Smooth Model Predictive Path Integral Control Without Smoothing

Taekyung Kim, Gyuhyun Park, Kiho Kwak, Jihwan Bae, and Wonsuk Lee

***Abstract*—We present a sampling-based control approach that can generate smooth actions for general nonlinear systems without external smoothing algorithms. Model Predictive Path Integral (MPPI) control has been utilized in numerous robotic applications due to its appealing characteristics to solve non-convex optimization problems. However, the stochastic nature of sampling-based methods can cause significant chattering in the resulting commands. Chattering becomes more prominent in cases where the environment changes rapidly, possibly even causing the MPPI to diverge. To address this issue, we propose a method that seamlessly combines MPPI with an input-lifting strategy. In addition, we introduce a new action cost to smooth control sequence during trajectory rollouts while preserving the information theoretic interpretation of MPPI, which was derived from non-affine dynamics. We validate our method in two nonlinear control tasks with neural network dynamics: a pendulum swing-up task and a challenging autonomous driving task. The experimental results demonstrate that our method outperforms the MPPI baselines with additionally applied smoothing algorithms.**

***Index Terms*—Optimization and optimal control, planning under uncertainty, model learning for control, autonomous vehicle navigation, field robots.**

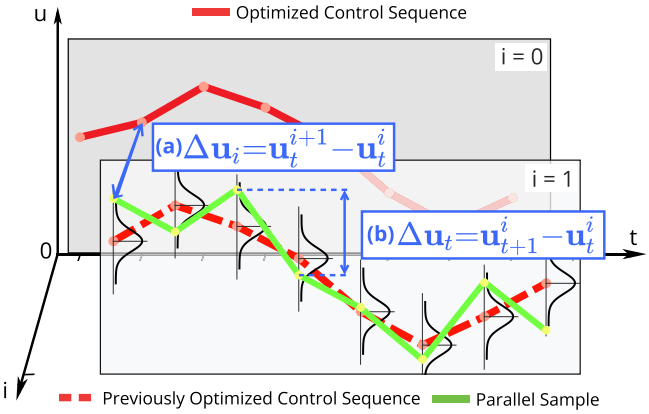

Fig. 1. A simplified representation of the MPPI algorithm during each optimization iteration. For clarity, we only visualize one sampled trajectory (in green). (a) Amount of changes between previously computed control sequence and the next control sequence (along the "i-axis"). (b) Amount of changes in control values during MPPI rollouts (along the "t-axis"), which are hard to be minimized by the MPPI baseline.

I. Introduction

CONTROLLING an autonomous vehicle in complex environments is a challenging problem. When the vehicle drives on a structured road, it can be modeled as a linear or a kinematic system that is easy to solve. However, the nature of the real world dictates non-linearity and this characteristic is highlighted more at high vehicle speeds and low road surface friction levels. The majority of autonomous driving research has been focused on normal driving conditions, while areas of aggressive maneuvering in highly nonlinear environments have not been fully addressed by prior work.

Gradient-based Model Predictive Control (MPC) methods have been introduced as powerful solutions for solving the problems of nonlinear control systems [1], [2]. Despite the fact that there have been successful real-world applications with these methods [3], [4], they still possess the limitation that the cost function has to be differentiable. Sampling-based Model Predictive Path Integral (MPPI) control has been proposed to optimize both convex and non-convex objectives [5] and has benefitted greatly from recent advances in Graphics Processing Units (GPUs), as a large number of samples can be computed in parallel to achieve better real-time performance.

While MPPI presents general performances in various situations, it may encounter chattering when implemented on an actual platform, which is a common characteristic of sampling-based methods [6]. Rapid changes in action commands are widely known to burden the actuators and cause the system to become unstable. Meanwhile, the key property of MPPI for achieving high-level control performance is to draw samples upon the previously optimized control sequence. However, this property can instead hinder the entire sequence from converging quickly to the optimal distribution when the environment changes too rapidly [7]. Under such circumstances, in particular, chattering becomes more prominent.

Therefore, a smoothing algorithm is often employed to smooth the control sequence in practice. However, because the optimization procedure and this external smoothing mechanism are completely decoupled, the smoothed sequence may unintentionally lose its optimality. In addition, it may present a

Manuscript received 24 February 2022; accepted 14 July 2022. Date of publication 21 July 2022; date of current version 4 August 2022. This letter was recommended for publication by Associate Editor A. Kuntz and Editor H. Kurniawati upon evaluation of the reviewers' comments. *(Corresponding author: Wonsuk Lee.)*

Taekyung Kim, Kiho Kwak, Jihwan Bae, and Wonsuk Lee are with the Ground Technology Research Institute, Agency for Defense Development, Daejeon 34186, Republic of Korea (e-mail: tkkim.robot@gmail.com; kkwak.add@gmail.com; mierojihan1008@gmail.com; wsblues82@gmail.com).

Gyuhyun Park is with the Department of Mechanical Engineering, Seoul National University, Seoul 08826, Republic of Korea, and also with the Ground Technology Research Institute, Agency for Defense Development, Daejeon 34186, Republic of Korea (e-mail: pkh8543@gmail.com).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2022.3192800, provided by the authors.

Digital Object Identifier 10.1109/LRA.2022.3192800

significant problem if the algorithm behaves incorrectly, even if it is tuned to perform well in most situations. In the worst case, the resulting sequence may violate the physical constraints or diverge.

