Probabilistic Model-Based Reinforcement Learning Unmanned Surface Vehicles Using Local Update Sparse Spectrum Approximation

Yunduan Cui, *Member, IEEE*, Wenbo Shi, Huan Yang, *Member, IEEE*, Cuiping Shao, Lei Peng, *Member, IEEE*, and Huiyun Li, *Senior Member, IEEE*

***Abstract*—In this article, we focus on the computational efficiency of probabilistic model-based reinforcement learning (MBRL) in unmanned surface vehicles (USV) under unforeseeable and unobservable external disturbances. A novel MBRL approach, local update spectrum probabilistic model predictive control (LUSPMPC), is proposed to fully release the superiority of the probabilistic model approximated in the frequency domain in computational efficiency while mitigating its risk of overfitting during the learning procedure. It employs a local update strategy to relieve the violation of Bochner's theory, and a frequency clipping trick to encourage the approximated model to focus on the features in the low-frequency domain. Evaluated by the position-keeping task in a real USV data-driven simulation, LUSPMPC shows its significant advantages in computational efficiency while achieving better learning capability, generalization capability, and control performances in a wide range of sparse scales compared with the baseline MBRL approaches that approximate their models in sample space and frequency domain, and therefore becomes an appealing solution for MBRL USV system defending against rapidly changing ocean disturbances.**

***Index Terms*—Gaussian processes (GP), reinforcement learning (RL), unmanned surface vehicles (USV).**

Manuscript received 28 February 2022; revised 12 December 2022 and 3 April 2023; accepted 30 April 2023. Date of publication 8 May 2023; date of current version 19 January 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62103403 and Grant 62004212, in part by National Key Research and Development Program of China under Grant 2020YFB2104300, and in part by Guangdong Provincial Basic and Applied Basic Research Foundation under Grant 2020B515130004. Paper no. TII-22-0884. *(Corresponding author: Huiyun Li.)*

Yunduan Cui, Cuiping Shao, Lei Peng, and Huiyun Li are with the CAS Key Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China, and also with the Guangdong-Hong Kong-Macao Joint Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: cuiyunduan@gmail.com; cp.shao@siat.ac.cn; lei.peng@siat.ac.cn; hy.li@siat.ac.cn).

Wenbo Shi and Huan Yang are with the College of Computer Science and Technology, Qingdao University, Qingdao 266071, China (e-mail: shi_wenbo@outlook.com; cathy_huanyang@hotmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2023.3274229.

Digital Object Identifier 10.1109/TII.2023.3274229

I. Introduction

BOTH industry and academia have been rapidly developing unmanned surface vehicles (USV) that offer advantages in safety and efficiency in order to tackle the shortage of skilled professionals in the growing marine shipping industry [1]. The corresponding autonomous technologies of USV have been abundantly studied in various scenarios, such as position control path following and collision avoidance [2], [3], [4], [5], [6], [7], [8]. Although human intervention is usually required in the current works, e.g., the heuristic selection of parameters based on human experiences toward the excellent controller for specific USV tasks [3] and the data-driven neural networks controller achieved by massive sampling on a human knowledge-based numerical USV model [9], the fully autonomous USV have been a long-term ambition [1].

As one appealing solution to a self-learning fully autonomous USV, reinforcement learning (RL) [10] drives the agent to learn control policies through trial-and-error interactions with the unknown environment. Despite the detailed evaluations of the popular model-free RL algorithms in the USV domain by simulations [11], [12], [13], [14], [15], the corresponding implementation in the real-world USV remains limited due to the poor capability of handling the uncertain disturbances caused by the real ocean environments. Based on the successful implementations of Gaussian processes (GP) [16] in predicting disturbances [17] and modeling underwater vehicle dynamics [18] in ocean environments, it turns to a potential modeling approach in the model-based RL (MBRL) to tackle this issue by describing the system uncertainties as a collection of Gaussian distributions. Probabilistic inference for learning control (PILCO) [19] employed the propagation of GP uncertainties in long-term prediction using analytic moment matching [20] and achieved great control performances in systems with uncertainties. However, this method is not applicable to USV as its closed-loop feedback policy with full horizon planning is computationally demanding to properly alleviate the unforeseeable and frequently changing disturbances in the ocean environment. Kamthe and Deisenroth [21] proposed Gaussian process-based model predictive control (GP-MPC) [21] by introducing the model predictive control (MPC) into PILCO. It successfully moderated the real-time disturbances in simulated cart–pole and double pendulum tasks. Extended from GP-MPC, sample-efficient probabilistic model

predictive control (SPMPC) [22] successfully developed an MBRL system specialized for USV to learn the position control policy of a regular size boat in the real ocean environment without human's prior knowledge.

On the other hand, the computational complexity of the GP model was not well addressed in the existing approach [22] that turned to a difficult balance between data capacity and control performance by employing the traditional sparse approach over the sample space [23]. Based on the outstanding performance of approximating the kernel functions in the frequency domain [24], the sparse spectrum GP (SSGP) achieved a superior tradeoff between predictive accuracy and computational complexity compared with the traditional sparse approaches [25]. Pan et al. [26] successfully extended it to autonomous driving tasks by proposing the analytic moment matching in the frequency domain. Despite the great potential of these approaches in both theory and engineering, their current research mainly focused on a fixed data set, and remains a long distance to the MBRL USV system.

In this article, we tackle the computational complexity issue of the current GP-based MBRL approaches by proposing a novel MBRL approach based on the SSGP model, local update spectrum probabilistic model predictive control (LUSPMPC), that stably learns the approximation of GP kernel functions in the frequency domain from the iteratively generated samples of RL. According to the existing approaches using SSGP that optimize their frequency samples without constraints on a fixed data set [25], [26], a local update strategy that limits the overlarge change of a sampled frequency in optimization is proposed to relieve the violation of Bochner's theory caused by capriciously changing the frequency samples in the iterative learning framework of RL, which negatively affects the learning capability of MBRL approach using SSGP model due to the high risk of overfitting. A frequency clipping trick is further proposed to encourage the MBRL agent to approximate the kernel functions by features in the low-frequency domain to mitigate the effects of high-frequency noise in the engineering application on USV. The proposed approach was implemented in the MBRL USV system and evaluated by a position-keeping task in the real USV data-driven simulation following the existing work [22]. The experimental results indicated the significant advantages of the proposed method in learning capability and control performance compared with the direct combination of the existing MBRL work with the SSGP model [25], [26]. Compared with the existing work using sparse GP [22] that had been successfully implemented to the real-world USV, our approach enjoyed superior control performances while reducing about 30%–80% computational cost with a wide range of the sparse scales.

According to the relationship between our approach and the related works concluded in Table I, our work integrates SSGP into the MBRL system specified to USV with the following contributions.

1) On the side of algorithm, the existing works SSGP [25] and SSGP-exact moment matching (EMM) [26] focus on the theory of SSGP and its extension to analytic moment matching [20], which are irrelevant to RL. LUSPMPC relieves the risk of overfitting by introducing additional constraints in the optimization of frequency samples and, therefore, expands the application of SSGP in the domain of RL with superior computational efficiency and learning capability.
2) On the side of USV control, the proposed method attempts to integrate SSGP to the existing probabilistic MBRL framework specialized for USV [21], [22]. Evaluated by a simulation driven by real USV data, it demonstrates the advantages of better control performance and less computational complexity compared with existing works. Enjoying both a probabilistic model of system uncertainties in the ocean environment and rapid control against disturbances with limited computational resources, this work expands the potential of MBRL as an emerging direction of fully autonomous USV.

TABLE I
COMPARISON OF THE PROPOSED AND RELATED APPROACHES

Approach	Frequency	Analytical plan	RL	USV
SSGP [25]	◯	×	×	×
SSGP-EMM [26]	◯	◯	×	×
GP-MPC [21]	×	◯	◯	×
SPMPC [22]	×	◯	◯	◯
LUSPMPC (ours)	◯	◯	◯	◯

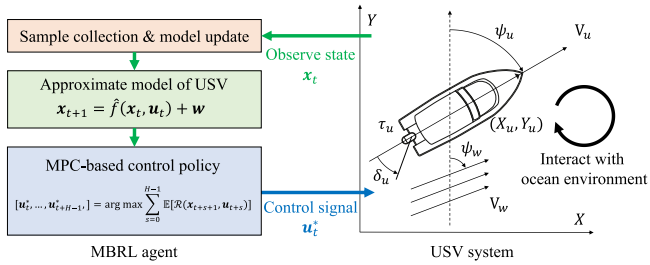


Fig. 1. Overview of MBRL USV control problem studied in this work.

