RL-Driven MPPI: Accelerating Online Control Laws Calculation With Offline Policy

Yue Qu, Hongqing Chu, Shuhua Gao, Jun Guan, Haoqi Yan, Liming Xiao, Shengbo Eben Li, *Senior Member, IEEE*, and Jingliang Duan, *Member, IEEE*

Abstract*—Model Predictive Path Integral (MPPI) is a recognized sampling-based approach for finite horizon optimal control problems. However, the efficacy and computational efficiency of prevailing MPPI methods are heavily reliant on the quality of rollouts. This is problematic because it is hard to sample a low-cost trajectory using random control sequences, thereby leading to inferior performance and computational efficiency, especially under constrained resources. To address this issue, we propose a data-efficient MPPI method called reinforcement learning-driven MPPI (RL-driven MPPI), which significantly reduces the dependency on the quantity and quality of samples. RL-driven MPPI employs an offline-online policy learning scheme, where the offline policy learned by RL serves as the initial solution and the initial rollout generator of MPPI, effectively combining the strengths of both RL and MPPI. The rollouts generated by RL typically correspond to a lower cost-to-go compared to random sampling, which significantly boosts the sample efficiency and convergence speed of MPPI. Moreover, the value function learned by RL offers an accurate estimation for infinite-horizon cost-to-go, enabling it to serve as a terminal term for the cost criteria of MPPI. This approach empowers MPPI to approximate an infinite-horizon cost with a shorter prediction horizon, thus enhancing real-time performance at each time step. An unmanned aerial vehicle control task is conducted to evaluate the proposed method. Results indicate that the proposed RL-driven MPPI method exhibits superior control performance and sample efficiency.

Manuscript received 16 November 2023; revised 4 December 2023; accepted 15 December 2023. Date of publication 29 December 2023; date of current version 29 April 2024. This work was supported in part by NSF China under Grants 52202487, 62273256, and 62203191, in part by the Fundamental Research Funds for the Central Universities under Grant FRF-OT-23-02, and in part by the Young Elite Scientists Sponsorship Program by CAST under Grant 2023QNRC001. *(Yue Qu and Hongqing Chu contributed equally to this work.) (Corresponding author: Jingliang Duan.)*

Yue Qu is with the School of Mechanical Engineering, Nanjing Vocational University of Industry Technology, Nanjing 210023, China, and also with the School of Mechanical Engineering, University of Science and Technology Beijing, Beijing 100083, China (e-mail: quyue868@gmail.com).

Hongqing Chu is with the College of Automotive Studies, Tongji University, Shanghai 201804, China (e-mail: chuhongqing@tongji.edu.cn).

Shuhua Gao is with the School of Control Science and Engineering, Shandong University, Jinan 250100, China (e-mail: shuhuagao@sdu.edu.cn).

Jun Guan is with the School of Automation, Jiangsu University of Science and Technology, Zhenjiang 212100, China (e-mail: jguan@just.edu.cn).

Haoqi Yan, Liming Xiao, and Jingliang Duan are with the School of Mechanical Engineering, University of Science and Technology Beijing, Beijing 100083, China (e-mail: yhq925@xs.ustb.edu.cn; xlming@xs.ustb.edu.cn; duanjl@ustb.edu.cn).

Shengbo Eben Li is with the State Key Lab of Automotive Safety and Energy, School of Vehicle and Mobility, Tsinghua University, Beijing 100084, China, and also with the Center for Intelligent Connected Vehicles and Transportation, Tsinghua University, Beijing 100084, China (e-mail: lishbo@tsinghua.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIV.2023.3348134.

Digital Object Identifier 10.1109/TIV.2023.3348134

***Index Terms*—Model predictive control (MPC), reinforcement learning (RL), unmanned aerial vehicle (UAV).**

I. Introduction

Optimal control methods are the powerhouse behind many challenging control and decision-making tasks, which can find a sequence of optimal control laws by solving a corresponding optimization problem defined by the predesignated cost and system dynamics [1]. In addition to the optimal control method for linear systems such as linear quadratic regulator (LQR), many powerful nonlinear control techniques have also been developed. However, most existing optimization methods for nonlinear systems, including approximate dynamic programming [2], iterative LQR [3], interior point methods [4], and gradient-descent-based methods [5], require the derivative or continuity of the running cost, limiting the flexibility when designing cost functions. Besides, even with a continuous cost, it may still be computationally untenable to use these methods to find the optimal solution for complex high-dimensional systems with limited computation power.

To learn optimal control laws for complex systems subject to flexible cost criteria, one popular approach is to precompute a controller offline using samples and perform it online. The most representative method that fits this description is the well-known reinforcement learning (RL) [1]. RL enables the agent to directly learn a parameterized policy using samples collected from interactions with complex systems [6], [7], [8], [9], [10], [11]. One sample corresponds to a tuple of the state, action, and the associated running cost, so RL requires no derivatives of either the dynamics or running cost. Most commonly used RL algorithms for continuous control settings are developed in the actor-critic (AC) architecture [12]. In AC architectures, the actor represents the parameterized policy to be learned, and the critic characterizes the value function, which describes the expected cumulative cost following the current policy. With the support of the high approximation ability of neural networks (NNs), many RL algorithms for continuous control settings have been developed over the past decade, including deep deterministic policy gradient (DDPG) [13], proximal policy optimization (PPO) [14], soft actor critic (SAC) [15], and distributional soft actor critic (DSAC) [16]. Although these algorithms have achieved great success in many challenging simulated tasks, their application in practice is still very limited. This limitation arises from the fact that, as an offline method, the learned policy may behave disastrously in a state that was not encountered during the offline training process.

Another well-known approach to eliminating the limitation of model complexity and cost continuity is the model predictive path integral control (MPPI), which enables the agent to learn an optimal control input at each time step by using the information of online rollout samples [17], [18]. Different from RL, the sampled trajectories of MPPI do not actually occur but are rolled out through the model. MPPI can be deemed as a sample-based version of the well-known model predictive control (MPC) method. It employs the path integral control framework to iteratively optimize a control sequence within a finite prediction horizon, and executes the first value of the control sequence as the online control input. The key idea behind MPPI is that the optimal finite-horizon cost-to-go is transformed into an expectation over all possible future trajectories using the Feynman-Kac lemma [19]. This expectation can be estimated by rollouts from the initial state, which is then used to generate the updated law of the control sequence. Its success paved the road toward several extensions and revisions of the basic framework. [20] generalizes the MPPI for systems with non-affine dynamics. [21] addresses a class of dynamics with compound Poisson noise. The tube-based MPPI augments the classical MPPI with an ancillary controller to reduce the effect of external disturbance [22], [23], or to cope with the state and control constraints of linear systems [24] and general nonlinear system [25]. In these works, the ancillary controller such as iterative linear quadratic gaussian control (iLQG) in [22], [23] or constrained covariance steering in [24] implicitly needs a smooth cost function and dynamic. Tsallis variational inference MPC [26] incorporates deformed logarithm and exponent functions into the operating likelihood to assign high weights to elite samples, thus achieving lower cost than the classical MPPI.

A major issue of existing MPPI methods is that their control performance and online computation efficiency depend on the quality of rollouts. This is problematic because it is hard to sample a low-cost trajectory using random control sequences, especially for high-dimensional dynamics with a long prediction horizon. To improve the policy performance and calculation speed of MPPI, this paper proposes a novel sample-based optimal control method, called RL-driven MPPI, for policy learning of complex systems with flexible cost criteria. Specifically, our contributions are summarized as follows:

1) The proposed RL-driven MPPI algorithm adopts a novel offline-online policy learning scheme, where the offline policy learned by RL serves as the initial solution and the initial rollout generator of MPPI, thus obtaining the merits of both RL and MPPI. For one thing, the final online control performance will be bounded below by RL since MPPI improves the control performance on the basis of rollouts generated by offline RL policy. For another thing, the rollouts provided by RL generally correspond to a lower cost-to-go than random sampling, which can greatly increase the sample efficiency and convergence speed of MPPI at each online time step.
2) We utilize distributional soft actor critic (DSAC), one of the state-of-the-art RL algorithms for continuous control, to learn an NN-based policy that maps the system state to the mean and variance of a Gaussian action distribution. By sampling the control input from the learned action distribution, various trajectories from the initial state will be generated as the initialization of MPPI. Besides, the learned value function of DSAC provides an accurate estimate for infinite-horizon cost-to-go, which could serve as a terminal term of the cost criteria of MPPI. This enables the MPPI to approximate an infinite-horizon cost using a short prediction horizon, thereby further improving the real-time performance of MPPI at each time step.
3) Different from existing MPPI methods [18], [27] that suppose the variance of action distribution being a fixed value for all states, the proposed method updates the mean and covariance of the control input simultaneously. We derive a sample-based covariance learning mechanism that strikes an optimal balance between exploration and exploitation, leading to better sample efficiency and control performance.

We put our proposed method to the test through an unmanned aerial vehicle (UAV) control task. The results clearly demonstrate that our method not only expedites the convergence process but also significantly reduces the cost-to-go.