In this letter, we propose the Smooth Model Predictive Path Integral (SMPPI) control method that combines MPPI with an input-lifting strategy to generate smooth actions without any additional smoothing algorithms. We include derivative actions as control inputs, in which this technique is commonly used to smooth jerky commands in control studies [8], [9]. To the best of our knowledge, we seamlessly apply this idea in the MPPI framework for the first time while preserving the information theoretic interpretation of optimal control. Additionally, our method offers a new smoothing mechanism which is ineligible in the MPPI baseline. The smoothing effects of our approach are twofold. First, it smooths the gap between the control sequences of previous and current iterations. We refer to this as smoothing along the "*i-axis*." Second, it smooths the optimized control sequence during trajectory rollouts. We refer to this as smoothing along the "*t-axis*." The amount of changes in the control values of each axis that can be reduced by our method are depicted in Fig. 1. Unlike MPPI, where efforts to realize smoothing along both axes would obstruct its ability to adapt to rapidly changing environments, SMPPI can achieve both smoothness and agility. We conduct experiments to compare our method with different smoothing methods in a classical control benchmark. We also demonstrate our idea in a challenging autonomous driving task.[1]

II. Background

This section provides a brief overview of the MPPI algorithm. The benefit of the sampling-based method is that it does not require a gradient of the objective function and the cost function [10]. Recent works including MPPI have integrated a sampling-based approach with MPC formulations to solve non-convex optimization problems [5], [6]. They combine information theoretic control and stochastic optimal control using free energy and the KL divergence to derive an optimal control distribution with importance sampling [11].

Consider a discrete time dynamic system with a state $\mathbf{x}_t \in \mathbb{R}^n$ and a control input $\mathbf{v}_t \in \mathbb{R}^m$. We applied the general noise assumption that holds that we could not directly control the system over $\mathbf{v}_t$, but could over the mean $\mathbf{u}_t$ of the density function of noise $\boldsymbol{\epsilon}_t$:

$$\mathbf{v}_t \sim \mathcal{N}(\mathbf{u}_t, \boldsymbol{\Sigma}).$$

Noise is simply defined as $\mathbf{v}_t = \mathbf{u}_t + \boldsymbol{\epsilon}_t$. Given a sequence of inputs $V = \{\mathbf{v}_0, \mathbf{v}_1, \dots \mathbf{v}_{T-1}\}$ and mean input variables $U = \{\mathbf{u}_0, \mathbf{u}_1, \dots \mathbf{u}_{T-1}\}$ along a finite time horizon $t \in \{0, 1, \dots T-1\}$, we can define the probability density function $q(V)$ as:

$$q(V) = \prod_{t=0}^{T-1} Z^{-1} \exp\left(-\frac{1}{2}(\mathbf{v}_t - \mathbf{u}_t)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{v}_t - \mathbf{u}_t)\right), \quad (1)$$

[1]Our video can be found at: https://youtu.be/fyngK8PCoyM

where $Z = ((2\pi)^m |\boldsymbol{\Sigma}|)^{\frac{1}{2}}$. Similarly, we can also define the uncontrolled density function $p(V)$ where U is usually 0:

$$p(V) = \prod_{t=0}^{T-1} Z^{-1} \exp\left(-\frac{1}{2}\mathbf{v}_t^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \mathbf{v}_t\right). \quad (2)$$

These two density functions correspond to the distributions of $\mathbb{Q}$ and $\mathbb{P}$, respectively. Here, we define an optimal density function using the free energy of the system [11], which corresponds to the optimal distribution $\mathbb{Q}^*$:

$$q^*(V) = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda} S(V)\right) p(V), \quad (3)$$

where η denotes the normalizing constant and $S(V)$ denotes the state-dependent cost. The input sequence in state cost is iteratively transformed into state values $\mathbf{x}$ through the non-affine system dynamics $\mathbf{F}$ [6]:

$$S(V; \mathbf{F}) = \phi(\mathbf{x}_T) + \sum_{t=0}^{T-1} c(\mathbf{x}_t),$$
$$\mathbf{x}_{t+1} = \mathbf{F}(\mathbf{x}_t, \mathbf{v}_t). \quad (4)$$

As proposed in [6], we can now derive the optimal control input by minimizing the KL divergence between $\mathbb{Q}$ and $\mathbb{Q}^*$:

$$\mathbb{E}_{\mathbb{Q}^*}[\mathbf{v}_t] = \int q^*(V) \mathbf{v}_t \mathrm{d}V. \quad (5)$$

Here, the best solution of (5) is to draw samples directly over $\mathbb{Q}^*$, but this is not possible. Importance sampling is therefore employed to compute the integral over the known distribution $\mathbb{Q}$:

$$\mathbb{E}_{\mathbb{Q}^*}[\mathbf{v}_t] = \int w(V) q(V) \mathbf{v}_t \mathrm{d}V$$
$$= \mathbb{E}_{\mathbb{Q}_{U,\boldsymbol{\Sigma}}}[w(V)\mathbf{v}_t], \quad (6)$$

where the importance weighting term is:

$$w(V) = \left(\frac{q^*(V)}{p(V)}\right)\left(\frac{p(V)}{q(V)}\right)$$
$$= \frac{1}{\eta} \exp\left(-\frac{1}{\lambda}\left(S(V) + \frac{1}{2}\lambda \sum_{t=0}^{T-1} \mathbf{u}_t^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{u}_t + 2\boldsymbol{\epsilon}_t)\right)\right). \quad (7)$$

Let $\mathcal{E}$ be a noise sequence $\{\boldsymbol{\epsilon}_0, \boldsymbol{\epsilon}_1, \dots \boldsymbol{\epsilon}_{T-1}\}$ and K be the number of trajectory samples. Finally, we have the iterative optimal control update law to compute the weights when the sampled trajectory cost $\{C(V^0), C(V^1), \dots C(V^{K-1})\}$ is given:

$$\mathbf{u}_t^{i+1} = \mathbf{u}_t^i + \sum_{k=0}^{K-1} w(\mathcal{E}^k) \boldsymbol{\epsilon}_t^k \quad (8)$$

$$w(\mathcal{E}^k) = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda}\left(C(V^k) - \beta\right)\right), \quad (9)$$

where we subtract the minimum state cost β to ensure that at least one sample has a numerically non-zero importance sampling

weight. The trajectory cost in (9) is then simplified as follows:

$$C(V^k) = S(V^k) + \lambda \sum_{t=0}^{T-1} \mathbf{u}_t^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{\epsilon}_t^k. \tag{10}$$

III. Smooth Model Predictive Path Integral Control

A. Limitations of External Smoothing Algorithms

The sampling-based approach of the MPPI algorithm makes possible the consideration of involving non-convex objectives in optimization problems. However, the stochastic nature of this method can cause significant chattering in the resulting commands. Due to the unique property by which perturbed trajectories are sampled around the previously optimized mean control sequence U, MPPI cannot respond effectively when the optimal control distribution greatly deviates from the previous iteration. Undesirable chattering effects stand out more in the scenarios mentioned above.