The rest of this article is organized as follows. The target MBRL problem of USV control was formulated in Section II. Section III detailed the proposed LUSPMPC. The experimental results were demonstrated in Section IV. Finally, Section V concludes this article.

II. Problem Statement

We state the MBRL problem of USV control in this section. Following the overview described in Fig. 1, the USV system observes states including its global position (X_u, Y_u), heading Ψ_u, velocity V_u, current engine throttle τ_u, and rudder angle δ_u, where the subscript u of these states indicates "USV." The system detects the wind direction and velocity Ψ_w, V_w, where the subscript w indicates "wind." Although the system is also affected by other disturbances, such as currents and waves, we consider them as unobservable noises in this work.

1) Markov Decision Process (MDP): We usually describe the MBRL problem as an MDP [10]. The state space $\mathcal{S}$ and the action space $\mathcal{A}$ are defined as the sets of the observed state $\boldsymbol{x}_t = [X_u^t, Y_u^t, \Psi_u^t, V_u^t, \tau_u^t, \delta_u^t, V_w^t \cdot \cos(\Psi_w^t), V_w^t \cdot \sin(\Psi_w^t)]$, and the

control signal $\boldsymbol{u}_t = [\tau_o^t, \delta_o^t]$, where t indicates the time step. The conditional distribution of the state in the next time step given the current state and action is predicted by the system model $f(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)$. The MBRL agent aims to learn a policy $\pi : \boldsymbol{x}_t \to \boldsymbol{u}_t$ that maximizes a task-related reward function $\mathcal{R}(\boldsymbol{x}_{t+1}, \boldsymbol{u}_t)$, which concerns the next step state and the current action.

2) Modeling the System: The MBRL agent iteratively learns an approximated system model $\hat{f}(\cdot)$ following the existing approaches [19], [22], [27]:

$$\boldsymbol{x}_{t+1} = \hat{f}(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{w} \tag{1}$$

where the noisy signals $\boldsymbol{w}$ represent the unobservable disturbances, such as current and wave. Different from the model-free RL updating value function and policy via Bellman equations, MBRL updates its agent by training the model with explored samples.

3) Planning Using Learned Model: The MBRL policy in this article is implemented by MPC following the existing works [21], [22], [27]. Given the current state $\boldsymbol{x}_t$ and the prediction horizon H, the MPC controller decides an action sequence to maximize the expected long-term reward

$$[\boldsymbol{u}_t^*, \ldots, \boldsymbol{u}_{t+H-1}^*] = \arg\max_{\boldsymbol{u}_t, \ldots, \boldsymbol{u}_{t+H-1}} \sum_{s=0}^{H-1} \mathbb{E}[\mathcal{R}(\boldsymbol{x}_{t+s+1}, \boldsymbol{u}_{t+s})].$$
$$\text{s.t.} \quad \boldsymbol{x}_{t+s+1} \in \mathcal{S}, \boldsymbol{u}_s \in \mathcal{A}. \tag{2}$$

At each step, $\mathbb{E}[\cdot]$ calculates the exception of a reward function $\mathcal{R}(\boldsymbol{x}_{t+s+1}, \boldsymbol{u}_{t+s})$, and the future state $\boldsymbol{x}_{t+s+1}$ is predicted by the learned model following (1) given the initial state $\boldsymbol{x}_t$ and the candidate control sequence $[\boldsymbol{u}_t, \ldots, \boldsymbol{u}_{t+H-1}]$. Following the MPC settings in existing works [22], [27], the states are not manually restricted, and the ranges of the action sequence are regularized within $[-1, 1]$. Once the optimal action sequence $[\boldsymbol{u}_t^*, \ldots, \boldsymbol{u}_{t+H-1}^*]$ is obtained, the MPC controller executes $\boldsymbol{u}_t^*$ and moves to the next step $t+1$. This process is repeated as a closed-loop controller that implicitly forms a policy $\boldsymbol{u}_t^* = \pi(\boldsymbol{x}_t)$. Note that this MPC-based policy does not have the model identification procedure in advance, and its performances, therefore, strongly depend on the quality of the model learned by RL.

4) Updating Agent: Following the demonstration in Fig. 1, the MBRL agent in this work can be divided into two components: the approximate system model that predicts future states based on the current observed states, and the MPC-based policy that plans control signals to maximize the long-term reward based on the observed states and the approximate model. Define the data set as $\mathcal{D}$, the MBRL agent is updated by periodically retaining the approximated system model following the loop given as follows.

1) Optimize $\boldsymbol{u}_t^*$ by the MPC-based policy based on the observation in the previous step $\boldsymbol{x}_t$.
2) Execute $\boldsymbol{u}_t^*$, and drive the USV with the ocean environment and observe the next state $\boldsymbol{x}_{t+1}$.
3) Expand data set $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{x}_t, \boldsymbol{u}_t^*, \boldsymbol{x}_{t+1}\}$.
4) Set $t = t+1$, return to 1).

III. Approach

A. Learning SSGP Model

The original GP is a collection of random variables, any finite number of which have a joint Gaussian distribution [16]. Define $\tilde{\boldsymbol{x}}_t := (\boldsymbol{x}_t, \boldsymbol{u}_t)$ and combine the input states $\boldsymbol{x}_t \in \mathbb{R}^S$ and the control signals $\boldsymbol{u}_t \in \mathbb{R}^C$, $\boldsymbol{y}_{t+1} \in \mathbb{R}^O$ is the observed state in the next step, $w_a \sim \mathcal{N}(0, \sigma_{w_a}^2)$ as Gaussian noise, GP separately models the system for each dimension of the observed state $a = 1, \ldots, O$ as $\hat{f}_a$ following:

$$\boldsymbol{y}_{t+1}^a = \hat{f}_a(\tilde{\boldsymbol{x}}_t) + w_a. \tag{3}$$

The unobservable disturbances $\boldsymbol{w}$ are assumed to follow Gaussian distributions in GP for an analytical calculation. Despite the difficulty of fully describing disturbances following unknown distributions, GP is an ideal modeling approach of USV according to its successful applications in ocean engineering [17], [18]. Define $\sigma_{f_a}^2$ as variance and $\tilde{\boldsymbol{\Lambda}}_a$ as the diagonal matrix of squared length-scales, GP is specified by a covariance squared exponential (SE) kernel of two state-action pairs $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$

$$k_a(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j) = \sigma_{f_a}^2 \exp\left(-\frac{1}{2}(\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j)^T \tilde{\boldsymbol{\Lambda}}_a^{-1} (\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j)\right). \tag{4}$$

Set n training samples as input set $\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_n]$ and target set $\boldsymbol{Y} = [\boldsymbol{y}_2, \ldots, \boldsymbol{y}_{n+1}]$, the hyper parameters of GP $\theta_a = [\sigma_{f_a}^2, \tilde{\boldsymbol{\Lambda}}_a, \sigma_{w_a}^2]$ are learned by maximizing the log marginal likelihood [16]

$$\log p(\boldsymbol{Y}_a|\theta_a) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}|\boldsymbol{K}^a + \sigma_{w_a}^2 \boldsymbol{I}^2| - \frac{1}{2}\boldsymbol{Y}_a^T \boldsymbol{\beta}_a \tag{5}$$

where $\boldsymbol{K}^a$ is the matrix with elements $K_{i,j}^a = k_a(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j)$ calculated in (4) over $\tilde{\boldsymbol{X}}$, and $\boldsymbol{\beta}_a = (\boldsymbol{K}^a + \sigma_{w_a}^2 \boldsymbol{I})^{-1} \boldsymbol{Y}_a$. Given a new input $\tilde{\boldsymbol{x}}_*$ and define $\boldsymbol{k}_{a*} = k_a(\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{x}}_*)$ and $k_{a**} = k_a(\tilde{\boldsymbol{x}}_*, \tilde{\boldsymbol{x}}_*)$, its posterior mean and variance are calculated by

$$m_{f_a}(\tilde{\boldsymbol{x}}_*) = \boldsymbol{k}_{a*}^T (\boldsymbol{K}^a + \sigma_{w_a}^2 \boldsymbol{I})^{-1} \boldsymbol{Y}_a = \boldsymbol{k}_{a*}^T \boldsymbol{\beta}_a \tag{6}$$

$$\sigma_{f_a}^2(\tilde{\boldsymbol{x}}_*) = k_{a**} - \boldsymbol{k}_{a*}^T (\boldsymbol{K}^a + \sigma_{w_a}^2 \boldsymbol{I})^{-1} \boldsymbol{k}_{a*}. \tag{7}$$

The original GP provides a data-driven probabilistic model of a target system with stability: a control system based on a full GP dynamics model with SE kernel function has been proved to converge to a unique and invariant limiting distribution from any start states [28].