The remainder of this paper is organized as follows: In Section II we present the problem statement for MPPI and reinforcement learning. In Section III an overview of DSAC and MPPI is first presented, and then the proposed framework RL-driven MPPI is provided. Section IV encompasses a suite of simulations. In Section V, we discuss our future work and conclude the paper.

II. Preliminaries

A. Problem Statement

Consider the general time-invariant discrete-time dynamic system

$$x_{i+1} = f(x_i, v_i) \tag{1}$$

with state $x_i \in \mathcal{X} \subset \mathbb{R}^n$, control input $v_i \in \mathcal{U} \subset \mathbb{R}^m$, and the system dynamics function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. f can be a generally nonlinear and input-nonaffine system. We assume that the control inputs v_t obey a diagonal Gaussian distribution:

$$v_t \sim \mathcal{N}(u_t, \sigma_t^2), \tag{2}$$

where $\sigma_t^2 \in \mathbb{R}^{m\times m}$ denotes the covariance matrix. Without loss of generality, the covariance matrix can be infinitesimal to cover the case where the optimal policy is deterministic.

The infinite-horizon optimal control seeks a sequence of control inputs minimizing the infinite-horizon accumulated cost:

$$\min_{\{u_0,\sigma_0\},\,\{u_1,\sigma_1\},\,\cdots} \sum_{t=0}^{\infty} l(x_t)$$
$$\text{s.t.} \quad \tilde{}(1). \tag{3}$$

In this paper, we assume that the running cost $l(x_t)$ solely relies upon the state, which can be a non-differentiable or discontinuous function. For control problems with flexible running cost

$l(x_t)$ and complex systems, the infinite-horizon cost can be an infinite value for all possible policies, leading to an infeasible optimization problem. For the purposes of generalization, two commonly used formulations are considered in this study: (1) discounted infinite-horizon (InfH) optimal control and (2) model predictive control (MPC) based on finite-horizon (FH) optimization.

B. Discounted Infinite-Horizon Optimal Control and Reinforcement Learning (RL)

For each state x_t, the discounted infinite-horizon optimal control problem is formulated as

$$\begin{aligned}&\min_{\pi}\ V_{\text{InfH}}^{\pi}(x_t) := \mathbb{E}\left[\sum_{i=t}^{\infty}\gamma^{i-t}l(x_i)\right]\\ &\text{s.t.}\quad \{u_i,\sigma_i\}=\pi(x_i),\ (1),\ \tilde{}(2),\end{aligned}\tag{4}$$

where $\pi(x_i)$ is the policy function which maps the state to the mean and diagonal standard deviation of the stochastic policy, $V_{\text{InfH}}^{\pi}(x_t)$ is the state value function originating from x_t and following policy π, and $\gamma \in (0,1)$ is the discount factor that balances the importance of current and future reward, which can also ensure that $V_{\text{InfH}}^{\pi}(x_t)$ is a finite value for bounded running cost $l(x_i)$.

Reinforcement learning (RL) serves as a popular sample-based method to finding the nearly optimal policy of (4), which does not require the differentiability of $l(x_i)$ and supports direct optimization over the set of stochastic policies. To solve (4), an actor-critic architecture should be utilized, where the actor corresponds to a parameterized policy function and the critic corresponds to a parameterized value function. With the help of the well-known policy iteration iterative framework, the parameterized policy and value functions will gradually approach the nearly optimal solutions offline. Since the final policy is learned based on the samples collected from interactions with environments, it may behave disastrously in a state that was not reached during the offline training process. In addition, the optimality of the learned policy cannot be guaranteed due to the existence of approximation errors and overestimation problems. These disadvantages limit the practical applications of RL.

C. Finite-Horizon Optimal Control and Model Predictive Path Integral Control (MPPI)

At each time step t, MPC finds the optimal control sequence $\{u_t^\star,\sigma_t^\star\},\{u_{t+1}^\star,\sigma_{t+1}^\star\},\dots,\{u_{t+N-1}^\star,\sigma_{t+N-1}^\star\}$ by solving the following FH optimization problem

$$\begin{aligned}&\min_{\mathrm{U},\Sigma}\ V_{\text{FH}}^{\mathrm{U},\Sigma}(x_t) := \mathbb{E}\left[\sum_{i=t}^{t+N-1} l(x_i)+\mathsf{T}(x_{t+N})\right]\\ &\text{s.t.}\quad (1),\end{aligned}\tag{5}$$

where N is the prediction horizon, $\mathrm{U} := \{u_t,u_{t+1},\dots,u_{t+N-1}\}$ is the N-step mean action sequence, $\Sigma := \{\sigma_t^2,\sigma_{t+1}^2,\dots,\sigma_{t+N-1}^2\}$ is the N-step variance sequence, $V_{\text{FH}}^{\mathrm{U},\Sigma}(x_t)$ is the expected FH cost-to-go under $\{\mathrm{U},\Sigma\}$, and $\mathsf{T}(x_{t+N})$ is the terminal cost. Only the first control input $v_t^\star \sim \mathcal{N}(u_t^\star,{\sigma_t^\star}^2)$ will be executed to obtain the next state x_{t+1}. This process is repeated at the next time step, resulting in a receding horizon control scheme.

As a sample-based MPC solver, model predictive path integral control (MPPI) enables the agent to learn the optimal control sequence at each time step by using online rollouts $\{x_t,v_t,x_{t+1},v_{t+1},\dots,x_{t+N-1},v_{t+N-1},x_{t+N}\}$. According to the Feynman-Kac lemma [19], the optimal finite-horizon cost-to-go $V_{\text{FH}}^\star(x_t)$ can be represented as an expectation over all possible future trajectories from x_t. By utilizing the sample-average trick, this expectation can be estimated using rollouts generated by several random control sequences. The estimated optimal cost-to-go can be further used to generate the updated law of the control sequence.

Different from RL, MPPI is an online method that calculates the optimal control input at each time step. Therefore, the control performance starting from arbitrary states $x_t \in \mathcal{X}$ can be guaranteed. However, the online calculation process of MPPI can be time-consuming, especially for high-dimensional dynamics with a long prediction horizon, since a large number of rollouts are required to obtain a satisfactory estimate of the optimal cost-to-go.

III. RL-DRIVEN MPPI

As discussed in Section II, RL learns a parameterized policy by solving (4) offline using samples collected from the interaction with system (1). The learned RL policy has excellent real-time performance in the online decision-making process but may fail in some states due to the suboptimality and rare visitations in the training process. As a comparison, MPPI finds the optimal control sequence at each time step by solving (5) utilizing rollouts from the current state. It outperforms RL policy in ensuring the policy feasibility at all feasible states $x_t \in \mathcal{X}$, but leads to poor online decision-making efficiency. In this section, we propose an RL-driven MPPI method to obtain the merits of both RL and MPPI, leading to better control and real-time performance for complex systems with flexible cost criteria.

A. Framework

The key insight behind the proposed RL-driven MPPI method is illustrated in Fig. 1, which consists of two parts: (1) the offline RL training module and (2) the online MPPI control module. In the offline RL training module, we learn a diagonal Gaussian policy $\pi(v|x): \mathcal{X}\to\mathcal{P}(v)$ using RL methods, which maps a given state to a probability distribution over control inputs. To be more specific, we model this policy by a neural network (NN), which takes the state as input and outputs the mean and diagonal standard derivation of the action distribution.

The learned stochastic policy will not be directly applied to online applications but will serve as a sample generator of MPPI. In particular, in the online MPPI control module, we randomly generate control sequences according to $v_i \sim \pi(\cdot|x_i)$ for $i \in \{t,t+1,\dots,t+N-1\}$ to obtain multiple rollouts from the current state x_t. The rollouts provided by RL generally correspond to lower cost-to-go than random sampling, which

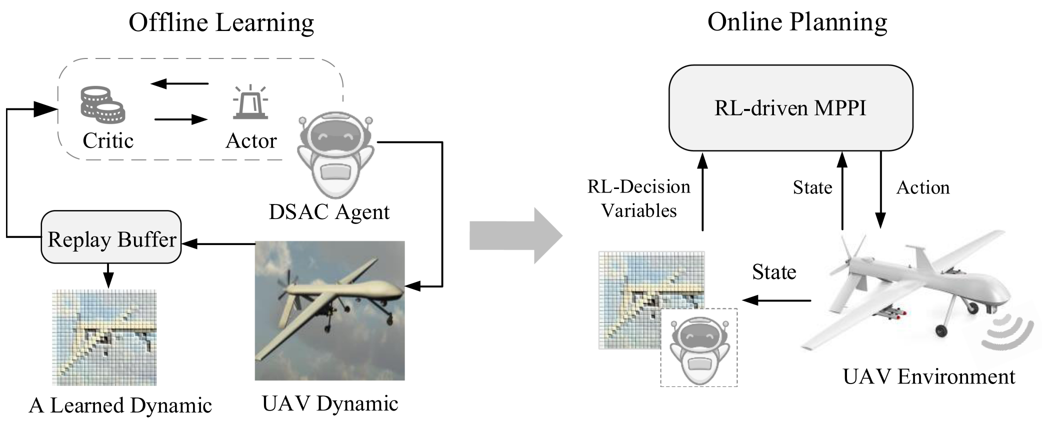


Fig. 1. Schematic diagram of RL-driven MPPI. The pre-trained NNs include DSAC-actor, DSAC-critic, and NN-model, which will be used to improve the performance and efficiency of a sampling-based online planning method.

can greatly increase the convergence speed and performance of MPPI at each online time step, especially for complex systems. Therefore, RL-driven MPPI is expected to perform better than the traditional MPPI in both control and real-time performance. Since the final control sequence of MPPI is obtained based on the offline RL policy, the online control performance will be bounded below by RL. RL-driven MPPI is also superior to RL in the final performance. In summary, we can say that the proposed RL-driven MPPI method attains the merits of both RL and MPPI. Next, we will introduce the offline training and online control modules in detail.