A common approach to mitigate such a problem is to smooth the resulting control sequence using, for instance, sliding window smoothing methods or filtering based methods [12], [13]. It was suggested that a Savitzky-Golay filter (SGF) [14] would be an effective solution for this control method [6]. It smooths the subsets of adjacent data by fitting the local polynomial approximations in a convolutional manner.

However, this type of approach can have negative effects, as external manipulations of control values are not considered in the optimization process. Consider a general state-action system dynamics with clamping function $\mathbf{g_u}$ for handling control constraints:

$$\mathbf{x}_{t+1} = \mathbf{F}\left(\mathbf{x}_t, \mathbf{g}_{\mathbf{u}}(\mathbf{v}_t)\right). \tag{11}$$

Because the perturbed control $\mathbf{v}_t$ is bounded by the physical limits of the actuators, the sampled noise should be bounded as follows:

$$\bar{\boldsymbol{\epsilon}}_t = \mathbf{g}_{\mathbf{u}}(\mathbf{v}_t) - \mathbf{u}_t. \tag{12}$$

Two possible ways exist to apply the SGF in the control sequence.

1) Smoothing Weighted Noise Sequence: First, we can apply smoothing to the weighted noise sequence:

$$\mathbf{u}_t^{i+1} = \mathbf{u}_t^i + \mathrm{SGF}\left(\sum_{k=0}^{K-1} w(\bar{\mathcal{E}}^k)\bar{\boldsymbol{\epsilon}}_t^k\right). \tag{13}$$

This makes data points smoother while compensating for the noise, which changes considerably along the time horizon. During this procedure, the control variable $\mathbf{u}$ may violate the constraint conditions given that the bounded noises are manipulated by the filter. On the other hand, the nature of this process makes it vulnerable to phase distortion [15]. In addition to violating the given constraints, the control sequence will diverge if the noises are overlapped repeatedly and amplified over time. We describe this phenomenon further in Section V-D.

2) Smoothing Control Sequence: Second, we can apply the SGF after the control sequence is updated using the weighted noise:

$$\mathbf{u}_t^{i+1} = \mathrm{SGF}\left(\mathbf{u}_t^i + \sum_{k=0}^{K-1} w(\bar{\mathcal{E}}^k)\bar{\boldsymbol{\epsilon}}_t^k\right). \tag{14}$$

Although this method guarantees that the control variables are meeting the bounded condition, it can cause a delay in the system response. It is common to provide history values as they are required in the convolution-based smoothing method. The early values in the control sequence are then affected by the history, which means that polynomial approximation hinders the controlled distribution from rapidly shifting to an optimal distribution [16]. Note that the first control action is the key value in the MPC problem.

B. Decoupling Control Space and Action Space

In the original MPPI, the running control cost in (10) only plays a role in reducing the distance between control values of the current and previous iterations [17]. Chattering innately occurs because the control trajectory is sampled randomly, and the variance along the "t-axis" in the new control trajectory is not taken into account during optimization. To this end, it is possible to impose an additional cost in terms of the variation of the input, i.e., chattering on the "t-axis." [18] suggested an additional running cost for variation of the inputs, but it cannot be applied generally, as it assumes that the dynamics are affine in control under special conditions. We intend to expand the MPPI baseline [6], which is likely to be adopted more extensively, but arbitrary modifications for the additional cost associated with input sequences violate the theoretical derivation of MPPI as the control cost is determined inherently through the importance sampling scheme.

In order to handle such an issue while preserving the MPPI framework, therefore, we suggest the lifting of the control variables as derivative actions. This strategy can mitigate the chattering problem in terms of two points of view.

1) Smoothing Along the "i-Axis": Henceforth, let us decouple the control space and action space by defining action sequence $A = \{\mathbf{a}_0, \mathbf{a}_1, \ldots \mathbf{a}_{T-1}\}$. Noisy sampling is performed on a higher order control space U. Then, the resulting control sequence is integrated to become a smoother action sequence. The control distribution is hereby distinguished from the MPPI baseline. The variance of the injected noise is adjusted according to the physical limit of the input rate of change. This in turn reduces the variation along the "i-axis" and consequently mitigates damage to the actuator.

Note that one can attempt to smooth resulting actions in the MPPI framework by reducing the variance in the same manner as described above. In this setting, however, MPPI would be unable to respond to rapidly changing environments. On the other hand, increasing the variance to cover a wider action range would not only be physically implausible but would also eventually result in chattering. The method proposed here is free from this trade-off, as its new search space U can be handled adaptively depending on the environment.

2) Smoothing Along the "t-Axis": In contrast to MPPI, our control distribution corresponds to derivative action, allowing

Algorithm 1: SMPPI.