Due to the polynomial computational complexity of the sample size [16], GP is not directly applicable to large datasets. Compared with the traditional sparse approach [23] that reduces the burden of computing kernel by selecting pseudo inputs in the sample space, SSGP [25] simplifies the calculation of GP in the view of frequency domain, as described in Fig. 2. Define $\sigma_{k_a} = \frac{\sigma_{f_a}}{\sqrt{m}}$, the SE kernel in (4) can be properly approximated by m random frequencies $\{\omega_{a,i}\}_{i=1}^m$ sampled independently following $p(\omega_a) \sim \mathcal{N}(\boldsymbol{0}, \tilde{\boldsymbol{\Lambda}}_a)$ according to Bochner's theorem [24]

$$k_a(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j) = \int p(\omega_a) e^{j\omega_a^T(\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j)} \mathrm{d}\omega_a \approx \frac{1}{m} \sum \boldsymbol{\phi}_a(\tilde{\boldsymbol{x}}_i) \boldsymbol{\phi}_a(\tilde{\boldsymbol{x}}_j)^T \tag{8}$$

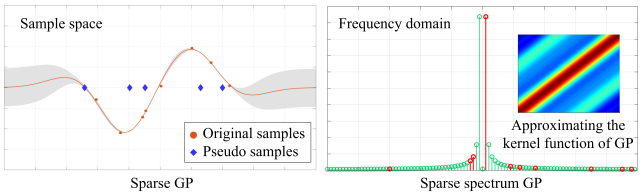


Fig. 2. Principle of the sparse GP [23] that selects pseudoinputs in sample space and SSGP [25] that approximates kernel in the frequency domain.

where $\boldsymbol{\omega}_a$ is the matrix of $\{\omega_{a,i}\}_{i=1}^m$, and the feature vector of any state-action pair $\boldsymbol{\phi}_a(\tilde{\boldsymbol{x}})$ is calculated as

$$\boldsymbol{\phi}_a(\tilde{\boldsymbol{x}}) = \begin{bmatrix} \sigma_{k_a}\cos(\boldsymbol{\omega}_a^T\tilde{\boldsymbol{x}}) \\ \sigma_{k_a}\sin(\boldsymbol{\omega}_a^T\tilde{\boldsymbol{x}}) \end{bmatrix}. \tag{9}$$

Applying linear Gaussian model [29], the posterior mean and variance are calculated as follows:

$$m_{f_a}(\tilde{\boldsymbol{x}}_*) = \boldsymbol{\phi}_a(\tilde{\boldsymbol{x}}_*)^T\boldsymbol{A}_a^{-1}\boldsymbol{\Phi}_a\boldsymbol{Y}_a = \boldsymbol{\phi}_a(\tilde{\boldsymbol{x}}_*)^T\boldsymbol{\alpha}_a \tag{10}$$

$$\sigma_{f_a}^2(\tilde{\boldsymbol{x}}_*) = \sigma_{w_a}^2\boldsymbol{\phi}_a(\tilde{\boldsymbol{x}}_*)^T\boldsymbol{A}_a^{-1}\boldsymbol{\phi}_a(\tilde{\boldsymbol{x}}_*) + \sigma_{w_a}^2 \tag{11}$$

where $\boldsymbol{\Phi}_a = [\boldsymbol{\phi}_a(\tilde{\boldsymbol{x}}_1), \ldots, \boldsymbol{\phi}_a(\tilde{\boldsymbol{x}}_n)]$, $\boldsymbol{A}_a = \boldsymbol{\Phi}_a\boldsymbol{\Phi}_a^T + \frac{m\sigma_{w_a}^2}{\sigma_{f_a}^2}\boldsymbol{I}$, and $\boldsymbol{\alpha}_a = \boldsymbol{A}_a^{-1}\boldsymbol{\Phi}_a\boldsymbol{Y}_a$. Including the randomly generated frequencies in the hyperparameters $\theta_a = [\sigma_{f_a}^2, \tilde{\boldsymbol{\Lambda}}_a, \sigma_{w_a}^2, \boldsymbol{\omega}_a]$, the learning of SSGP are achieved by maximizing the following log marginal likelihood function with $\gamma_a = \boldsymbol{Y}_a^T\boldsymbol{Y}_a - \boldsymbol{Y}_a^T\boldsymbol{\Phi}_a^T\boldsymbol{\alpha}_a$:

$$\log p(\boldsymbol{Y}_a|\theta_a) = -\frac{\gamma_a}{2\sigma_{w_a}^2} - \frac{1}{2}\log|\boldsymbol{A}_a| + m\log\frac{m\sigma_{w_a}^2}{\sigma_{f_a}^2} - \frac{n}{2}\log 2\pi\sigma_{w_a}^2. \tag{12}$$

B. Uncertainties Propagation Using SSGP

Analytic moment matching [20] enables the GP prediction with uncertain input by assuming the following non-Gaussian distribution as a Gaussian one processing the same mean and variance

$$p(\hat{f}(\tilde{\boldsymbol{x}}_*)|\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) = \int p\left(\hat{f}(\tilde{\boldsymbol{x}}_*)|\tilde{\boldsymbol{x}}_*\right)p(\tilde{\boldsymbol{x}}_*)\mathrm{d}\tilde{\boldsymbol{x}}_*. \tag{13}$$

Define the input as $p(\tilde{\boldsymbol{x}}_*) \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_*, \tilde{\boldsymbol{\Sigma}}_*)$, it turns to the propagation of uncertainties through function $h(\tilde{\boldsymbol{x}}_*, \tilde{\boldsymbol{\Sigma}}_*) \sim \mathcal{N}(\boldsymbol{\mu}'_*, \boldsymbol{\Sigma}'_*)$, which significantly smooths the nonlinear characteristics of GP caused by large state uncertainties and contribute to better performances of USV under unobservable disturbances according to Cui et al. [22].

The analytic moment matching specialized for SSGP was proposed in[26], and it calculates the mean of $h(\tilde{\boldsymbol{x}}_*, \tilde{\boldsymbol{\Sigma}}_*)$ by integrating the posterior mean in (10) over the input distribution $p(\tilde{\boldsymbol{x}}_*) \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_*, \tilde{\boldsymbol{\Sigma}}_*)$

$$\mu'_{a*} = \mathbb{E}_{\tilde{\boldsymbol{x}}_*}\left[m_{f_a}(\tilde{\boldsymbol{x}}_*)\right] = \mathbb{E}_{\tilde{\boldsymbol{x}}_*}\begin{bmatrix} \sigma_{k_a}\cos(\boldsymbol{\omega}_a^T\boldsymbol{x}) \\ \sigma_{k_a}\sin(\boldsymbol{\omega}_a^T\boldsymbol{x}) \end{bmatrix}^T\boldsymbol{\alpha}_a$$

$$= \begin{bmatrix} \sigma_{k_a}\exp\left(-\frac{1}{2}\|\boldsymbol{\omega}_a\|_{\tilde{\boldsymbol{\Sigma}}_*}^2\right)\cos(\boldsymbol{\omega}_a^T\boldsymbol{\mu}_*) \\ \sigma_{k_a}\exp\left(-\frac{1}{2}\|\boldsymbol{\omega}_a\|_{\tilde{\boldsymbol{\Sigma}}_*}^2\right)\sin(\boldsymbol{\omega}_a^T\boldsymbol{\mu}_*) \end{bmatrix}^T\boldsymbol{\alpha}_a. \tag{14}$$