B. RL-Driven MPPI : Offline RL Training Module

The aim of the offline RL training module is to learn an offline stochastic policy for random rollouts generating process. The representative RL algorithms that meet this basic requirement include distributional soft actor critic (DSAC) [16], soft actor critic (SAC) [15], proximal policy optimization (PPO) [14], and trust region policy optimization (TRPO) [28]. In this paper, we choose the DSAC algorithm as an example, which is an algorithm developed by our team [12], [16], [29] [1].

DSAC attempts to learn the distribution of the infinite-horizon cost to enhance the estimation accuracy of expected cost value, thereby improving policy performance. We define the state-action cost as

$$Z^\pi(x_t, v_t) = l(x_t) + \sum_{i=t+1}^{\infty} \gamma^{i-t}[l(x_i) - \alpha\mathcal{H}(\pi(\cdot|x_i))], \quad (6)$$

which is a random variable due to the randomness in the policy π, where $\mathcal{H}(\pi(\cdot|x)) = \mathop{\mathbb{E}}_{v\sim\pi(\cdot|x)}[-\log\pi(v \mid x)]$ is the policy entropy and α is the temperature weighing the relative importance of the entropy term against the running cost. The distributional cost function $\mathcal{Z}^\pi(Z^\pi(x,v)|x,v) : \mathcal{S}\times\mathcal{A} \to \mathcal{P}(Z^\pi(x,v))$ maps (x,v) to a distribution over state-action cost.

[1]Access open-source code of DSAC at https://github.com/Jingliang-Duan/DSAC-T

Based on this definition, DSAC aims to learn a stochastic policy to minimize the expected infinite-horizon cost, i.e.,

$$\pi^*(\cdot|x) = \arg\min \mathop{\mathbb{E}}_{v\sim\pi(\cdot|x)}[Q^\pi(x_t, v_t) + \alpha\log\pi(v \mid x)], \quad (7)$$

where

$$Q^\pi(x_t, v_t) := \mathop{\mathbb{E}}_{Z\sim\mathcal{Z}^\pi(\cdot|x,v)}[Z^\pi(x_t, v_t)] \quad (8)$$

is the expected infinite-horizon state-action cost.

We employ neural networks (NNs) $\mathcal{Z}_\theta(\cdot|x,v)$ and $\pi_\phi(\cdot|x)$ to model the distributional cost function and stochastic policy as diagonal Gaussians respectively, where θ and ϕ are NN parameters to be learned. Each NN has two outputs, one corresponding to the mean value of the distribution and the other to the standard deviation. For exposition simplicity, we denote the mean output of $\mathcal{Z}_\theta(\cdot|x,v)$ as $Q_\theta(x,v)$, and denote the mean and standard deviation of $\pi_\phi(\cdot|x)$ as $u_\phi(x)$ and $\sigma_\phi(x)$, respectively.

Like most RL algorithms, DSAC alternates between policy evaluation and policy improvement to learn an offline policy. In the policy evaluation step, the distributional cost NN $\mathcal{Z}_\theta(\cdot|x,v)$ is updated by minimizing the Kullback-Leibler divergence between the target cost distribution and current distribution, which is expressed as

$$J_\mathcal{Z}(\theta) = -\mathop{\mathbb{E}}_{\substack{(x_t,v_t,l_t,x_{t+1})\sim\mathcal{B},\\ v_{t+1}\sim\pi_\phi(\cdot|x_{t+1}),\\ Z_{t+1}\sim\mathcal{Z}_\theta(\cdot|x_t,v_t)}}\left[\log\mathcal{P}(\hat{Z}(x_t,v_t)|\mathcal{Z}_\theta(\cdot|x_t,v_t))\right]. \quad (9)$$

where $\mathcal{B}$ is the experience replay buffer, $\hat{Z}(x_t, v_t)$ is the target state-action cost which is calculated by

$$\hat{Z}(x_t, v_t) = l(x_t) + \gamma(Z(x_{t+1}, v_{t+1}) + \alpha\log\pi(v_{t+1}|x_{t+1})).$$

According to (7), in the policy improvement process, the policy NN is updated by minimizing

$$J_\pi(\phi) = \mathop{\mathbb{E}}_{x_t\sim\mathcal{B}, v_t\sim\pi_\phi(\cdot|x_t)}[Q_\theta(x_t,v_t) + \alpha\log(\pi_\phi(v_t|x_t))]. \quad (10)$$

C. RL-Driven MPPI: The Online MPPI Control Module

Next, we will discuss how to find a better online control sequence based on the learned RL policy using MPPI. Suppose that we are given a sequence of control inputs $\boldsymbol{\nu} := \{v_t, v_{t+1}, \dots, v_{t+N-1}\}$. For any initial state x_t, a unique state trajectory $\boldsymbol{X}^{\boldsymbol{\nu}} := \{x_t, x_{t+1}, \dots, x_{t+N}\}$ can be generated by implementing $\boldsymbol{\nu}$ on the system (1). Then the cost-to-go of the input sequence $\boldsymbol{\nu}$ can be defined as:

$$C(\boldsymbol{\nu}; x_t) := \sum_{i=t}^{t+N-1} \ell(x_i) + \mathsf{T}(x_{t+N}). \tag{11}$$

From (5), the relationship between $C(\boldsymbol{\nu}; x_t)$ and the expected FH cost-to-go $V_{\text{FH}}^{\text{U},\Sigma}(x_t)$ can be derived as

$$V_{\text{FH}}^{\text{U},\Sigma}(x_t) = \mathbb{E}_{\mathbb{P}_{\text{U},\Sigma}}[C(\boldsymbol{\nu}; x_t)]. \tag{12}$$

We can then define the probability density function (PDF) for $\boldsymbol{\nu}$ as

$$p(\boldsymbol{\nu} \mid \text{U}, \Sigma) := Z^{-1} \exp\left(-\frac{1}{2} \sum_{i=t}^{t+N-1} (v_i - u_i)^\top \sigma_i^{-1} (v_i - u_i)\right) \tag{13}$$

where $Z = (2\pi)^{\frac{mN}{2}} \prod_{i=t}^{t+N-1} |\sigma_i|^{\frac{1}{2}}$. Recall that m represents the dimension of the control input. We denote the corresponding distribution of $p(\boldsymbol{\nu} \mid \text{U}, \Sigma)$ as $\mathbb{P}_{\text{U},\Sigma}$.

If we could sample control inputs from the optimal distribution $\mathbb{P}_{\text{U}^\star,\Sigma^\star}$, the optimal policy sequence $\text{U}^\star$ and $\Sigma^\star$ can be easily obtained by counting the mean and standard deviation of the sampled inputs:

$$u_i^\star = \mathbb{E}_{\mathbb{P}_{\text{U}^\star,\Sigma^\star}}(v_i), \tag{14a}$$

$$\text{diag}[\sigma_i^\star] = \sqrt{\mathbb{E}_{\mathbb{P}_{\text{U}^\star,\Sigma^\star}}[(v_i - u_i) \circ (v_i - u_i)]}, \quad i = t, \dots, t+N-1, \tag{14b}$$

where $\circ$ represents the element-wise product operator. However, directly sampling from $\mathbb{P}_{\text{U}^\star,\Sigma^\star}$ is intractable, so we choose to provide an unbiased estimate of the optimal control solution using the importance sampling technique.

We denote the likelihood ratio $\omega(\boldsymbol{\nu})$ as

$$\omega(\boldsymbol{\nu}; \text{U}, \Sigma) := \frac{p(\boldsymbol{\nu} \mid \text{U}^\star, \Sigma^\star)}{p(\boldsymbol{\nu} \mid \text{U}, \Sigma)}. \tag{15}$$

It has been derived in [18], [27] that

$$\omega(\boldsymbol{\nu}; \text{U}, \Sigma) = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda}\text{C}(\boldsymbol{\nu}; \text{x}_\text{t}) + \text{D}(\boldsymbol{\nu}; \text{U}, \Sigma)\right), \tag{16}$$

where $\lambda \in \mathbb{R}^+$ is called the inverse temperature, $\eta = \int \omega(\boldsymbol{\nu}; \text{U}, \Sigma) \text{d}\boldsymbol{\nu}$, and

$$D(\boldsymbol{\nu}; \text{U}, \Sigma) := \sum_{i=t}^{t+N-1} \left(-v_i^\top \sigma_i^{-1} u_i + \frac{u_i^\top \sigma_i^{-1} u_i}{2}\right).$$

This allows us to estimate $\text{U}^\star$ and $\Sigma^\star$ using the sampling trajectories from arbitrary control distribution:

$$u_i^\star = \mathbb{E}_{\mathbb{P}_{\text{U},\Sigma}}[\omega(\boldsymbol{\nu}; \text{U}, \Sigma) v_i], \tag{17a}$$

$$\text{diag}(\sigma_i^\star) = \sqrt{\mathbb{E}_{\mathbb{P}_{\text{U},\Sigma}}[\omega(\boldsymbol{\nu}; \text{U}, \Sigma)(v_i - u_i) \circ (v_i - u_i)]}, \quad i = t, \dots, t+N-1. \tag{17b}$$

Under the condition of sufficient sampling, the optimal control sequence obtained according to $\mathbb{P}_{\text{U},\Sigma}$ is equivalent to that obtained based on the optimal distribution $\mathbb{P}_{\text{U}^\star,\Sigma^\star}$.