Given: $\mathbf{F}$: Dynamics model;
K, T: Number of samples, timesteps;
$(\mathbf{u}_0, \mathbf{u}_1, \dots \mathbf{u}_{T-1})$: Initial control sequence;
$(\mathbf{a}_0, \mathbf{a}_1, \dots \mathbf{a}_{T-1})$: Initial action sequence;
$\mathbf{\Sigma}, \lambda, \phi, c, \Omega$: Control parameters and cost functions;
for $i \leftarrow 0$ **to** *maximum iterations* **do**
  $\mathbf{x}_0 \leftarrow$ SubscribeState();
  for $k \leftarrow 0$ **to** $K-1$ **do**
    $\mathbf{x} \leftarrow \mathbf{x}_0$;
    Sample $\mathcal{E}^k = \left(\boldsymbol{\epsilon}_0^k \dots \boldsymbol{\epsilon}_{T-1}^k\right)$, $\boldsymbol{\epsilon}_t^k \in \mathcal{N}(0, \mathbf{\Sigma})$;
    for $t \leftarrow 0$ **to** $T-1$ **do**
      $\mathbf{v}_t^k = \mathbf{u}_t + \boldsymbol{\epsilon}_t^k$;
      $\mathbf{a}_t^k = \mathbf{a}_t + \mathbf{v}_t^k \Delta t$;
      $\mathbf{x} \leftarrow \mathbf{F}(\mathbf{x}, \mathbf{a}_t^k)$;
      $C_k \mathrel{+}= c(\mathbf{x}) + \lambda \mathbf{u}_t^{\mathrm{T}} \mathbf{\Sigma}^{-1} \boldsymbol{\epsilon}_t^k$;
    $C_k \mathrel{+}= \phi(\mathbf{x}) + \Omega(\mathbf{a}_0^k, \mathbf{a}_1^k, \dots \mathbf{a}_{T-1}^k)$;
  $w_k \leftarrow$ ComputeWeights$(C_0, C_1, \dots C_{k-1})$;
  for $t \leftarrow 0$ **to** $T-1$ **do**
    $U^{i+1} \leftarrow U^i + \left(\sum_{k=0}^{K-1} w_k^{\mathrm{T}} \mathcal{E}^k\right)$;
    $A^{i+1} \leftarrow A^i + U^{i+1} \Delta t$;
  SendToController$(\mathbf{a}_0)$;
  for $t \leftarrow 1$ **to** $T-1$ **do**
    $\mathbf{u}_{t-1}, \mathbf{a}_{t-1} \leftarrow \mathbf{u}_t, \mathbf{a}_t$;
  $\mathbf{u}_{T-1}, \mathbf{a}_{T-1} \leftarrow$ Initialize$(\mathbf{u}_{T-1}, \mathbf{a}_{T-1})$;

action variables to be treated as augmented state elements and hence making them independent of the control cost. Variation of action can now be easily included in the state cost such that chattering with regard to the "t-axis" can be minimized. Here, we introduce an extra action cost Ω to smooth the action sequence along the "t-axis" by minimizing the variance of A without violating the information theoretic interpretation of MPPI:

$$\Omega(A) = \sum_{t=1}^{T-1} (\mathbf{a}_t - \mathbf{a}_{t-1})^{\mathrm{T}} \boldsymbol{\omega} (\mathbf{a}_t - \mathbf{a}_{t-1}), \tag{15}$$

where $\boldsymbol{\omega}$ is the weighting parameter in the form of a diagonal matrix.

Finally, we obtain the following action sequence update law:

$$\mathbf{u}_t^{i+1} = \mathbf{u}_t^i + \sum_{k=0}^{K-1} w(\mathcal{E}^k) \boldsymbol{\epsilon}_t^k, \tag{16}$$

$$\mathbf{a}_t^{i+1} = \mathbf{a}_t^i + \mathbf{u}_t^{i+1} \Delta t, \tag{17}$$

and the trajectory cost (10) now takes the form:

$$C(V^k, A) = S(V^k) + \Omega(A + V^k \Delta t) + \lambda \sum_{t=0}^{T-1} \mathbf{u}_t^{\mathrm{T}} \mathbf{\Sigma}^{-1} \boldsymbol{\epsilon}_t^k. \tag{18}$$

Afterwards, we apply clamping function $\mathbf{g_a}$ to the action commands to impose box constraints on the vehicle's physical limits. The overall algorithm of SMPPI is shown in Alg. 1.

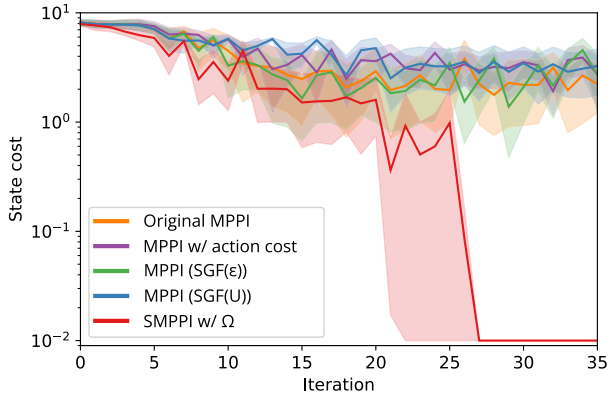

Fig. 2. Average state costs during optimization procedures with different control methods. Each iteration consists of 20 time steps. Note that we apply logarithmic scaling to the y-axis and that the costs are clamped to a minimum of 0.01.

IV. Experiments on an Inverted Pendulum

We tested MPPI and SMPPI in a simulated inverted pendulum swing-up task. One of the important characteristics of the pendulum swing-up task is that the available torque is insufficient to push the pendulum to reach an upright position in a single rotation. The controller should swing the pendulum multiple times to gather energy, and during this process, the optimal control sequence changes rapidly. Consequently, it is a simple yet effective benchmark for evaluating our method. To exploit the fact that our idea is generalized to allow non-affine dynamics, we used a neural network to learn the dynamics model in this experiment.

We designed the dynamics model with two fully-connected hidden layers, each with 32 neurons. The state-action dataset was repeatedly collected during the control procedure and the model was trained with the dataset every 50 time steps. No bootstrap dataset was used because the dimensionality of the inverted pendulum system is sufficiently low. The running state cost function $c(\mathbf{x})$ in (4) is formulated as follows:

$$c(\mathbf{x} \equiv [\theta, \dot{\theta}]^{\mathrm{T}}) = \theta^2 + 0.1\dot{\theta}^2. \tag{19}$$

We compared five different methods: i) the original MPPI without smoothing (abbreviated as "MPPI baseline"), ii) applying an additional action cost for smoothing on MPPI, iii) applying the SGF on the noise sequence (abbreviated as "MPPI (SGF($\boldsymbol{\epsilon}$))"), iv) applying the SGF on the control sequence (abbreviated as "MPPI (SGF(U))"), and v) SMPPI. In this task, control parameter λ was set to 10. For SMPPI, the action sequence cost parameter $\boldsymbol{\omega}$ was set to 1. They were tested with seven different initial angular velocities ($\dot{\theta} \in \{-3, -2, \dots 3\}$) starting from the initial position vertically downward. We fixed the random seed for a fair comparison. The state costs (19) during online optimization are shown in Fig. 2.