The variances are calculated based on the law of total variance and (11) and (14)

$$\Sigma'_{a*} = \mathbb{E}_{\tilde{\boldsymbol{x}}_*}\left[\sigma_{f_a}^2(\tilde{\boldsymbol{x}}_*)\right] + \mathbb{E}_{\tilde{\boldsymbol{x}}_*}\left[m_{f_a}^2(\tilde{\boldsymbol{x}}_*)\right] - \mathbb{E}_{\tilde{\boldsymbol{x}}_*}\left[m_{f_a}(\tilde{\boldsymbol{x}}_*)\right]^2$$
$$= \sigma_{w_a}^2 + \sigma_{w_a}^2\mathrm{Tr}\left(\boldsymbol{A}_a^{-1}\boldsymbol{\Psi}_a\right) + \boldsymbol{\alpha}_a^T\boldsymbol{\Psi}_a\boldsymbol{\alpha}_a - \mu'^2_{a*} \tag{15}$$

where $\boldsymbol{\Psi}_a$ is the expectation of the feature vectors' outer product over $p(\tilde{\boldsymbol{x}}_*)$. Define $i, j = 1, \ldots, m$ as the indicate of the elements in each $m \times m$ submatrix, it can be calculated by the product-to-sum trigonometric identities introduced in [26]

$$\boldsymbol{\Psi}_a = \mathbb{E}_{\tilde{\boldsymbol{x}}_*}\left[\boldsymbol{\phi}_a(\tilde{\boldsymbol{x}}_*)\boldsymbol{\phi}_a(\tilde{\boldsymbol{x}}_*)^T\right] = \frac{\sigma_{k_a}^2}{2}\begin{bmatrix} \boldsymbol{\Psi}_a^{cc} & \boldsymbol{\Psi}_a^{cs} \\ \boldsymbol{\Psi}_a^{sc} & \boldsymbol{\Psi}_a^{ss} \end{bmatrix}$$

$$\Psi_{a,ij}^{cc} = \mathbb{E}_{\tilde{\boldsymbol{x}}_*}\left[\cos\left((\omega_{a,i}+\omega_{a,j})^T\tilde{\boldsymbol{x}}_*\right) + \cos\left((\omega_{a,i}-\omega_{a,j})^T\tilde{\boldsymbol{x}}_*\right)\right]$$

$$\Psi_{a,ij}^{ss} = \mathbb{E}_{\tilde{\boldsymbol{x}}_*}\left[\cos\left((\omega_{a,i}-\omega_{a,j})^T\tilde{\boldsymbol{x}}_*\right) - \cos\left((\omega_{a,i}+\omega_{a,j})^T\tilde{\boldsymbol{x}}_*\right)\right]$$

$$\Psi_{a,ij}^{cs} = \mathbb{E}_{\tilde{\boldsymbol{x}}_*}\left[\sin\left((\omega_{a,i}+\omega_{a,j})^T\tilde{\boldsymbol{x}}_*\right) - \sin\left((\omega_{a,i}-\omega_{a,j})^T\tilde{\boldsymbol{x}}_*\right)\right]. \tag{16}$$

According to Deisenroth et al.[19] and Pan et al. [26], the computational complexity of traditional sparse GP [23] and SSGP has the same order: $\mathcal{O}(Onm^2)$ for the training and $\mathcal{O}(Om^2)$ for the predicting, and $\mathcal{O}(O^2m^2(S+C))$ for the analytical moment matching. The superior computational efficiency of SSGP comes from not only the additional flexibility of optimizing the characteristics of kernel function via frequency that could model the system dynamics with fewer features than sparse GP, but also the heavily calculated terms $\boldsymbol{\alpha}_a$, $\boldsymbol{A}_a^{-1}$, and $\boldsymbol{\Psi}_a$ irrelevant to the input $\tilde{\boldsymbol{x}}_*$ in (14) and (15) that can be computed in advance. On the other hand, such a flexibility resulted in a high risk of overfitting, as reported in [25].

C. Local Update Spectrum Probabilistic MPC

Optimizing the matrix of frequency samples $\boldsymbol{\omega}_a$ without constraint to maximize the log marginal likelihood function in (12), SSGP [25], [26] have demonstrated the advantages of efficiency and accuracy in both GP regression and uncertainties propagation on the fixed data set of supervised learning. On the other hand, directly applying SSGP to the data set iteratively generated in an RL framework could be improper. As shown in the left-hand side of Fig. 3, the freely optimized frequency samples can be far away from their initial Gaussian distribution and violate Bochner's theorem. Repeating this optimization in each iteration, therefore, turned to a deteriorated learning capability of the MBRL agent due to the unstable GP model with capriciously changing approximated SE kernel functions.

In this section, a novel MBRL approach, LUSPMPC, was proposed to tackle this issue. It first introduced a local update strategy to relieve the violation of Bochner's theorem during the learning procedure. As shown in the right-hand side of Fig. 3, an additional term $\Delta\omega_a$ was introduced in the feature vector

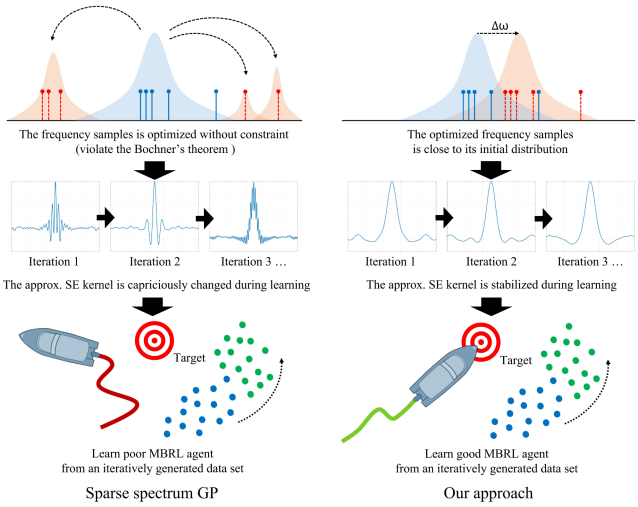


Fig. 3. Principle of the existing MBRL approach using SSGP and the proposed LUSPMPC.

Algorithm 1: RL process of LUSPMPC for USV.

Input sample set $(\tilde{\boldsymbol{X}}, \boldsymbol{Y})$, time of executing action Δt
$\hat{f}$ = `TrainGP(`$\tilde{\boldsymbol{X}}, \boldsymbol{Y}$`)`
for $i = 1, 2, ..., N_{trial}$ **do**
  $\boldsymbol{x}_0$ =`ResetState()`, $\boldsymbol{u}_0^* = [0, 0]$
  for $t = 1, 2, ..., L_{rollout}$ **do**
    # CPU 1
    `OperateActions(`$\boldsymbol{u}_{t-1}^*$`)`
    $[\boldsymbol{x}_t, \boldsymbol{y}_t]$ = `ObserveState()`
    # CPU 2
    $\hat{\boldsymbol{x}}_t = \hat{f}(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}^*)$
    search $[\boldsymbol{u}_t^*, ..., \boldsymbol{u}_{t+H-1}^*]$ with $\hat{\boldsymbol{x}}_t$ using Eq. (2)
    if $t > 1$ **then**
      $\tilde{\boldsymbol{x}}_{t-1} = (\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}^*)$
      $\tilde{\boldsymbol{X}} = \{\tilde{\boldsymbol{X}}, \tilde{\boldsymbol{x}}_{t-1}\}, \boldsymbol{Y} = \{\boldsymbol{Y}, \boldsymbol{y}_t\}$
  $\hat{f}$ = `TrainGP(`$\tilde{\boldsymbol{X}}, \boldsymbol{Y}$`)`
Return $\hat{f}$

$\boldsymbol{\Phi}_a^{\Delta} = [\boldsymbol{\phi}_a^{\Delta}(\tilde{\boldsymbol{x}}_1), \ldots, \boldsymbol{\phi}_a^{\Delta}(\tilde{\boldsymbol{x}}_n)]$ with elements

$$\boldsymbol{\phi}_a^{\Delta}(\tilde{\boldsymbol{x}}) = \begin{bmatrix} \sigma_{k_a} \cos\left((\boldsymbol{\omega}_a + \Delta\omega_a)^T \tilde{\boldsymbol{x}}\right) \\ \sigma_{k_a} \sin\left((\boldsymbol{\omega}_a + \Delta\omega_a)^T \tilde{\boldsymbol{x}}\right) \end{bmatrix}. \tag{17}$$

Instead of optimizing the frequency samples $\boldsymbol{\omega}_a$ in (9) without constraint, LUSPMPC switched $\Delta\omega_a$ within a given range $[-c_{\text{local}}, c_{\text{local}}]$ as a hyperparameter in (17) to fit the approximated SE kernel on the target data set without hugely breaking the initial distribution of $\boldsymbol{\omega}_a$. It stabilized the approximated SE kernel function during the learning procedure of MBRL agent.