In practical applications, the expectation operator is usually relaxed to numerical statistics based on a finite number of sampled trajectories. Then, an iterative update law can be derived to shift the control distribution close to the optimal gradually:

$$u_i^{k+1} = \frac{\sum_{z=1}^{Z} \left[b(\boldsymbol{\nu}^{(z)}; \text{U}^k, \Sigma^k) v_i^k\right]}{\sum_{z=1}^{Z} \left[b(\boldsymbol{\nu}^{(z)}; \text{U}^k, \Sigma^k)\right]},$$

$$\text{diag}(\sigma_i^{k+1}) = \sqrt{\frac{\sum_{z=1}^{Z} [b(\boldsymbol{\nu}^{(z)}; \text{U}^k, \Sigma^k)(v_i^k - u_i^k) \circ (v_i^k - u_i^k)]}{\sum_{z=1}^{Z} \left[b(\boldsymbol{\nu}^{(z)}; \text{U}^k, \Sigma^k)\right]}}, \quad i = t, \dots, t+N-1, \tag{18}$$

where $\boldsymbol{\nu}^{(z)} \sim \mathcal{N}(\text{U}^k, \Sigma^k)$, Z is the number of sampled trajectories, and

$$b(\boldsymbol{\nu}^{(z)}; \text{U}^k, \Sigma^k) := \exp\left(-\frac{1}{\lambda}\text{C}(\boldsymbol{\nu}^{(z)}; \text{x}_\text{t}) + \text{D}(\boldsymbol{\nu}^{(z)}; \text{U}^\text{k}, \Sigma^\text{k})\right).$$

D. Main Features of RL-Driven MPPI

In previous sections, we introduced the main principles of offline RL training and online MPPI control modules. Now, we are ready to present how we build the RL-driven method by combining these two modules. Compared with the existing MPPI studies [18], [27], the main characteristic of the proposed method is embodied in four aspects.

1) Rl-Initialization: A proper initialization is vital for sampling-based algorithms, especially for systems with large configuration searching spaces. Random initialization is a typical approach. In this way, the initial value may be far away from the optimum, which in turn requires an increased number of iterations.

To mitigate this challenge, RL-driven MPPI employs an RL-initialization strategy. In particular, the learned RL-policy $\pi_\phi(\cdot|x_t)$ maps the state sequence $\{x_{t:t+N-1}\}$ to the N-step mean sequence $\text{U}^0 = \{u_{t:t+N-1}\}$ and standard deviation sequence $\Sigma^0 = \{\sigma^2_{t:t+N-1}\}$ as initial parameters for MPPI, where the superscript 0 refers to the iterative initial. The state sequence $\{x_{t:t+N-1}\}$ can be obtained by repeatedly applying the mean value $u_\phi(x)$ of policy $\pi_\phi(\cdot|x)$ starting from x_t.

In contrast to a randomly initialized policy, the offline policy $\pi_{\phi^\star}(\cdot|x)$ learned by DSAC is undoubtedly closer to the optimal solution of (5). Consequently, one can reasonably anticipate that RL-driven MPPI will offer enhanced computational efficiency than the traditional MPPI. Additionally, as the trained policy network exhibits excellent real-time performance, there is no additional computational burden incurred.

2) Hybrid Sampling Strategy (HSS): As shown in (17), the optimal policy $(\text{U}^\star, \Sigma^\star)$ is achieved by relying on a forward sampling process where the low-cost solution will be assigned high probability mass. This means that the learning speed and accuracy of MPPI heavily rely on the quality of the sampled trajectories. Typically, in traditional MPPI methods, candidate solutions are drawn exclusively from the current control distribution $\mathbb{P}_{\mathrm{U}^k,\Sigma^k}$. However, when obtaining low-cost solutions from the current distribution $\mathbb{P}_{\mathrm{U}^k,\Sigma^k}$ proves challenging, MPPI can become prohibitively time-inefficient, necessitating a large number of samples or iterations to approach a nearly-optimal policy.

Inspired by recent progress in hybrid search methods [30], [31], in which the search space contains locally optimal solutions to improve the global convergence of sampling-based planning algorithms, we propose a hybrid sampling strategy (HSS) to improve the performance of MPPI, where candidate solutions are not only sampled from the current distribution $\mathbb{P}_{\mathrm{U}^k,\Sigma^k}$ but also the stochastic policy $\pi_{\phi^\star}(\cdot|x)$. In this case, we can informally treat $\pi_{\phi^\star}(\cdot|x)$ as a lower bound of the MPPI solution. We call the samples generated by $\pi_{\phi^\star}(\cdot|x)$ as guided samples.

Note that guided samples can be utilized repeatedly during the online MPPI iteration process. In other words, we only need to collect guided samples with the offline RL policy at the beginning of each online MPPI iteration process. At each iteration k, we will sort the guided samples and the samples obtained by (U^k, Σ^k) according to the cost-to-go. Finally, in line with strategies recommended in existing literature [32], [33], to maximize the utility of low-cost samples and minimize the impact of high-cost samples, we employ the top-Z samples to generate the next iterate.

3) Simultaneous Updating of Mean and Variance: Most existing MPPI methods [18], [27] only update the mean control inputs u_t while supposing that the variance matrix σ_t is a fixed value for all states. To pursue better performance, the variance should be adjusted adaptively according to the states to make a good balance between exploration and exploitation. Actually, we have already derived and given the update law of variance in Section III-C, see (14), (17), and (18). This enables us to select a relatively large initial variance to promote exploration. To prevent the phenomenon of premature convergence, we set a lower limit for the standard deviation:

$$\sigma^k = \max\left(\sigma^k, \sigma_{\min} I\right),$$

where $\sigma_{\min}$ is a positive scalar.

4) Terminal Cost Design: We adopt the Q-value NN ($Q_{\theta^\star}(x, v)$) learned by DSAC as the terminal cost term in (11):

$$\mathsf{T}\left(x_{t+N}\right) = Q_{\theta^\star}(x_{t+N}, u_{t+N}), \tag{19}$$

where $u_{t+N} = \mathbb{E}_{v_{t+N}\sim\pi_{\phi^\star}(\cdot|x_{t+N})}[v_{t+N}]$. According to the definition given in (8), $Q_{\theta^\star}(x_{t+N}, u_{t+N})$ describes the discounted infinite-horizon cost starting from x_{t+N}. Hence, the inclusion of the Q-value enables $C(\boldsymbol{\nu}; x_t)$ in (11) well approximates the infinite-horizon cost-to-go, leading to a better MPPI controller. Moreover, since $Q_{\theta^\star}(x_t, u_t)$ estimates the cost-to-go from any x_t, we can choose a relatively small predictive horizon N to relieve the time burden of long-horizon rollout without losing too much control performance.

Algorithm 1: RL-driven MPPI (RLMPPI).

```
    Return: Offline RL training:
 1:  Initialize parameters $\theta$ and $\phi$ of value and policy NNs
 2:  repeat
 3:      Update $\theta$ by minimizing $J_{\mathcal{Z}}(\theta)$ in (9)
 4:      Update $\phi$ by minimizing $J_\pi(\phi)$ in (10)
 5:  until Convergence
 6:  return $\mathcal{Z}_{\theta^\star}(\cdot|x, v)$ and $\pi_{\phi^\star}(\cdot|x)$
    Online MPPI control:
 7:  Given initial state $x_0^{\mathrm{real}}$ and set $t = 0$
 8:  Given the set of terminal states of the control process $\mathbb{X}_{\mathrm{end}}$
 9:  while $x_t^{\mathrm{real}} \notin \mathbb{X}_{\mathrm{end}}$ do
10:      Let $x_t = x_t^{\mathrm{real}}$
11:      Get $x_{t+1:t+N-1}$ by applying $u_{\phi^\star}(x)$ starting from $x_t$
12:      Initial $(\mathrm{U}^0, \Sigma^0)$ with
             $\begin{cases}\mathrm{U}^0 = u_{\phi^\star}(\{x_{t:t+N-1}\})\\ \Sigma^0 = \sigma_{\phi^\star}(\{x_{t:t+N-1}\})^2\end{cases}$
13:      Sample $N_{\mathrm{RL}}$ guided rollouts using $\pi_{\phi^\star}(\cdot|x_t)$
14:      Put $N_{\mathrm{RL}}$ guided rollouts in set $\mathcal{D}_\pi$
15:      for $k = 0 \ldots K-1$ do
16:          Sample $N_{\mathrm{MPPI}}$ rollouts using $(\mathrm{U}^k, \Sigma^k)$
17:          Put these rollouts in set $\mathcal{D}_k$
18:          Calculate cost in (11) with (19) over $\mathcal{D}_\pi \bigcup \mathcal{D}_k$
19:          Select top-$Z$ control sequences according to cost
20:          Update $(\mathrm{U}^{k+1}, \Sigma^{k+1})$ using (18)
21:      end for
22:      Get $(\mathrm{U}^K, \Sigma^K) = (\{u^K_{t:t+N-1}\}, \{(\sigma^K_{t:t+N-1})^2\})$
23:      Apply $u_t^K$ as control input and obtain $x_{t+1}^{\mathrm{real}}$
24:      $t = t + 1$
25:  end while
```

The pseudocode for the proposed RL-driven MPPI algorithm (RLMPPI), which incorporates these three core features, can be found in Algorithm 1. To prevent any misunderstandings, we use x_t^{real} to represent the state obtained during the actual control process.

