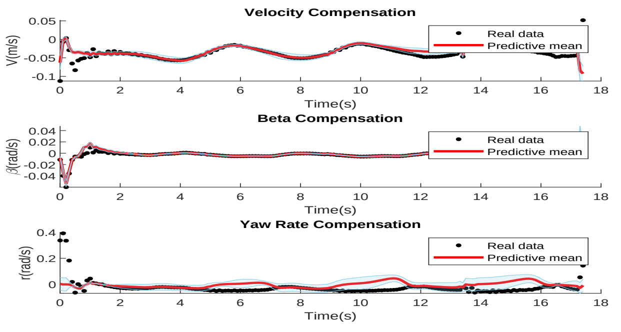
As shown in Fig. 10 and Fig. 11, all the controllers, represented by different colors, successfully track the trajectory while drifting. The comparison results of the controllers are presented in Fig. 11, Fig. 12, and Table VI. Since MARTY is a nonlinear controller that does not consider constraints when mapping control inputs, the vehicle's state, as depicted in Fig. 12, is less smooth compared to the other controllers. Its advantage is the use of numerical methods for nonlinear mapping, resulting in minimal computational time. The DDTC, which optimizes both drifting and trajectory tracking simultaneously, produces smoother state outputs. However, due to the need to consider both objectives simultaneously, its average lateral error and maximum lateral error are 0.89 m and 1.67 m, respectively, as shown in Table VI, which are 105% and 8% larger than those of the TMPC. MPC-HDDC separates trajectory tracking and drift state control, using a point mass model to formulate the MPC problem for trajectory tracking. Compared to MARTY, this approach results in smoother state transitions and enhanced tracking performance. However, the inherent nonlinear mapping of the drift state continues to obtain suboptimal control inputs, making its tracking performance inferior to that of TMPC. The results summarized in Table VI indicate that by decoupling the drifting and tracking problems, the TMPC achieved a 55%, 43%, and 51% reduction in average lateral error compared to MARTY, MPC-HDDC, and DDTC, respectively, demonstrating better trajectory tracking capability.

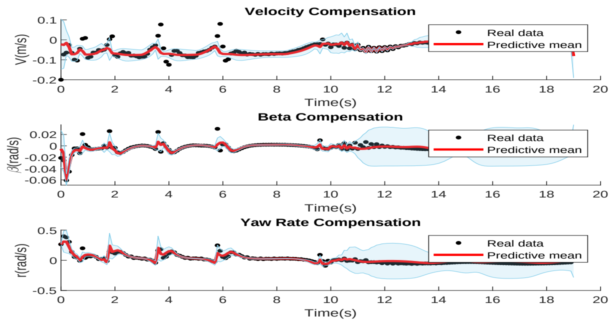
Table VI: Simulation Results of drift controllers.

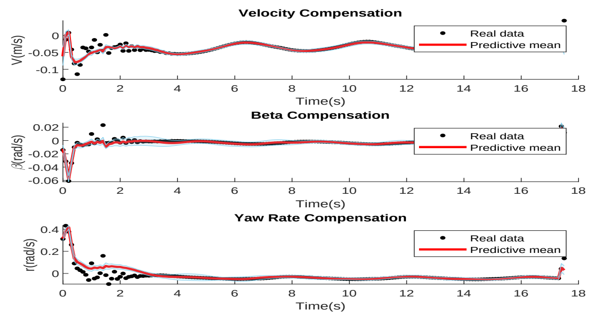
Controller	**Avg. e(m)**	**Avg. e Reduction (%)**	**Max. e(m)**	Computation time(ms)
MARTY [9]	0.9598	54.78	1.9221	**0.6759**
MPC-HDDC [17]	0.7679	43.49	1.8092	3.2329
DDTC [15]	0.8888	51.14	1.6743	2.7649
TMPC (**Ours**)	**0.4342**	-	**1.5481**	9.2895