LUSPMPC further employed a frequency clipping trick following existing works [30], [31] to reduce the negative effect of over high-frequency noise to the approximated SE kernel function in the engineering applications of USV. It clipped the initial frequency samples following a Gaussian distribution $p(\boldsymbol{\omega}_a) \sim \mathcal{N}(\boldsymbol{0}, \tilde{\boldsymbol{\Lambda}}_a)$ within a given range $\widehat{\boldsymbol{\omega}}_a = \text{clip}(\boldsymbol{\omega}_a, -c_{\text{clip}}, c_{\text{clip}})$, which is utilized into (17) as the fixed-frequency samples.

Overall, define $\boldsymbol{A}_a^{\Delta} = \boldsymbol{\Phi}_a^{\Delta}\boldsymbol{\Phi}_a^{\Delta T} + \frac{m\sigma_{w_a}^2}{\sigma_{f_a}^2}\boldsymbol{I}$ and $\gamma_a^{\Delta} = \boldsymbol{Y}_a^T\boldsymbol{Y}_a - \boldsymbol{Y}_a^T\boldsymbol{\Phi}_a^{\Delta T}\boldsymbol{A}_a^{\Delta -1}\boldsymbol{\Phi}_a^{\Delta}\boldsymbol{Y}_a$, the marginal likelihood function in LUSPMPC is calculated by

$$\log p(\boldsymbol{Y}_a|\theta_a^{\Delta}) = -\frac{\gamma_a^{\Delta}}{2\sigma_{w_a}^2} - \frac{1}{2}\log|\boldsymbol{A}_a^{\Delta}| + m\log\frac{m\sigma_{w_a}^2}{\sigma_{f_a}^2} - \frac{n}{2}\log 2\pi\sigma_{w_a}^2. \tag{18}$$

The hyperparameters $\theta_a^{\Delta} = [\sigma_{f_a}^2, \tilde{\boldsymbol{\Lambda}}_a, \sigma_{w_a}^2, \Delta\omega_a]$ are optimized with additional constraints as follows:

$$\begin{aligned} &\theta_a^{\Delta *} = \arg\max_{\theta_a^{\Delta}} \log p(\boldsymbol{Y}_a|\theta_a^{\Delta}) \\ &\text{s.t.} \quad \Delta\omega_a \in [-c_{\text{local}}, c_{\text{local}}] \\ &\qquad \widehat{\boldsymbol{\omega}}_a = \text{clip}(\boldsymbol{\omega}_a, -c_{\text{clip}}, c_{\text{clip}}) \\ &\qquad p(\boldsymbol{\omega}_a) \sim \mathcal{N}(\boldsymbol{0}, \tilde{\boldsymbol{\Lambda}}_a). \end{aligned} \tag{19}$$

Employing c_{local} and c_{clip} as constraints of (19), LUSPMPC could customize its kernel functions to fit the current data set without a high risk of overfitting and, therefore, enjoy superior performances than the original SSGP.

The RL process of LUSPMPC system specific for USV is detailed in Algorithm 1. At the beginning, the GP model $\hat{f}$ is initialized by a preprepared sample set $(\tilde{\boldsymbol{X}}, \boldsymbol{Y})$ following (19) where $\Delta\omega_a \in [-c_{\text{local}}, c_{\text{local}}]$ is updated with a fixed $\widehat{\boldsymbol{\omega}}_a$. The GP model will be updated in N_{trial} trials during the RL process. At each trial, the USV system is reset to an initial state $\boldsymbol{x}_0$ with a zero control action $\boldsymbol{u}_0^*$ and interacts with the environment in an L_{rollout}-step rollout. At each step t, the control signals and the MPC controller are separately executed by two CPU cores following the parallel framework introduced in [22] to alleviate the bias caused by the MPC optimization time of $\boldsymbol{u}_t^*$ when the USV continuously operates the previous control signals $\boldsymbol{u}_{t-1}^*$. CPU 1 executes $\boldsymbol{u}_{t-1}^*$ and observes the USV state $\boldsymbol{x}_t$ after Δt. $\boldsymbol{y}_t$ is obtained by removing the unpredictable wind information from $\boldsymbol{x}_t$. At the same time, CPU 2 predicts $\hat{\boldsymbol{x}}_t = \hat{f}(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}^*)$ to compensate the bias caused by $\boldsymbol{u}_{t-1}^*$ at the current step. The MPC controller searches the optimal control sequence following (2) from $\hat{\boldsymbol{x}}_t$ where the unpredictable disturbances are assumed fixed in the H-steps prediction. For $t > 1$, its initial control sequence is partially inherited from the one optimized in the previous step for better system stability, only the last control signal $\boldsymbol{u}_{t+H-1}$ is set to zeros. After obtaining the optimal control sequence $[\boldsymbol{u}_t^*, \ldots, \boldsymbol{u}_{t+H-1}^*]$, only the first control signal $\boldsymbol{u}_t^*$ will be operated in the next step by CPU 1. For step $t > 1$, the interaction samples $(\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1}^*)$ and $\boldsymbol{y}_{t-1}$ are added to sample set $(\tilde{\boldsymbol{X}}, \boldsymbol{Y})$. The GP model will be iteratively updated at the end of each trial by the current sample set as one RL loop. In the training procedure, LUSPMPC adds an exploration noise following Gaussian distribution $\boldsymbol{w}_e \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_e)$ and updates its agent after each rollout by the current data set. After training, it runs without exploration noise $\boldsymbol{w}_e$ and model update in the testing procedure.

IV. Experiments

A. Experimental Settings

In this section, the proposed method was evaluated by the USV simulation introduced in the existing MBRL approach specified for USV [22], which contributed to a successful implementation of the real-world USV system. This simulation approximated the USV dynamics in various ocean environments with different wind and current settings based on the real USV driving data. Generated following the initial distributions, the velocities and directions of both wind and current continuously change at each step following uniform distributions. In order to sufficiently evaluate the proposed MBRL approach's learning capability, generalization capability, control performance, and model quality in the USV domain with the existing baseline approach, the stochastic disturbances of ocean environment were set as $\psi_w = 37^\circ + U(-30, 30)^\circ, \psi_c = 100^\circ + U(-30, 30)^\circ, v_w = 2.0 + U(0, 0.1)$ m/s, and $v_c = 0.25 + U(0, 0.1)$ m/s based on the study and analysis in [22], where ψ_w and v_w are the angle and velocity of wind, ψ_c and v_c are the angle and velocity of the unobservable current, and U indicates the uniform distribution. Following the simulation model introduced in [22, Appendix], the influences of wind and current on the USV dynamics are independent: the movement of USV is separately affected by wind and current, whereas the orientation of USV is strongly affected by the wind. We focused on the position-keeping task in [22] where the reward function is defined as the minus squared Euclidean distance between the predicted mean of the boat position $\boldsymbol{p}_{t+1}$ and the target position $\boldsymbol{p}_{\text{target}} = [0, 0]$

$$\mathcal{R}(\boldsymbol{p}_{t+1}) = -\frac{1}{2}||\boldsymbol{p}_{t+1} - \boldsymbol{p}_{\text{target}}||^2. \tag{20}$$

Set the initial position $\boldsymbol{p}_0 = [0, 0]$, this reward function encourages the USV to keep its position by properly switching its engine throttle and rudder angle to counteract and utilize the frequently changing disturbances. The stochastic disturbances were generated within the control capability of USV following [22] so that a well-learned policy could always resist environmental disturbances.

Although (20) only considers USV's position offset, it is a proper metric of the given task as the USV's position is fully affected by its dynamics and external disturbances according to the results in [22]. Therefore, we believe it is sufficient to investigate the superiority of the proposed method in learning capability, model quality, and computational complexity over the existing work. In each trial, the initial GP model was trained by 500 samples generated by random control signals. The training took $N_{\text{trial}} = 20$ rollouts, each rollout has 50 steps. After training, the agent was tested by 30 longer rollouts with 100 steps. During each step, the MBRL agent predicts $H = 3$ steps in MPC, and the optimization time is limited within Δt. Each dimension of initial variance of MPC prediction $\boldsymbol{\Sigma}_0$ is set to 10^{-4}, and the variance of explore noise $\boldsymbol{w}_e$ is set to $\boldsymbol{\Sigma}_e = 0.1$. In the test procedure, the agent will stop collecting samples and updating its model. In total, five independent trials were conducted with different random seeds for statistical evidence.