Remark 1: In the RL-driven MPPI scheme, the RL module is designed to learn an offline stochastic policy for continuous control, as well as the associated value function. These learned elements are instrumental in accelerating the convergence rate and boosting the control efficacy of MPPI. Importantly, RL-driven MPPI is versatile and can be seamlessly integrated with a range of RL algorithms, such as SAC [15] and PPO [14], provided they are capable of learning a continuous stochastic policy.

Remark 2: As shown in (12), the objective of RL-driven MPPI aligns with the traditional MPC problem defined in (5). Consequently, RL-driven MPPI can be viewed as a sampling-based approach to solving the traditional MPC problem. This implies that if RL-driven MPPI converges to the optimal solution, it should maintain the stability properties inherent in the original MPC problem. However, the original MPC problem might lead to an unstable controller due to factors like inadequate terminal

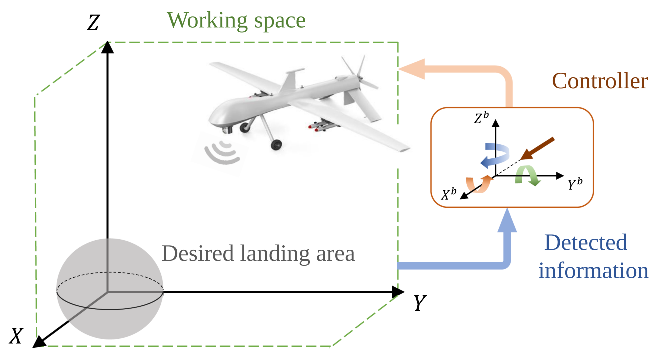


Fig. 2. Illustration of the waypoint approach task.

TABLE I
DETAILS OF SYSTEM STATE AND CONTROL INPUT

	Explanation	Symbol	Unit
state x_t	position	$p = (p_x, p_y, p_z) \in \mathbb{R}^3$	[m]
	velocity	$\nu = (\nu_x, \nu_y, \nu_z) \in \mathbb{R}^3$	[m/s]
	angular velocity	$w = (w_x, w_y, w_z) \in \mathbb{R}^3$	[deg/s]
	attitude	$q = (q_1, q_2, q_3, q_4) \in \mathbb{R}^4$	
input v_t	thrust	T_1	[N]
	torque	T_2, T_3, T_4	[Nm]

cost design or a limited prediction horizon. RL-driven MPPI can mitigate these issues in certain cases, particularly because incorporating the Q-value allows $C(\boldsymbol{\nu}; x_t)$ in (11) to closely approximate the infinite-horizon cost-to-go. An in-depth study of the stability of RL-driven MPPI will be left for future work.

IV. Numerical Experiments

To verify the proposed RL-driven MPPI method, we choose the waypoint approach task of a 6-degree-of-freedom (DOF) unmanned aerial vehicle (UAV) as an example. Results indicate that our algorithm outperforms MPPI in both final performance and iteration efficiency.

A. Problem Description

The control objective is to drive the 6-DOF UAV to land in a desired region with the shortest path while realizing a minimum flight time, as depicted in Fig. 2. The workspace of this experiment is a cube with a side length of 10 m. The initial position of the UAV is randomly given within the workspace with a pre-designed landing area. The dynamic of the UAV is assumed to be unknown, but the samples (x_t, v_t, x_{t+1}) can be obtained by applying a given policy to a simulator. The simulator is developed based on the continuous-time model $f_c(x_t, v_t)$ introduced in [34], which is discretized using the forward Euler method:

$$x_{t+1} = x_t + f_c(x_t, v_t)\Delta t. \tag{20}$$

where Δt is the time interval, and the details of system states and control inputs are listed in Table I. For comprehensive understanding, a detailed formulation of $f_c(x, v)$ is included in Appendix A. The target region for this control task

TABLE II
PARAMETERS OF ENVIRONMENT

Parameter	Value	Parameter	Value
Δt	0.05 [s]	ϖ_ν	0.01
ϱ	1.5 [m]	ϖ_w	0.001
p_{target}	$[0, 0, 0]^\top$ [m]	ϖ_q	0.001
ϖ_p	0.02		

is conceptualized as a sphere situated at waypoint $\boldsymbol{p}_{\text{target}} = [p_{\text{target}_x}, p_{\text{target}_y}, p_{\text{target}_z}]^\top$:

$$\mathbb{X}_{\text{end}} = \left\{p \in \mathbb{R}^3 \middle| \|p - p_{\text{target}}\|_2 \le \varrho\right\}, \tag{21}$$

where ϱ is the sphere radius.

We express the running cost $l(x_t)$ as the minus of the reward function $r(x_t)$, i.e.,

$$l(x_t) = -r(x_t).$$

It is obvious that $\arg\min \sum_{t=0}^{\infty} l(x_t)$ is equivalent to $\arg\max \sum_{t=0}^{\infty} r(x_t)$. The reward function is designed as

$$r = \begin{cases} 200, & \text{if } p \in \mathbb{X}_{\text{end}} \\ -\varpi_p \|p - p_{\text{target}}\|^2 - \varpi_\nu \|\nu\|^2 - \varpi_w \|w\|^2 - \varpi_q e_q, & \text{else} \end{cases} \tag{22}$$

where $\varpi_p, \varpi_v, \varpi_w, \varpi_q$ are constant positive weights, e_q represents the attitude error, which can be defined according to [35]:

$$e_q := \frac{1}{2} \operatorname{Tr}\left(I - R^\top(q_{\text{target}})\, R(q)\right), \tag{23}$$

where q_{target} is the desired attitude and $R \in \mathbb{R}^{3\times3}$ is the direction cosine matrix which is given in [35]. In the reward function, the first term on the second line of (22) imposes a penalty for deviation of the UAV's current position from its intended target p_{target}. The second and third terms penalize high velocity and angular velocity to encourage smoother control policies. The final term addresses attitude error, promoting the UAV's maintenance of the desired attitude. The environment parameters are listed in Table II.

B. Algorithm Details

We compare the proposed RL-driven MPPI method against the pure DSAC, MPPI, and another sampling-based MPC method called cross-entropy-motion-based MPC (CEM-MPC) [32]. Since MPPI iteratively computes the control inputs online, it requires a knowledge of the UAV model. Therefore, we learn a parameterized transition model $\hat{f}_\psi(x_t, u_t)$ during the offline RL training process by

$$\psi_{k+1} \leftarrow \arg\min_{\psi} \mathop{\mathbb{E}}_{(x_t, u_t, x_{t+1})\sim\mathcal{B}} (\hat{f}_{\psi_k}(x_t, u_t) - x_{t+1})^2, \tag{24}$$

where $\mathcal{B}$ is the replay buffer of RL. The learned model is utilized for both the online MPPI module of RL-driven MPPI and the traditional MPPI method. The return distribution, the stochastic policy, and the approximate dynamic model are represented

TABLE III
DETILED HYPER-PARAMETER

	Hyperparameters	Value
RL module	Replay Buffer Size	1×10^6
	Batch Size	256
	Discount Factor (γ)	0.99
	Critic Learning Rate	1×10^{-5}
	Actor Learning Rate	2×10^{-5}
	Learning Rate of α	1×10^{-5}
	Model Learning Rate	1×10^{-4}
	Expected Entropy $\mathcal{H}$	$\mathcal{H} = -4$
	Learning Rate for Target	0.005
	Maximum Episode Length	1000
	Training Iteration Number	1×10^6
MPPI module	Iteration Number K	6
	Prediction Horizon N	15
	Inverse Temperature λ	1
	Number of Rollouts $N_{\mathrm{RL}} = N_{\mathrm{MPPI}} = Z$	125
	Minimum Standard Deviation $\sigma_{\min}$	0.5
CEM module	Iteration Number	10
	Prediction Horizon	15
	Elite Fraction	0.05
	CE Sampling Ratio	0.5