The results show that only SMPPI successfully stabilized the pendulum in the upright position in all cases. As seen in our video, the other methods showed similar results. They continually failed to push the pendulum to the upright position because they did not apply enough torque during the swinging motions.

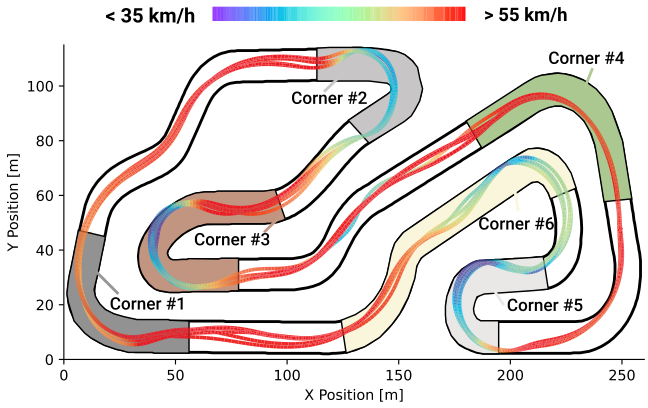

Fig. 3. Trajectory of our proposed method with a reference speed of 60 km/h.

In the case of MPPI baseline i), it was vulnerable to chattering and could not converge into an optimal control sequence when the optimal distribution changed rapidly. Imposing an additional action cost ii), which was employed to alleviate chattering, also failed because this strategy violates the information theoretic interpretation of MPPI while using non-affine dynamics. Methods using external smoothing algorithms iii)−iv) failed due to the negative effects of external smoothing. We discuss these issues further in Section V-D.

V. Experiments on a Vehicle Simulator

A. Experimental Setup

Our research goal is to perform high-speed driving successfully under challenging conditions, including those with unknown friction surfaces and sharp corners. The performance of the model-based control policy is highly dependent on the accuracy of the model [19], [20]. The vehicle model should be trained with all possible maneuvers to minimize any model bias. Because the control sequence update is processed by evaluating the perturbed control trajectories with sampling, situations in which control values change significantly over time that make the vehicle unstable should be included in the training dataset. Obtaining such data from an actual vehicle would be dangerous. Therefore, we used CarMaker to collect driving data and to evaluate the control performances. This is a widely used, high-fidelity vehicle simulator that precisely solves nonlinear dynamics in real time.

We built a race track modeled after an actual kart circuit known as "KART 2000" in Kirchlengern, Germany (see Fig. 3). The length of the track was 1016 m and it had two moderate curves and four sharp curves. Note that when the vehicle reaches the entry of sharp corners, the controller should reduce the vehicle's speed and apply a large steering angle quickly. Otherwise, the vehicle will understeer and collide with the track boundary.

In some prior research on autonomous driving on race tracks, electric vehicles without transmissions were used [4], [6], [8]. Unlike these vehicles, the majority of current vehicles use an automatic transmission to shift gears. The dynamic characteristics of the vehicle will definitely change as a gear shifts. Therefore, throttle control is not feasible for use in common vehicles. Here, we simply overcome this issue by taking desired speed (v_{des}) as the feedforward command of MPPI. A low-level feedback controller then manages the throttle and brake via a Proportional-Integral (PI) loop. A Volvo XC90 was used as the control vehicle.

TABLE I
Overall Architecture of the Vehicle Model

	Size	Activation
Input	$(\lvert\mathbf{x}\rvert + \lvert\mathbf{a}\rvert) \times h$	-
Hidden Layer 1	2×Input Size	ReLU
Hidden Layer 2	4×Input Size	ReLU
Hidden Layer 3	6×Input Size	ReLU
Hidden Layer 4	2×Input Size	Linear
Output	$\lvert\mathbf{x}\rvert$	-

B. Training the Neural Network Vehicle Model

There are two main characteristics that make the vehicle dynamics challenging to model: First, vehicle motions under unknown friction are difficult to predict. In particular, the lateral tire forces will slip into the nonlinear region on low friction corners [21]. Second, the automatic gear shifting mechanism is highly complex to model explicitly. We propose to model such complex and nonlinear dynamics using a neural network with state-action history, which enables the network to capture time-varying behavior [22]. The overall structure of our vehicle model, which is a feedforward fully-connected neural network, is shown in Table I. The terms $|\mathbf{x}|$ and $|\mathbf{a}|$ denote correspondingly the size of the state and the action space, and h represents the number of state-action pairs in the previous history. A Rectified Linear Unit (ReLU) is used for the activation of hidden layers. The Mean Squared Error (MSE) loss was used as the loss function and the Adam optimization was used for mini-batch gradient descent.

We collected a human-controlled driving dataset in a manner similar to that utilized in our prior work on the training of a dynamics model [23]. The vehicle state is defined as $\mathbf{x} = [v_x, v_y, r]^{\mathrm{T}}$, where v_x and v_y are the longitudinal and lateral velocities, and r is the yaw rate. The action command is defined as $\mathbf{a} = [\delta, v_{des}]^{\mathrm{T}}$, where δ is the steering angle command. We carefully selected three distinct maneuvers:

i) Zig-zag driving at low speeds ($20 - 25$ km/h) on the race track.
ii) High-speed driving on the race track while trying to maintain a speed of 40 km/h as much as possible.
iii) Sliding maneuvers during combinations of acceleration and deceleration with small, medium, and large steering angles. Subsequently, we controlled the vehicle with rapidly changing commands for random movements, representing the sampled noisy trajectories of MPPI. These maneuvers were performed on flat ground in both the left and right directions.