The MBRL agent was implemented by Python based on GPflow [32] accelerated by the computational graph of Tensorflow. We utilized bound optimization by quadratic approximation [33] in NLopt[1] for the optimization of MPC. All experiments were conducted on a computational server with an Intel Xeon(R) W-2245 CPU and 64 GB memory.

B. Evaluation of Local Update and Frequency Clip

In this experiment, we evaluated the advantages of the local update and the frequency clip in LUSPMPC to the MBRL USV using the SSGP model. We denoted the combination of existing work SPMPC [22] and SSGP [25] that directly employs SSGP model with neither local update nor frequency clip by SPMPC (SSGP) as a reasonable baseline approach. In total, 24 configurations of SPMPC (SSGP) and LUSPMPC were evaluated following the right-hand side of Fig. 4. Configurations 1–4 are the SPMPC (SSGP) with different $c_{\text{clip}} = [\text{N/A}, 1.0, 0.5, 0.25]$, where N/A indicates no frequency clip. Configurations 5–24 are the LUSPMPC with $c_{\text{local}} = [1.0, 0.5, 0.25, 0.1, 0.0]$ and different c_{clip}.

The learning curve of the RL training procedure was evaluated on the left-hand side of Fig. 4. All baseline approaches (configurations 1–4) converged to larger average offsets (> 10 m), which indicate the limited learning capability of directly employing SSGP into SPMPC's RL framework. Note $c_{\text{local}} = \text{N/A}$ indicated that the local update trick was not employed and $\boldsymbol{\omega}_a$ can be freely optimized, it is inequivalent to $c_{\text{local}} = 0.0$ where $\Delta\omega_a$ was fixed during training. As a comparison, overall, the converged offset decreases with the different c_{local} from 1.0 to 0.1 (configurations 5 to 20). Configurations 21–24 that prevent optimizing frequency in (18) by setting $c_{\text{local}} = 0.0$ turn to degraded control performances than configurations 9–20 while still outperforming the baseline approaches. These results demonstrate the positive effect of c_{local} that stabilizes the approximated kernel by adding constraints to the optimization of SSGP's likelihood function. The superiority of a proper c_{clip} in improving the learning capability can also be observed from configurations 5 to 20 while only employing c_{clip} in the baseline approach without c_{local} did not work.

After training, the control performances and generalization capability of all configurations were evaluated in the test procedure. The average offsets and median offsets were presented in Table II. The success rates (final/overall) were defined as the average rates of successfully holding the distance between the USV and the target within 7 m threshold at the end of rollout/during the whole rollout. The threshold is selected close to the length of the simulated boat introduced in [22] (7.93 m), which turns into a reasonable metric of successfully keeping the USV's position. The learned models' quality was investigated in Fig. 5 including the predicted errors of the position in *X*- and *Y*-axes, orientation, and velocity in one-step prediction.

We can observe that directly applying frequency clip to SPMPC (SSGP) (configurations 1–4) improves neither the control performance nor the model quality. Configurations 1–4

[1][Online]. Available: http://github.com/stevengj/nlopt

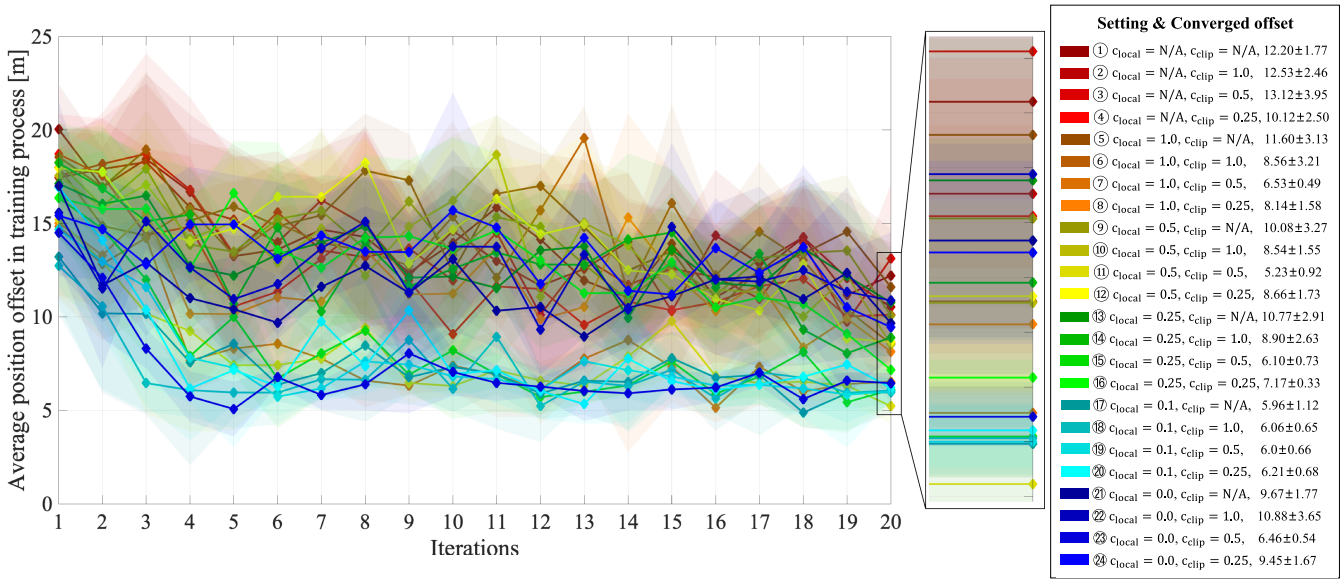


Fig. 4. Learning curves of the RL training procedure in LUSPMPC with different configurations of parameters.

TABLE II
TEST RESULTS OF LUSPMPC WITH DIFFERENT CONFIGURATIONS OF PARAMETERS

Approach	Parameters	Average [m]	Median [m]	Success rate (final)	Success rate (overall)
SPMPC (SSGP)	① $c_{\text{local}} = \text{N/A}, c_{\text{clip}} = \text{N/A}$	15.93 ± 12.16	12.41	24.67%	2.67%
	② $c_{\text{local}} = \text{N/A}, c_{\text{clip}} = 1.0$	10.21 ± 8.93	7.38	51.33%	4.67%
	③ $c_{\text{local}} = \text{N/A}, c_{\text{clip}} = 0.5$	11.61 ± 11.04	7.43	41.33%	4.0%
	④ $c_{\text{local}} = \text{N/A}, c_{\text{clip}} = 0.25$	15.78 ± 14.52	10.91	34.67%	1.33%
LUSPMPC	⑤ $c_{\text{local}} = 1.0, c_{\text{clip}} = \text{N/A}$	13.12 ± 10.58	10.22	42.0%	4.67%
	⑥ $c_{\text{local}} = 1.0, c_{\text{clip}} = 1.0$	11.12 ± 8.78	8.40	38.0%	10.0%
	⑦ $c_{\text{local}} = 1.0, c_{\text{clip}} = 0.5$	5.34 ± 5.77	4.72	92.0%	32.0%
	⑧ $c_{\text{local}} = 1.0, c_{\text{clip}} = 0.25$	6.62 ± 7.16	4.99	74.67%	25.33%
	⑨ $c_{\text{local}} = 0.5, c_{\text{clip}} = \text{N/A}$	11.39 ± 8.94	8.80	44.0%	4.67%
	⑩ $c_{\text{local}} = 0.5, c_{\text{clip}} = 1.0$	8.74 ± 6.91	7.16	47.33%	16.0%
	⑪ $c_{\text{local}} = 0.5, c_{\text{clip}} = 0.5$	6.16 ± 9.27	4.32	90.0%	46.67%
	⑫ $c_{\text{local}} = 0.5, c_{\text{clip}} = 0.25$	6.69 ± 8.54	5.08	74.67%	36.67%
	⑬ $c_{\text{local}} = 0.25, c_{\text{clip}} = \text{N/A}$	6.96 ± 4.98	6.24	61.33%	22.67%
	⑭ $c_{\text{local}} = 0.25, c_{\text{clip}} = 1.0$	5.10 ± 3.21	5.05	83.33%	30.0%
	⑮ $c_{\text{local}} = 0.25, c_{\text{clip}} = 0.5$	3.78 ± 3.35	**3.72**	96.67%	62.67%
	⑯ $c_{\text{local}} = 0.25, c_{\text{clip}} = 0.25$	4.44 ± 2.80	4.56	90.0%	38.67%
	⑰ $c_{\text{local}} = 0.1, c_{\text{clip}} = \text{N/A}$	4.43 ± 5.48	4.69	90.0%	43.33%
	⑱ $c_{\text{local}} = 0.1, c_{\text{clip}} = 1.0$	4.92 ± 2.56	4.48	94.67%	49.33%
	⑲ $c_{\text{local}} = 0.1, c_{\text{clip}} = 0.5$	$\mathbf{3.64 \pm 2.17}$	3.80	**98.67%**	**67.33%**
	⑳ $c_{\text{local}} = 0.1, c_{\text{clip}} = 0.25$	3.65 ± 2.14	3.82	98.0%	62.67%
	㉑ $c_{\text{local}} = 0.0, c_{\text{clip}} = \text{N/A}$	7.87 ± 9.94	6.94	55.33%	8.0%
	㉒ $c_{\text{local}} = 0.0, c_{\text{clip}} = 1.0$	6.93 ± 4.47	6.52	70.0%	22.67%
	㉓ $c_{\text{local}} = 0.0, c_{\text{clip}} = 0.5$	4.90 ± 4.36	4.59	85.33%	46.0%
	㉔ $c_{\text{local}} = 0.0, c_{\text{clip}} = 0.25$	6.68 ± 6.41	5.69	76.0%	30.67%