C. Model Error Learning Comparsion

Regarding the comparison in model error learning, Fig. 9 demonstrates that compared to FGMPC, FCMPC, and SVMPC-1, the SVMPC prediction performance is better. Specifically, as shown in Table V for the GP section, the prediction errors of the three states for the SVMPC in the final lap are reduced by half compared to the third lap. The three GPs in the SVMPC cover 86.78%, 91.38%, and 83.33% of the actual error data within the 95% confidence interval at the end. FCMPC suffers from underfitting due to the prediction variance estimation. Although it covers the highest percentage of data, it has the largest prediction errors according to Table III. While FGMPC demonstrates more accurate predictions than those controllers in the sideslip angle dimension in Table II, the final prediction error of the sideslip angle is only 0.0001 m smaller than that of SVMPC and SVMPC-1. Moreover, FGMPC fails to meet real-time requirements (0.35 s), with its computation time being ten times that of SVMPC (0.03 s) as shown in Fig. 13.


(a) Full GP based error prediction in FGMPC.


(b) FITC based error prediction in FCMPC.


(c) Single VFE based error prediction in SVMPC-1 (**Ours**).

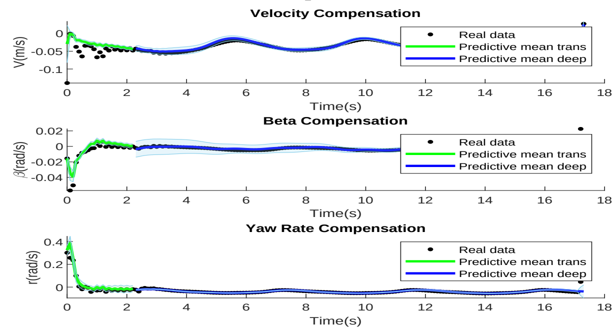

(d) Two local VFE based error prediction in SVMPC (**Ours**).

Figure 9: Comparison of GP prediction performance with 95% confidence level in different controllers at the final lap.

Figure 10: Animation in CarSim at time 11.5s. The green car represents HDDC, the blue car represents DDTC, the yellow car represents TMPC (**ours**), and the purple car represents MARTY.

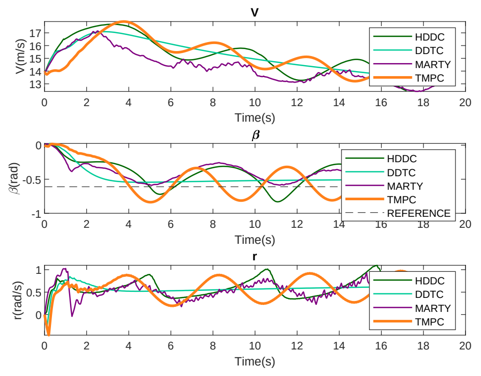


Figure 12: Vehicle states in CarSim simulation: TMPC (**ours**) vs. state-of-the-art controllers.

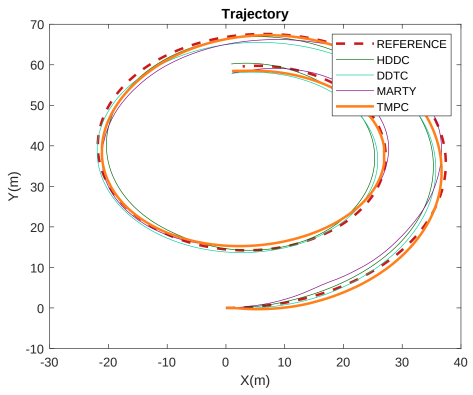


Figure 11: Trajectory tracking in CarSim simulation: TMPC (**ours**) vs. state-of-the-art controllers.

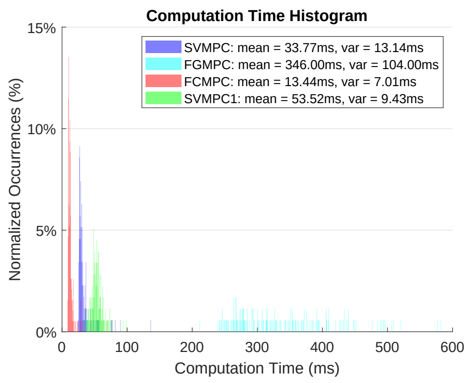


Figure 13: Computation time.