Each maneuver was done with multiple road friction coefficients $\mu \in \{0.4, 0.5, \ldots 1.0\}$. The maneuvers on the race track were done in both clockwise and counter-clockwise directions. Each one was logged for two minutes. We collected a total

TABLE II
TRAINING RESULTS OF THE NEURAL NETWORK VEHICLE MODEL

	v_x [m/s]		v_y [m/s]		r [rad/s]	
	$\mathbf{E}_{RMS}$	$\mathbf{E}_{max}$	$\mathbf{E}_{RMS}$	$\mathbf{E}_{max}$	$\mathbf{E}_{RMS}$	$\mathbf{E}_{max}$
Test	0.0311	0.5530	0.0216	0.4723	0.0175	0.2506
Val.	0.0252	0.3744	0.0136	0.1500	0.0123	0.1400

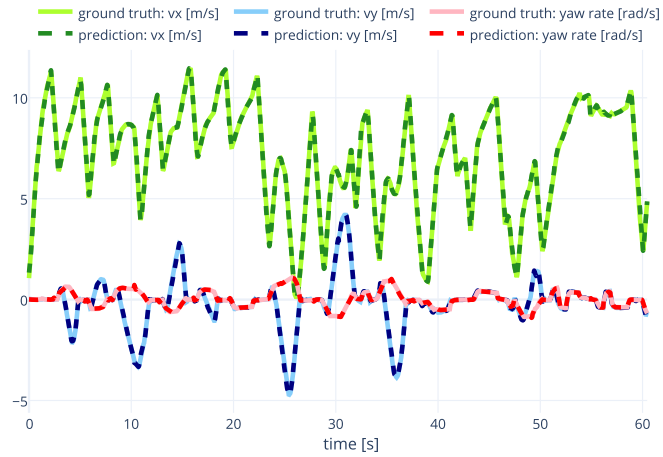

Fig. 4. Estimation results on the validation dataset.

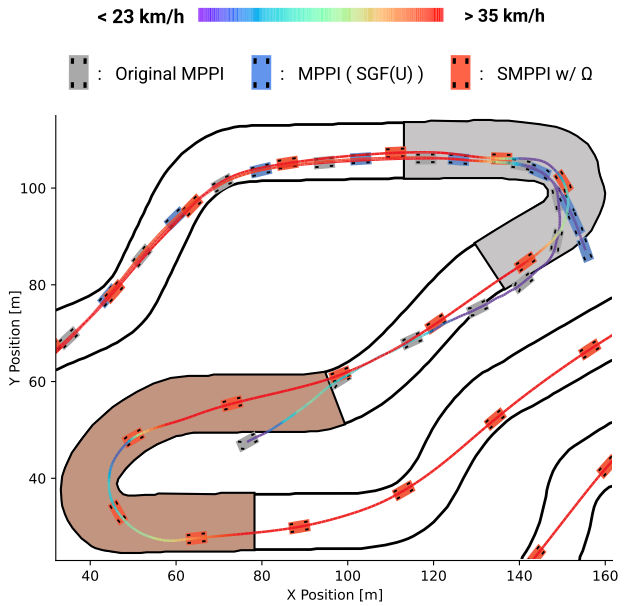

Fig. 5. Visualization of trajectories of the compared controllers. The friction coefficient of *corner #2* (in gray) is 0.85 and that of *corner #3* (in brown) is 0.75.

of 35 maneuvers, which amounted to 70 minutes of driving. The dataset was split into two portions: 70% for training and 30% for testing. To evaluate the generalization performance of the trained model, the validation dataset was collected on the same race track while the friction coefficients were modified. From *Corner #1* to *Corner #6* (depicted in Fig. 3), the friction coefficients were assigned with values that were not included in the training dataset: $[0.95, 0.85, 0.75, 0.65, 0.55, 0.45]$, respectively. The other regions were assigned values of $\mu = 0.8$.

The test and validation errors after training are shown in Table II. The results show that our trained model can precisely represent the vehicle system dynamics regardless of the road friction. The estimation results on the validation data are visualized in Fig. 4. The Root Mean Square Error (RMSE) is denoted as $\mathbf{E}_{RMS}$ and the max error is denoted as $\mathbf{E}_{max}$.

C. Cost Function and Other Parameters

Following prior work [6], we designed a state-dependent cost function $c(\mathbf{x})$, which consisted of three components:

$$c(\mathbf{x}) = \alpha_1 \text{Track}(\mathbf{x}) + \alpha_2 \text{Speed}(\mathbf{x}) + \alpha_3 \text{Slip}(\mathbf{x}). \tag{20}$$

The track cost indicates whether the vehicle is inside the track or outside of the given boundary. The states of the sampled trajectories are transformed into global positions (p_x, p_y) and they lookup the values of a two-dimensional cost map $\mathbf{M}$, which represents the area of the outer boundary as 1 and the area of the inner boundary as 0. Our sampling-based controller allows this impulse-like cost function:

$$\text{Track}(\mathbf{x}) = (0.9)^t \, 10000 \mathbf{M}(p_x, p_y). \tag{21}$$

The speed cost is the quadratic cost to achieve the reference vehicle speed:

$$\text{Speed}(\mathbf{x}) = (v_x - v_{ref})^2. \tag{22}$$

The slip cost penalizes the sideslip angle to plan a stable future trajectory. Additionally, it imposes a hard cost to reject samples which are estimated to have a larger sideslip angle than 0.2 rad (approximately 11.46°) in the future:

$$\text{Slip}(\mathbf{x}) = \sigma^2 + 10000 I\left(\{|\sigma| > 0.2\}\right)$$
$$\sigma = -\arctan\left(\frac{v_y}{\|v_x\|}\right), \tag{23}$$

where I is an indicator function.

We conducted a grid search for each controller to identify the best control parameters exhibiting the most successful performance. The sampling variance was adjusted to cover most of the control range of each controller. MPPI used $\mathbf{\Sigma} = \text{Diag}(4.0, 3.0)$ for the action noise and SMPPI used $\mathbf{\Sigma} = \text{Diag}(0.7, 0.4)$ for the derivative action noise. The feedforward control frequency was 10 Hz and the controllers used a time horizon of 4 s. The control parameters K and λ were correspondingly set to 10000 and 15.0 during the experiments. For SMPPI, it used $\boldsymbol{\omega} = \text{Diag}(0.8, 0.8)$.

D. Experimental Results

We conducted an ablation study for our framework. The track used in these experiments is identical to that used to collect the validation data. We measured the average lap time taken by the vehicle to traverse each corner during five laps around the track, where the reference speed was set to 40 km/h. If the vehicle went outside a boundary, it was placed at the start point and had to start a new lap. The same pre-trained vehicle model was used and there was no extra training during experiments. The results are shown in Table III.