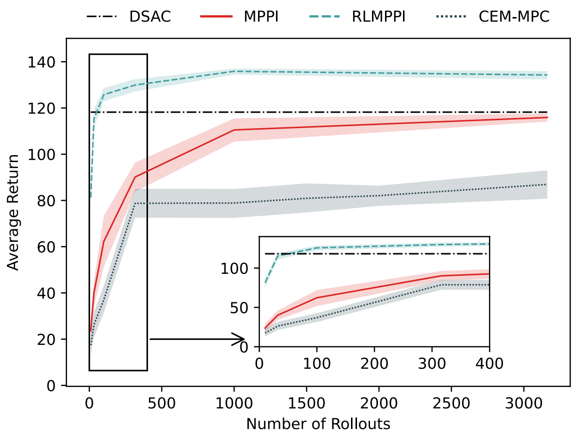
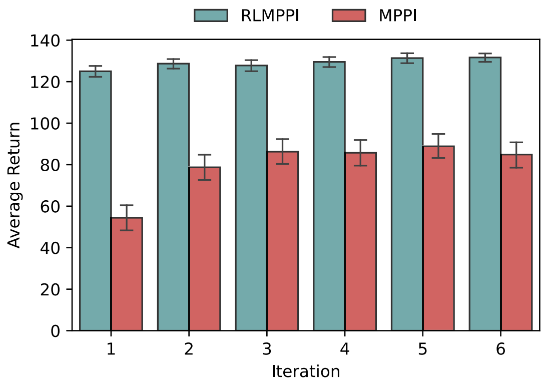


Fig. 3. Average return of different methods. The solid lines correspond to the mean and the shaded regions correspond to a 95% confidence interval over 500 runs. The grey dashed line indicates the expected return of the policy obtained by DSAC. We assume that $N_{\mathrm{RL}} = N_{\mathrm{MPPI}} = Z$ for RL-driven MPPI.

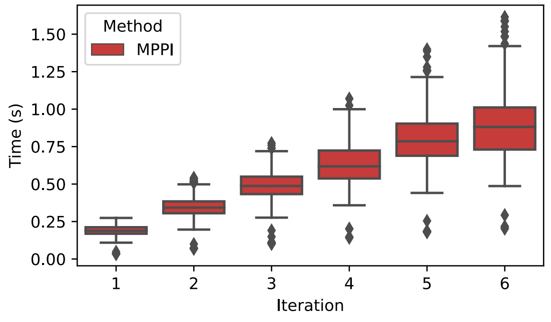
TABLE IV
RETURN AND SUCCESS RATE AT $N_{\mathrm{rollouts}} = 250$

Method	RLMPPI	RL	MPPI	CEM-MPC
Final Average Return	**129.9**	118.2	81.9	78.9
Success Rate	**98%**	90.9%	66%	65.8%

Each data point represents the average of 500 tests. The UAV's initial state is randomized for each test.



(a)



(b)

Fig. 4. Comparison between RL-driven MPPI and MPPI. Each point plot is drawn based on values of 500 runs. (a) Evolution of the average return. (b) Evolution of the average calculation time.

by fully connected NNs with the same architecture. Each NN contains five hidden layers with 256 units activated by GELU. Other detailed hyper-parameters are shown in Table III.

C. Result Analysis

Fig. 3 displays the average return (accumulated reward) of RL-driven MPPI, MPPI, and CEM-MPC, with respect to the number of rollouts N_{rollouts} per online iteration. The performance of the policy learned by DSAC is shown as a horizontal line. For the RL-driven MPPI method, $N_{\mathrm{rollouts}} = N_{\mathrm{RL}} + N_{\mathrm{MPPI}}$. Note that N_{RL} guided rollouts are collected only once at the beginning of the MPPI module (see Algorithm 1). The average return when N_{rollouts} reaches 250 is detailed in Table IV.

The results demonstrate that DSAC outperforms MPPI and CEM-MPC significantly. DSAC also exhibits superior performance compared to RL-driven MPPI when the number of rollouts $N_{\mathrm{rollouts}} \leq 50$. However, as the sample size increases, the advantage of RL-driven MPPI gradually becomes more pronounced. This is because the offline RL policy provides a nearly optimal initial estimate and serves as an effective rollout generator, resulting in significantly higher performance for RL-driven MPPI compared to MPPI and CEM-MPC with the same number of rollouts. In other words, RL-driven MPPI achieves the same average return with significantly fewer samples. Overall, RL-driven MPPI exhibits higher policy performance and sample efficiency than MPPI and CEM-MPC. Additionally, we evaluate the control success rates of each method, as shown in Table IV. The results indicate that RL-driven MPPI consistently achieves a much higher success rate compared to the other baseline methods.

Fig. 4(a) displays the return with respect to the online iteration number. It can be seen that RL-driven MPPI reaches the optimal performance after only one iteration, while MPPI requires at least five iterations. As shown in Fig. 4(b), the online calculation time is positively correlated to the iteration number. Overall,

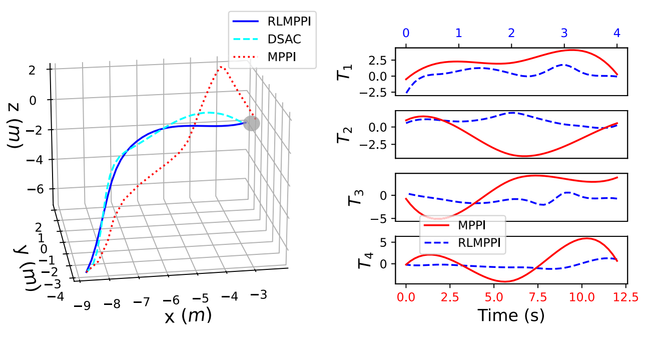


Fig. 5. Control curves of a simulation. (a) UAV trajectories of different methods. The gray sphere represents the landing area. (b) The control input of RL-driven MPPI and MPPI.

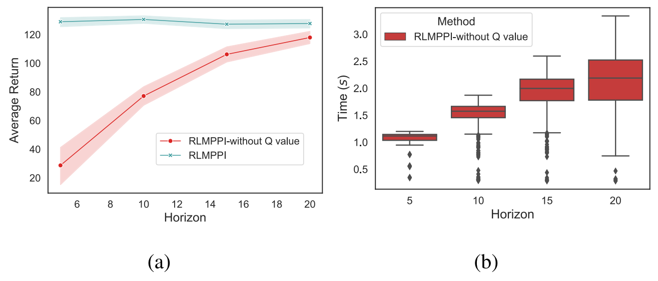


Fig. 7. Comparison between RLMPPI and RLMPPI-without Q value. Each point plot is drawn based on values of 500 runs. $N_{\text{rollouts}} = 250$. (a) Evolution of the average return. The shaded regions correspond to a 95% confidence interval. (b) Evolution of the mean completion time.

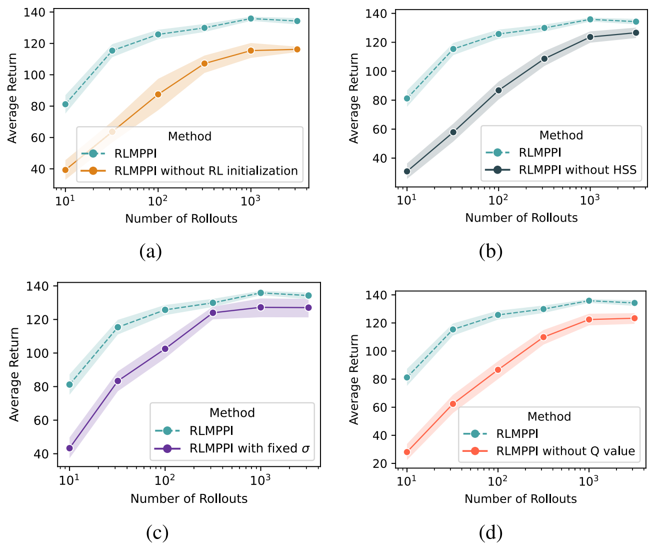


Fig. 6. Average return of RL-driven MPPI and its ablation versions in testing. (a) RLMPPI vs RLMPPI-without RL initialization. (b) RLMPPI vs RLMPPI-without HSS. (c) RLMPPI vs RLMPPI-with fixed σ. (d) RLMPPI vs RLMPPI-without Q value.

RL-driven MPPI can significantly reduce calculation time due to the shorter iteration steps and fewer rollout requirements.

Fig. 5(a) shows a numerical result generated by RL-driven MPPI, which allows the UAV to complete the waypoint approach task with a smoother trajectory and shorter completion time (see Fig. 5(b)).