The bold entities represent the best performances.

also had large standard deviations in offset, which indicated the poor generalization capability in the test procedure with stochastically initialized and continuously changing environmental disturbances. According to the results of configurations 5, 9, 13, 17, and 21 given in Table II, LUSPMPC with only local update achieved the best performance with configuration 17 $c_{\text{local}} = 0.1$. Compared with configuration 1, it significantly reduced 72.19% average offset, 54.93% standard deviations of offset, and 62.21% median offset, and improved the final success rate from 24.67% to 90.0% and the overall success rate from 2.67% to 43.33%. Based on Fig. 5, it also achieved better model quality by reducing 83.33% predicted error in the *X*-axis position, 68.42% in the *Y*-axis position, 73.91% in the velocity, and 58.84% in the orientation. This result indicated the advantages of the local update in LUSPMPC compared with the original SSGP (configuration 1) in control performances, model quality, and generalization capability.

The results of configurations 18, 19, and 20 demonstrated better control performance and generalization capability achieved by both local updating and properly clipping the frequency.

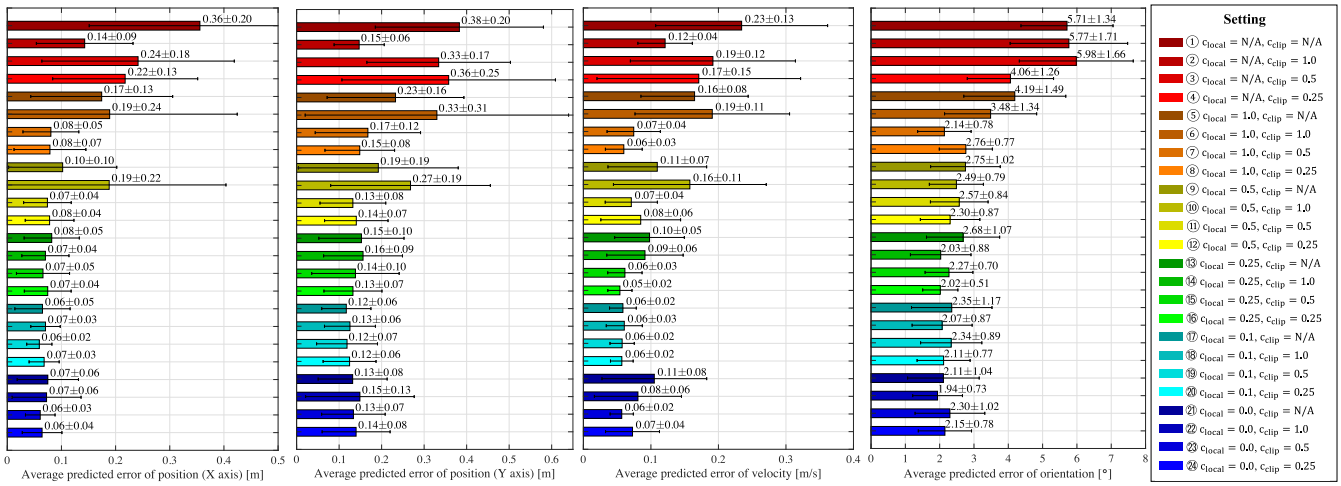


Fig. 5. Average model prediction errors of the GP models learned in LUSPMPC with configurations of parameters.

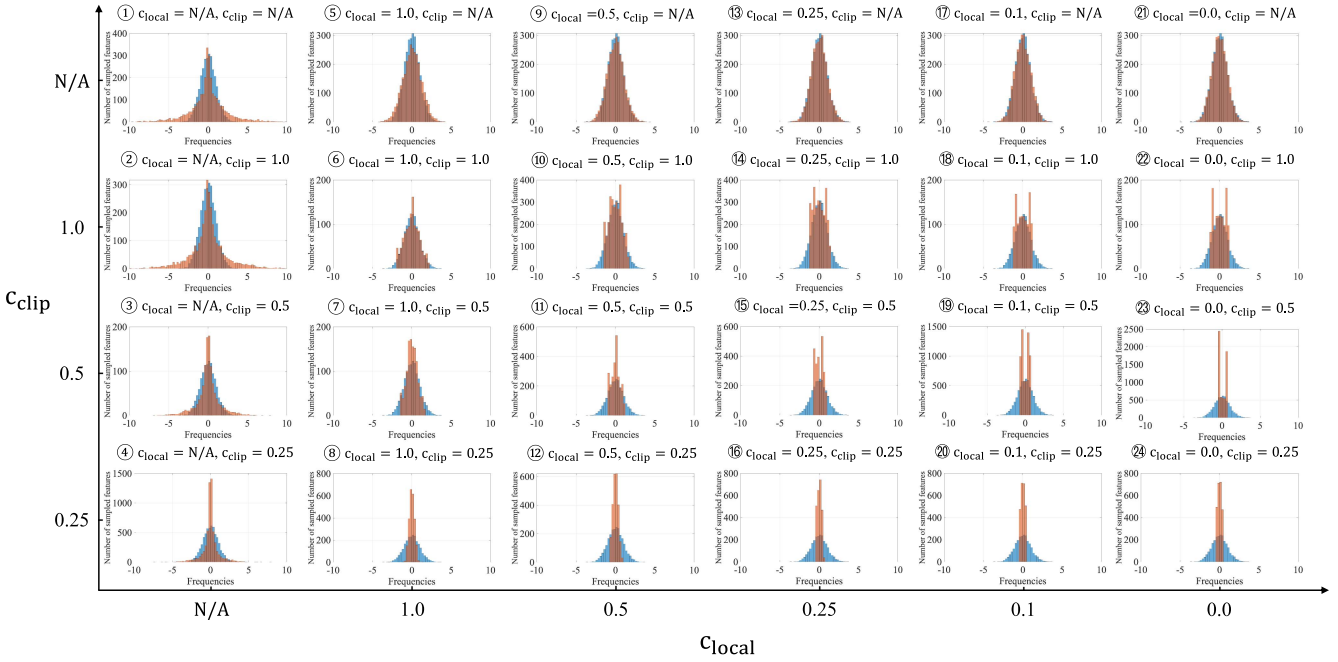


Fig. 6. Distribution of the frequency samples after training in LUSPMPC with different configurations of parameters. The blue and red bars represent the frequency samples initialized following the normal distribution and optimized after training.

Setting $c_{\text{local}} = 0.1, c_{\text{clip}} = 0.5$ (configuration 19), LUSPMPC achieved the overall best performance in Table II: it reduced 77.15% and 82.15% in the average and standard deviations of offset, 69.38% in the median offset, and improved 74.0% in the final success rate and 64.66% in the overall success rate while maintaining the high model quality compared with the original SSGP (configuration 1).

The effect of different c_{local} with a fixed $c_{\text{clip}} = 0.5$ was then studied in configurations 7, 11, 15, 19, and 23. Configuration 23 prevented the optimization of frequency samples by setting $c_{\text{local}} = 0$. Although it had a close converge performance compared with configuration 19 in the training procedure (see Fig. 4), it resulted in a low generalization capability (about 100% increased standard deviation in offset) with overall degraded control performances due to the limited flexibility of adapting the approximated kernel function to the given samples, which is suggested by the related works [25], [26]. On the other hand, releasing the constraints c_{local} in configurations 7 and 11 increased the predicted errors of model in position, orientation, and velocity due to the overlarge update of the frequency samples, which turned to more capriciously changing approximated kernel functions during the training procedure, as shown in Fig. 3. These deteriorating models turned to worse performances and generalization capability in Table II compared with configuration 19 but still outperformed SPMPC (SSGP) in configurations 1–4.