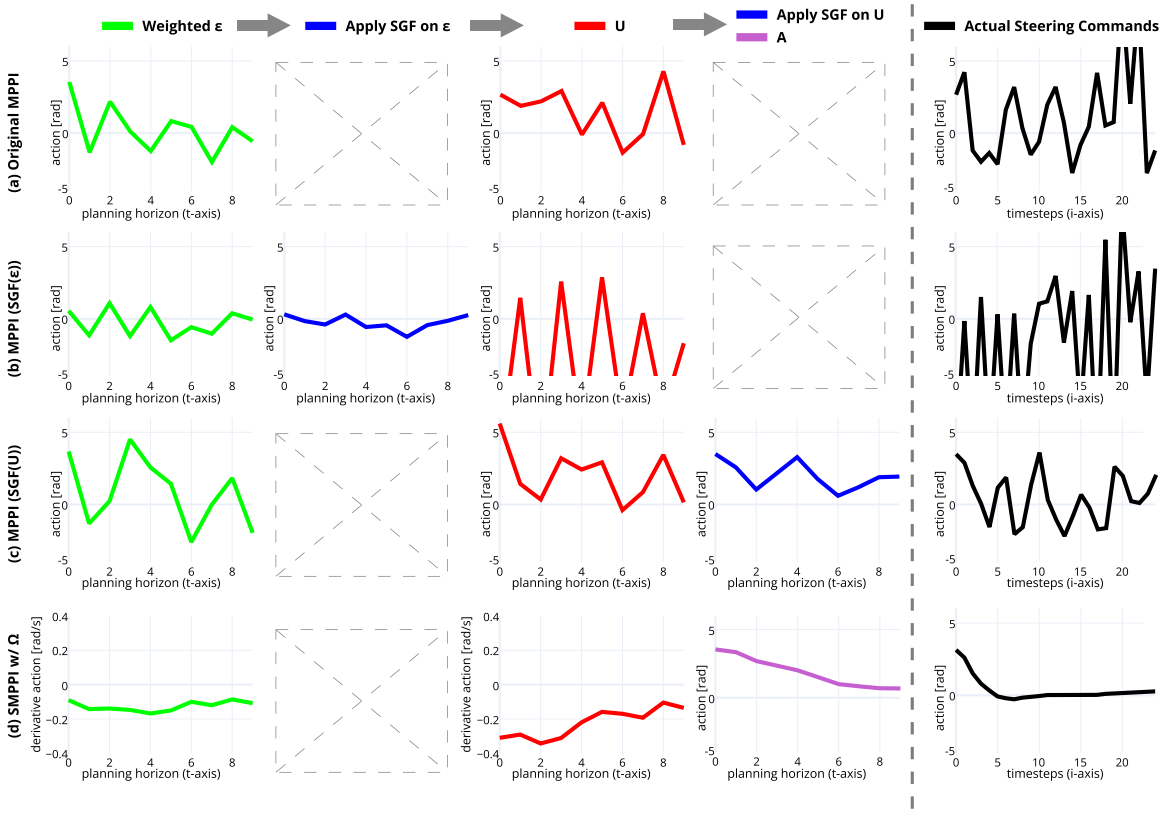

Fig. 6. The first 10 steps of the weighted sum of noise sequences (in green) and action sequences (in red) from different controllers right after entering *Corner #2* ($t = i = 0$ at this instant). For clarity, only the steering angle commands are visualized. The results after applying the SGF filter are shown in blue. For SMPPI, the augmented action sequence is shown in purple. The actual steering commands applied to the vehicle for 25 steps (2.5 s) are shown on the right-hand side (in black).

TABLE III
AVERAGE LAP TIMES (IN [S]) ON SIX CORNERS WITH DIFFERENT CONTROL METHODS. FOR SMPPI WITH Ω, THE MINIMUM SPEED (IN [KM/H]) AND THE MAXIMUM SLIP ANGLE (IN [°]) WHILE PASSING THROUGH EACH CORNER ARE ALSO ANALYZED BELOW

Lap time [s]	#1	#2	#3	#4	#5	#6
Original MPPI	11.08	9.60	N/A	N/A	N/A	N/A
MPPI (SGF(ϵ))	11.75	9.90	N/A	N/A	N/A	N/A
MPPI (SGF(U))	7.82	N/A	N/A	N/A	N/A	N/A
SMPPI w/o Ω	8.39	7.71	11.64	8.77	9.86	13.11
SMPPI w/ Ω	7.36	7.66	10.83	8.47	9.39	12.86
Min. Speed [km/h]	24.71	20.37	23.30	36.51	28.24	29.77
Max. Slip [°]	5.34	11.09	9.64	2.91	5.32	4.94

While all methods showed promising results on straight roads and on *Corner #1*, MPPI with different control update laws (8), (13), (14) failed to complete *Corner #2* and *Corner #3*, which are the sharpest corners of the track. The vehicles collided with the boundary or lost control after entering these turns. In contrast, both of the proposed methods successfully controlled the vehicle to traverse all of the laps at high speeds. One notable finding is that SMPPI with the additional action cost Ω showed faster lap times in all corners than SMPPI without an action cost. This suggests that reducing chattering along the "t-axis" can improve the control performance on the MPPI structure. Clearly, reducing the unnecessary noise in the steering and throttle commands is strongly related to increasing the speed of the vehicle. The trajectories taken by the three controllers are shown in Fig. 5.