D. Ablation Study

To demonstrate different improvement strategies of DSAC policy on MPPI, we propose four sub-variants of RL-driven MPPI, which ablate RL-initialization, HSS, updated σ, and Q-value-based terminal cost, respectively. These four sub-variants are named RLMPPI without RL initialization, RLMPPI without HSS, RLMPPI with fixed σ, and RLMPPI without Q-value, respectively. The test results of these algorithms are shown in Fig. 6. The relevant conclusions can be divided into four main parts:

1) The effectiveness of RL initialization: Fig. 6(a) shows that RLMPPI significantly outperforms RLMPPI without RL initialization, suggesting RL-initialization is an effective approach to improve the performance of RLMPPI.
2) The effectiveness of HSS: The result in Fig. 6(b) has shown that the algorithm with HSS has a higher average return than that without. However, when the number of rollouts becomes large enough, the algorithm can not benefit more from HSS anymore. It is reasonable because when the sample size is large enough, agents can automatically find good solutions. Overall, HSS facilitates the algorithm to quickly converge to a feasible (if not optimal) solution with fewer rollouts, thereby making the computation time more manageable.
3) The effectiveness of updating σ: Fig. 6(c) shows that in the presence of low sample size, RLMPPI benefits more from a learned σ, since it can automatically adapt to the current state. When $N_{\text{rollout}} \geq 1000$, the average return curves gradually overlap, indicating that the effect of the learned σ on performance gradually decreases.
4) The effectiveness of estimating terminal cost using Q-value: As shown in Figs. 6(d) and 7(a), leveraging Q-value as the terminal cost can lead to better performance. Theoretically, without Q-value, the policy can only be locally optimal over a finite predictive horizon, which is why RLMPPI gains significant improvement on the average return than that of RLMPPI without Q value with a shorter prediction horizon. Besides, we wish to point out that obtaining an optimal solution over a long predictive horizon may be computationally expensive (seen in Fig. 7(b)). Additionally, long predictive horizons tend to magnify modeling errors, negatively affecting performance (seen in Fig. 7(a)). The learned Q-value makes our algorithm benefit a lot of computational efficiency by estimating the discounted return over an infinite horizon.

We additionally compare the performance of RL-driven MPPI under different values of the minimum standard deviation $\sigma_{\min}$, the inverse temperature λ, and the quantity of samples used in formulating the new control iterate as per (18). The minimum standard deviation is employed to prevent premature convergence during the online MPPI process. Fig. 8(a) graphs the average return under different values of $\sigma_{\min}$. The result

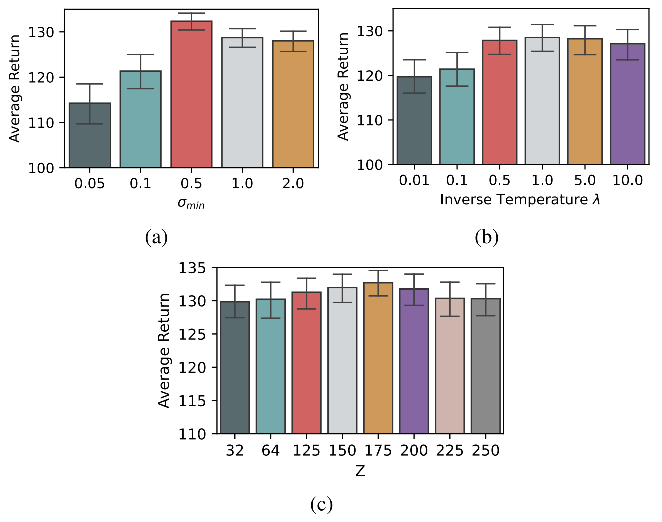


Fig. 8. Performance under different hyperparameters. (a) Average return under different $\sigma_{\min}$. (b) Average return under different λ. (c) Average return under different Z.

TABLE V
EFFECT OF VARYING WEIGHTS ON DIFFERENT PERFORMANCE COMPONENTS

Weight	Value	Finish time [s]	$\mathbb{E}[\|\Delta p\|]$ [m]	$\mathbb{E}[\|\nu\|]$ [m/s]	$\mathbb{E}[\|w\|]$ [deg/s]	$\mathbb{E}[e_q]$
ϖ_p	0.05	5.95	8.10	18.85	4.87	10.85
	0.02	5.37	8.59	18.05	4.69	10.72
	0.01	6.11	8.95	18.30	4.63	10.33
ϖ_ν	0.02	6.54	8.70	16.63	4.65	10.53
	0.01	5.37	8.59	18.05	4.69	10.72
	0.005	5.25	8.55	20.34	4.57	10.44
ϖ_w	0.005	5.78	8.66	18.70	4.53	10.79
	0.001	5.37	8.59	18.05	4.69	10.72
	0.0002	6	8.56	17.84	4.96	10.66
ϖ_q	0.005	5.67	8.45	18.62	4.99	10.66
	0.001	5.37	8.59	18.05	4.69	10.72
	0.0002	5.77	8.63	18.68	4.58	10.77

Each value is an average derived from 500 trials with random starting positions. We let $\Delta P = p - P_{\text{Target}}$.

shows that when $\sigma_{\min} = 0.5$, RL-driven MPPI achieves the best performance. On the other hand, the inverse temperature, λ, influences the proportional weight assigned to each rollout cost. A lower λ encourages MPPI to focus more on a select few rollouts, while an excessively high λ tends to evenly weight all rollouts. The result shown in Fig. 8(b) indicates that when $\lambda = 1$, RL-driven MPPI yields the best performance. RL-driven MPPI utilizes only the top-Z samples for policy update, a method that has been effectively employed in various studies [32], [33]. As depicted in Fig. 8(c), RL-driven MPPI exhibits enhanced performance when Z is set between 125 and 200, out of the total 250 candidate samples. However, it is important to note that a higher Z value can increase the computational load for each iteration. Therefore, a Z value of 125 is chosen to ensure effective computation without significantly compromising performance.

To assess the influence of the weighting coefficients ϖ_p, ϖ_ν, ϖ_w, and ϖ_q in (22) on control performance, we conduct an experiment where we set a baseline for these coefficients ($\varpi_p = 0.02$, $\varpi_\nu = 0.01$, $\varpi_w = 0.001$, and $\varpi_q = 0.001$) and alter only one coefficient at a time during the MPPI runs. Table V presents the average time taken by the UAV to reach the goal in each episode, along with the performance component associated with each reward term in (22). Essentially, selecting the reward coefficients involves balancing different aspects of performance to meet specific operational requirements.

V. CONCLUSION

In this paper, we present RL-driven MPPI, an online planning framework created with the goal of enhancing the practicality of traditional MPPI in terms of computational efficiency and control performance. Our approach involves utilizing a DSAC agent, which is acquired through offline training to serve as a sample generator, providing an initial estimate, and gauging the terminal value. Our simulations indicate that compared to MPPI, RL-driven MPPI's computational efficiency is significantly improved, with the method requiring fewer sample sizes, iteration steps, and a shorter predictive horizon. Furthermore, RL-driven MPPI delivers competitive return performance. We evaluate the efficacy of different DSAC policy enhancement strategies on MPPI. In future work, we plan to extend RL-driven MPPI to solve general optimal control problems with inequality constraints. We expect that, due to the constant advancement of computing with GPUs, sampling-based optimized structures can, in the future, be suited to real UAV applications.

APPENDIX A
DYNAMICS OF UAV

The UAV simulator is constructed based on the continuous-time model $f_c(x, v)$ as introduced in [34]. This model is characterized by

$$\begin{aligned}
\dot{p}^{\mathrm{n}} &= R_{\mathrm{b}}^{\mathrm{n}} \nu^{\mathrm{b}} \\
m\dot{\nu}^{\mathrm{b}} &= R_{\mathrm{n}}^{\mathrm{b}} f_{\mathrm{g}}^{\mathrm{n}} + f_{\text{thrust}}^{b} \\
\dot{q}_{\mathrm{b,n}} &= \frac{1}{2}\Omega\left(q_{\mathrm{b,n}}\right)\begin{bmatrix} 0 \\ w^{\mathrm{b}} \end{bmatrix} \\
J\dot{w}^{\mathrm{b}} &= -\Upsilon\left(w^{\mathrm{b}}\right) J w^{\mathrm{b}} + M^{\mathrm{b}}, \qquad (25)
\end{aligned}$$

where the superscripts b, n indicate the body frame and inertial frame, respectively, which are simplified in the notation of state and action for clarity (see Table I). $p^{\mathrm{n}} \in \mathbb{R}^3$ is the position in the inertial frame, while $\nu^{\mathrm{b}} \in \mathbb{R}^3$ and $w^{\mathrm{b}} \in \mathbb{R}^3$ represent the velocity and angular velocity in the body frame. $f_{\mathrm{g}}^{\mathrm{n}} = [0, 0, mg]^\top$ denotes the gravitational force vector. $f_{\text{thrust}}^{\mathrm{b}} = [T_1, 0, 0]^\top$ and $M^{\mathrm{b}} = [T_2, T_3, T_4]^\top$ are the control vectors representing thrust and torque in the UAV's body. $R_{\mathrm{b}}^{\mathrm{n}}$ is the transformation matrix from the body frame to the inertial frame and J is the inertia matrix (see [34] for more details). The quaternion $q_{\mathrm{b},n} = [\eta_{\mathrm{b,n}}, \mu_{\mathrm{b,n}}^\top]^\top \in \mathbb{R}^4$ describes the rotation of the body frame relative to the inertial frame, where $\eta_{\mathrm{b,n}}$ is the scalar part and $\mu_{\mathrm{b,n}}^\top$ is the vector part. The quaternion multiplication operator

$\Omega(q_{\mathrm{b,n}})$ is defined as:

$$\Omega\left(q_{\mathrm{b,n}}\right) := \begin{bmatrix} \eta_{\mathrm{b,n}} & -\mu_{\mathrm{b,n}}^{\top} \\ \mu_{\mathrm{b},n}^{\top} & \eta_{\mathrm{b,n}}\mathrm{I} + \Upsilon\left(\mu_{\mathrm{b,n}}\right) \end{bmatrix}, \tag{26}$$

where

$$\Upsilon\left(w^{\mathrm{b}}\right) := \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix}. \tag{27}$$

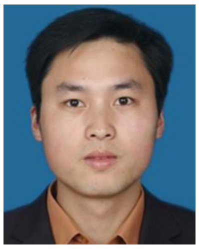
REFERENCES

[1] S. E. Li, *Reinforcement Learning for Sequential Decision and Optimal Control.*, Singapore: Springer, 2023.
[2] J. Duan et al., "Relaxed actor-critic with convergence guarantees for continuous-time optimal control of nonlinear systems," *IEEE Trans. Intell. Veh.*, vol. 8, no. 5, pp. 3299–3311, May 2023.
[3] J. Ma, Z. Cheng, X. Zhang, M. Tomizuka, and T. H. Lee, "Alternating direction method of multipliers for constrained iterative LQR in autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 23031–23042, Dec. 2022.
[4] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM J. Optim.*, vol. 9, no. 4, pp. 877–900, 1999.
[5] Z. Liu et al., "Recurrent model predictive control: Learning an explicit recurrent controller for nonlinear systems," *IEEE Trans. Ind. Electron.*, vol. 69, no. 10, pp. 10437–10446, Oct. 2022.
[6] J. Lu, L. Han, Q. Wei, X. Wang, X. Dai, and F.-Y. Wang, "Event-triggered deep reinforcement learning using parallel control: A case study in autonomous driving," *IEEE Trans. Intell. Veh.*, vol. 8, no. 4, pp. 2821–2831, Apr. 2023.
[7] G. Li, Y. Yang, S. Li, X. Qu, N. Lyu, and S. E. Li, "Decision making of autonomous vehicles in lane change scenarios: Deep reinforcement learning approaches with risk awareness," *Transp. Res. Part C: Emerg. Technol.*, vol. 134, 2022, Art. no. 103452.
[8] X. He, H. Yang, Z. Hu, and C. Lv, "Robust lane change decision making for autonomous vehicles: An observation adversarial reinforcement learning approach," *IEEE Trans. Intell. Veh.*, vol. 8, no. 1, pp. 184–193, Jan. 2023.
[9] J. Duan, W. Cao, Y. Zheng, and L. Zhao, "On the optimization landscape of dynamic output feedback linear quadratic control," *IEEE Trans. Autom. Control*, early access, May 12, 2023, doi: 10.1109/TAC.2023.3275732.
[10] J. Duan, J. Li, X. Chen, K. Zhao, S. E. Li, and L. Zhao, "Optimization landscape of policy gradient methods for discrete-time static output feedback," *IEEE Trans. Cybern.*, early access, Oct. 26, 2023, doi: 10.1109/TCYB.2023.3323316.
[11] Y. Yin, S. E. Li, K. Tang, W. Cao, W. Wu, and H. Li, "Approximate optimal filter design for vehicle system through actor-critic reinforcement learning," *Automot. Innov.*, vol. 5, no. 4, pp. 415–426, 2022.
[12] W. Wang et al., "GOPS: A general optimal control problem solver for autonomous driving and industrial control applications," *Commun. Transp. Res.*, vol. 3, 2023, Art. no. 100096.
[13] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[15] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[16] J. Duan, Y. Guan, S. E. Li, Y. Ren, Q. Sun, and B. Cheng, "Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 11, pp. 6584–6598, Nov. 2022.
[17] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1433–1440.
[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] G. F. Lawler, *Introduction to Stochastic Processes.*, London, U.K.: Chapman & Hall/CRC, 2018.
[20] G. Williams et al., "Information theoretic MPC for model-based reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 1714–1721.
[21] Z. Wang, G. Williams, and E. A. Theodorou, "Information theoretic model predictive control on jump diffusion processes," in *Proc. IEEE Amer. Control Conf.*, 2019, pp. 1663–1670.
[22] G. Williams, B. Goldfain, P. Drews, K. Saigol, J. M. Rehg, and E. A. Theodorou, "Robust sampling based model predictive control with sparse objective information," in *Proc. Robot.: Sci. Syst.*, 2018.
[23] M. S. Gandhi, B. Vlahov, J. Gibson, G. Williams, and E. A. Theodorou, "Robust model predictive path integral control: Analysis and performance guarantees," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 1423–1430, Apr. 2021.
[24] I. M. Balci, E. Bakolas, B. Vlahov, and E. Theodorou, "Constrained covariance steering based tube-MPPI," in *Proc. IEEE Amer. Control Conf.*, 2022, pp. 4197–4202.
[25] 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.
[26] Z. Wang et al., "Variational inference MPC using Tsallis divergence," 2021, *arXiv:2104.00241*.
[27] 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.
[28] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.
[29] J. Duan, W. Wang, L. Xiao, J. Gao, and S. E. Li, "DSAC-T: Distributional soft actor-critic with three refinements," 2023, *arXiv:2310.05858*.
[30] J. Zhao, M. Mao, X. Zhao, and J. Zou, "A hybrid of deep reinforcement learning and local search for the vehicle routing problems," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 11, pp. 7208–7218, Nov. 2021.
[31] Y. Zhu, Z. Li, F. Wang, and L. Li, "Control sequences generation for testing vehicle extreme operating conditions based on latent feature space sampling," *IEEE Trans. Intell. Veh.*, vol. 8, no. 4, pp. 2712–2722, Apr. 2023.
[32] M. Kobilarov, "Cross-entropy motion planning," *Int. J. Robot. Res.*, vol. 31, no. 7, pp. 855–871, 2012.
[33] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo Method.*, Hoboken, NJ, USA: Wiley, 2016.
[34] E. Oland, R. Schlanbusch, and R. Kristiansen, "Underactuated waypoint tracking of a fixed-wing UAV," *IFAC Proc. Volumes*, vol. 46, no. 30, pp. 126–133, 2013.
[35] J. B. Kuipers, *Quaternions and Rotation Sequences: A Primer With Applications to Orbits, Aerospace, and Virtual Reality.*, Princeton, NJ, USA: Princeton Univ. Press, 1999.

Yue Qu received the B.S. degree in mechanical engineering from the Nanjing University of Science and Technology, Nanjing, China, in 2017, and the Ph.D. degree in control science and engineering from the Nanjing University of Science and Technology, Nanjing, China, in 2023. She is currently a Lecturer with the Nanjing Vocational University of Industry Technology, Nanjing. Her research interests include model predictive control and reinforcement learning with applications to unmanned aerial vehicles.

Hongqing Chu received the B.S. degree in vehicle engineering from the Shandong University of Technology, Zibo, China, in 2012, and the Ph.D. degree in control theory and control engineering from Jilin University, Changchun, China, in 2017. He is currently an Assistant Professor with Tongji University, Shanghai, China. His research interests include optimal control of intelligent vehicle, advanced motion control of intelligent chassis, and advanced driver assistance system.

Shuhua Gao received the B.S. degree in mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2012, the M.S. degree in mechanical engineering from Beihang University, Beijing, China, in 2015, and the Ph.D. degree in electrical and computer Engineering from the National University of Singapore, Singapore, in 2020. He is currently a Researcher with the School of Control Science and Engineering, Shandong University, Jinan, China. His research interests include control theory and learning systems and their engineering applications.



Jun Guan received the Ph.D. degree from the National Key Laboratory Transient Physics, Nanjing University of Science and Technology, Nanjing, China, in 2018. He is currently an Associate Professor with the Jiangsu University of Science and Technology, Zhenjiang, China. His research interests include control theory and machine learning and their engineering applications.

Haoqi Yan received the B.S. degree in 2021 from the School of Mechanical Engineering, University of Science and Technology Beijing, Beijing, China, where he is currently working toward the Ph.D. degree in mechanical engineering. His research interests include reinforcement learning, optimal control, and their applications in robotic arm control.

Liming Xiao received the B.S. degree in vehicle engineering in 2023 from the University of Science and Technology Beijing, Beijing, China, where he is currently working toward the M.S. degree in mechanical engineering. His research interests include reinforcement learning, optimal control, and self-driving decision-making.

Shengbo Eben Li (Senior Member, IEEE) received the M.S. and Ph.D. degrees from Tsinghua University, Beijing, China, in 2006 and 2009, respectively. He was with the Stanford University, Stanford, CA, USA, University of Michigan, Ann Arbor, MI, USA, and University of California, Berkeley, Berkeley, CA, USA. He is currently a tenured Professor with Tsinghua University. His research interests include intelligent vehicles and driver assistance, reinforcement learning and distributed control, and optimal control and estimation. He is an Associated Editor for *IEEE Intelligent Transportation Systems Magazine* and IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.

Jingliang Duan (Member, IEEE) received the Ph.D. degree with the School of Vehicle and Mobility, Tsinghua University, Beijing, China, in 2021. He was a visiting student Researcher with the Department of Mechanical Engineering, University of California, Berkeley, Berkeley, CA, USA, in 2019, and Research Fellow with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, from 2021 to 2022. He is currently an Associate Professor with the School of Mechanical Engineering, University of Science and Technology Beijing, Beijing, China. His research interests include reinforcement learning, optimal control, and self-driving decision-making.