Finally, we examined in more detail the actual optimization procedure of each method to verify our idea. The sequences computed by the four controllers right after taking *Corner #2* are visualized in Fig. 6. While passing through this sharp corner, the neural network was able to estimate that the vehicle would slide off the track if it did not slow down due to the friction limits. Therefore, the controller was required to apply the brakes and a large positive steering angle to make an appropriate right turn. During this procedure, the gap between the optimal and the current control sequence increased in an instant. The control sequence in the MPPI baseline attempted to respond to the rapidly changing optimal distribution. However, chattering occurred (see Fig. 6(a)) and caused the vehicle to lose its stability (see Fig. 5). This is because perturbed samples incorporating high-frequency noise survived after importance sampling, to minimize the impulse-like state cost despite the chattering. In the case of "MPPI (SGF(ϵ))," the external smoothing filter distorted the phase of the weighted noise. The noise sequence then escalated the chattering into the control sequence and finally led to divergence (see Fig. 6(b)). On the other hand, smoothing was suitably applied in "MPPI (SGF(U))," but this rather hindered the controller from quickly generating the correct command (see Fig. 6(c)). This behavior is also well illustrated in Fig. 5. In contrast, SMPPI successfully responded to the environment and generated a smooth action sequence without the need for an additional smoothing filter. The derivative action sequence still possessed inevitable chattering, but it was integrated to

become a smoother action sequence. The actual commands applied to the vehicle were also shown to be stable (along the "i-axis"), an achievement realized due to the strong benefits of SMPPI.

VI. Conclusion

We presented the Smooth Model Predictive Path Integral algorithm, which is designed to generate smooth control commands within the MPPI algorithm. Our method can attenuate chattering, which is a natural problem of sampling-based algorithms, without the need to design external smoothing filters. In particular, our method outperforms the MPPI baseline in cases in which the environment changes rapidly. In addition, our proposed method is not confined to employing action smoothing cost, as the control domain is shifted to the derivative action distribution. We showed that SMPPI can improve the general performance when it is applied to autonomous driving tasks. SMPPI was demonstrated to be capable of controlling an autonomous vehicle with agility on sharp and slippery corners. Furthermore, this chattering-free controller is also beneficial in reducing damage to the actuators.

Appendix

We also demonstrated our method on a more aggressive driving task. The reference speed was set to 60 km/h. We collected an extra training dataset of high-speed maneuvers on the race track, following the strategy described in Section V. Then, we trained the vehicle model with the augmented dataset. Finally, the SMPPI controller was deployed on the validation race track. From *Corner #1* to *Corner #6* (depicted in Fig. 3), the friction coefficients were assigned the following corresponding values: $[1.0, 0.95, 0.9, 0.85, 0.8, 0.75]$. The other regions were set such that $\mu = 0.9$. All five laps were completed at high speeds on challenging roads with varying amounts of friction. The trajectory taken by the vehicle is visualized in Fig. 3. We were able to observe the controller taking "out-in-out" trajectories on sharp corners because this is the best way to preserve high speeds and prevent large slip angles according to the predictions of the neural network model. We encourage readers to watch our supplementary video, where all of our experiments are shown in detail.

References

[1] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *Proc. Int. Conf. Inform. Control, Automat. Robot.*, 2004, pp. 222–229.

[2] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 1168–1175.

[3] U. Rosolia and F. Borrelli, "Learning how to autonomously race a car: A predictive control approach," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 6, pp. 2713–2719, Nov. 2020.

[4] J. Kabzan et al., "AMZ driverless: The full autonomous racing system," *J. Field Robot.*, vol. 37, no. 7, pp. 1267–1294, 2020.

[5] G. Williams et al., "Information theoretic MPC for model-based reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 1714–1721.

[6] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1603–1622, Dec. 2018.

[7] J. Yin, Z. Zhang, E. Theodorou, and P. Tsiotras, "Trajectory distribution control for model predictive path integral control using covariance steering," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 1478–1484.

[8] J. L. Vázquez, M. Brühlmeier, A. Liniger, A. Rupenyan, and J. Lygeros, "Optimization-based hierarchical motion planning for autonomous racing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 2397–2403.

[9] E. Chisari, A. Liniger, A. Rupenyan, L. Van Gool, and J. Lygeros, "Learning from simulation, racing in reality," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 8046–8052.

[10] M. Kobilarov, "Cross-entropy motion planning," *Int. J. Robot. Res.*, vol. 31, no. 7, pp. 855–871, 2012.

[11] E. A. Theodorou and E. Todorov, "Relative entropy and free energy dualities: Connections to path integral and KL control," in *Proc. IEEE 51st Conf. Decis. Control*, 2012, pp. 1466–1473.

[12] S. Särkkä, "Unscented rauch-tung-striebel smoother," *IEEE Trans. Autom. Control*, vol. 53, no. 3, pp. 845–849, Apr. 2008.

[13] H.-C. Ruiz and H. J. Kappen, "Particle smoothing for hidden diffusion processes: Adaptive path integral smoother," *IEEE Trans. Signal Process.*, vol. 65, no. 12, pp. 3191–3203, Jun. 2017.

[14] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Anal. Chem.*, vol. 36, no. 8, pp. 1627–1639, 1964.

[15] K. Adhikari, S. Tatinati, K. C. Veluvolu, and J. A. Chambers, "Physiological tremor filtering without phase distortion for robotic microsurgery," *IEEE Trans. Automat. Sci. Eng.*, vol. 19, no. 1, pp. 497–509, Jan. 2022.

[16] J. Wang, Y. Ye, X. Pan, X. Gao, and C. Zhuang, "Fractional zero-phase filtering based on the Riemann-Liouville integral," *Signal Process.*, vol. 98, pp. 150–157, 2014.

[17] S. Nakatani and H. Date, "Swing up control of inverted pendulum on a cart with collision by Monte Carlo model predictive control," in *Proc. IEEE 58th Annu. Conf. Soc. Instrum. Control Engineers Jpn.*, 2019, pp. 1050–1055.

[18] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *J. Guid., Control, Dyn.*, vol. 40, no. 2, pp. 344–357, 2017.

[19] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 7559–7566.

[20] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak, "Planning to explore via self-supervised world models," in *Proc. Int. Conf. Mach. Learn.*. PMLR, 2020, pp. 8583–8592.

[21] N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kegelman, and J. C. Gerdes, "Neural network vehicle models for high-performance automated driving," *Sci. Robot.*, vol. 4, no. 28, 2019, Art. no. eaaw1975.

[22] A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 3223–3230.

[23] J. Bae, T. Kim, W. Lee, and I. Shim, "Curriculum learning for vehicle lateral stability estimations," *IEEE Access*, vol. 9, pp. 89249–89262, 2021.