We finally summarized the results in this section based on the distribution of the frequency samples $\boldsymbol{\omega}$ over all configurations analyzed in Fig. 6 where the blue/red bars represent the frequency samples initialized following the normal distribution/

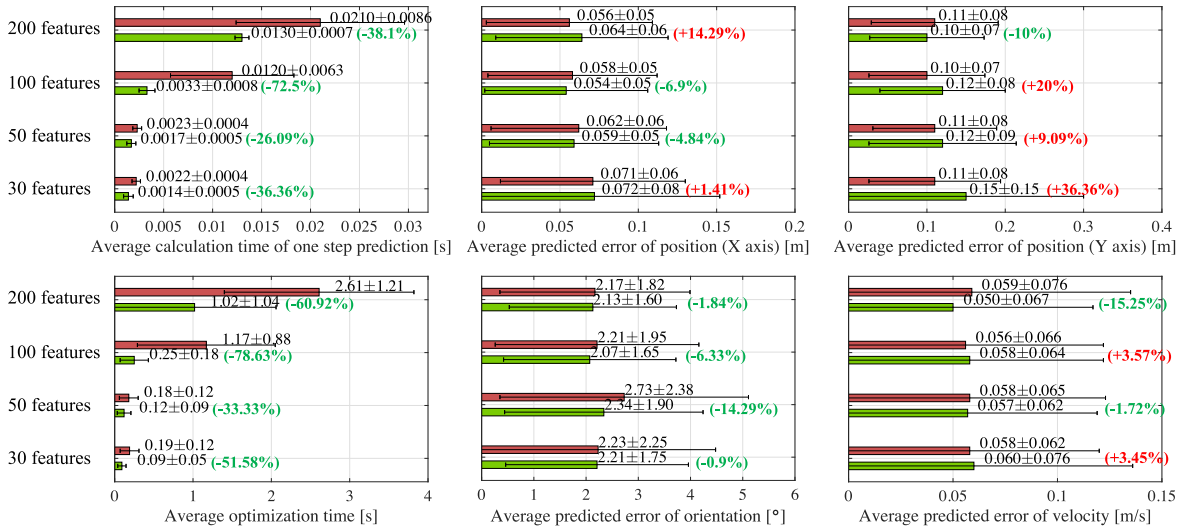


Fig. 7. Average calculation time and model prediction errors of the GP models learned in SPMPC (sparse GP) and LUSPMPC with different sparse scales. The red and green bars represent the result of SPMPC (sparse GP) and LUSPMPC.

optimized after training. All distributions were calculated over all input dimensions regularized by their own $\tilde{\Lambda}_a$. In LUSPMPC with $c_{\text{clip}} = \text{N/A}$ (configurations 5, 9, 13, 17, and 21), the local update contributed to a stable distribution close to the initial one after the RL training procedure since the change of frequency in optimization is limited. We believed it is the key to the advantages of learning capability, generalization capability, control performance, and model quality. As a comparison, the freely optimized frequency samples distribution in SPMPC (SSGP) was very different from the initial one according to the result of configuration 1. Although the frequency samples were initialized and clipped within the low-frequency domain in configurations 2–4, they covered the high-frequency domain after optimizations without the local update. These distributions turned to the hugely biased approximated kernel functions with poor control performances and low model quality. According to the results of configurations 18, 19, and 20, properly clipping the initial frequency samples based on the local update contributed to better control performance and generalization capability than configuration 17. (The same results can be observed in all configurations with the local update following the axis of c_{clip}.) Although Bochner's theory was damaged due to the biased distribution after clipping, it resulted in an approximated kernel function focusing on the low-frequency domain whose engineering value of improving the performance of MBRL without losing model accuracy was experimentally evaluated in this section.

C. Evaluation of Model Quality and Control Performance

In this section, the advantages of LUSPMPC over the baseline MBRL approach specified to USV, SPMPC (sparse GP) [22] that employed sparse GP [23] over sample space rather than SSGP [24] were evaluated. We set $c_{\text{local}} = 0.1, c_{\text{clip}} = 0.5$ in LUSPMPC according to the superior performance of configuration 19 on average offset, media offset, success rates, and predicted errors over other configurations compared in Table II and Fig. 5, Section IV-B. Both two approaches with different numbers of sparse samples, i.e., the number of frequency samples and pseudoinputs, were trained following the setting in Section IV-A. Compared with SPMPC, LUSPMPC reached the same feature size of the approximated kernel with half sparse samples since m-frequency samples are translated to a $2m$ feature vector via trigonometric functions following (9). We fairly compared these two approaches with the same feature size.

Table III and Fig. 7 demonstrated the control performances and model quality in the test procedure where the numbers in red and green presented the increase/decrease of each term in LUSPMPC compared with SPMPC using sparse GP with the same sparse scale. With a limited feature size 30, LUSPMPC learned a model with increased predicted errors in position (1.41% and 36.36% in X- and Y-axes, respectively), and velocity (3.45%) compared with SPMPC using Sparse GP. It turned to deteriorated performances in the average offset and the final success rate. With an increased feature size from 50 to 200, LUSPMPC stably outperformed the baseline approach in the control performances with close model qualities. It reduced about 2.67%–8.65% average offset and 2.56%–14.51% median offset while improving 1.34%–2.0% final success rate and 4.0%–28.0% overall success rate. From the one-step prediction time defined as the time of calculating the long-term reward with H steps' model prediction, and the optimization time defined as the time of the whole MPC procedure in one step shown in Fig. 7, we observed the significant advantages of LUSPMPC in computational complexity besides its superior control performances. It reduced about 26.09%–72.5% average one-step prediction time and 33.33%–78.63% average optimization time. Although the moment matching with Sparse GP and SSGP had the same order of computational complexity according to Deisenroth et al. [19] and Pan et al. [26], LUSPMPC enjoyed better computational efficiency than SPMPC using Sparse GP in the engineering implementation: the heavy calculated terms

TABLE III
TEST RESULT OF LUSPMPC COMPARED WITH SPMPC USING SPARSE GP

Approach	Sparse samples	Feature size	Average [m]	Median [m]	Success rate (final)	Success rate (overall)
LUSPMPC	15	30	4.1 ± 4.33 (+0.49%)	3.87 (−9.79%)	96.67% (−0.66%)	58.67% (+5.34%)
	25	50	3.64 ± 2.17 (−2.67%)	3.8 (−2.56%)	**98.67%** (+1.34%)	**67.33%** (+10.0%)
	50	100	$\mathbf{3.59 \pm 2.16}$ (−8.65%)	**3.74** (−9.66%)	96.97% (+1.64%)	61.33% (+4.0%)
	100	200	3.98 ± 3.65 (−7.87%)	3.89 (−14.51%)	97.33% (+2.0%)	58.67% (+28.0%)
SPMPC (sparse GP)	30	30	4.08 ± 2.29	4.29	97.33%	53.33%
	50	50	3.74 ± 2.26	3.9	97.33%	59.33%
	100	100	3.93 ± 2.52	4.14	95.33%	57.33%
	200	200	4.32 ± 2.58	4.55	95.33%	30.67%

The bold entities represent the best performances.

$\boldsymbol{\alpha}_a$, $\boldsymbol{A}_a^{-1}$, and $\boldsymbol{\Psi}_a$ in (14) and (15) are irrelevant to the input $\tilde{\boldsymbol{x}}_*$ and, therefore, could be computed in advanced.

In summary, with proper feature size, the proposed LUSPMPC achieved less offset and higher success rates in the USV position-keeping task while dramatically reducing the computational cost (from 33.33% to 78.63%) compared with the traditional work using Sparse GP [22]. Its superiority in control performance and computational efficiency contributes to a wide range of MBRL applications in the real-world USV where the computational resource is usually limited.

V. CONCLUSION

In this work, we proposed LUSPMPC for the better computational efficiency of MBRL in USV control problem. LUSPMPC employed a local update strategy to stably learn the GP approximation in the frequency domain from the iteratively generated data set. The approximated GP was further encouraged to focus on the low-frequency features by a frequency clip trick. Evaluated by the comprehensive comparisons of different configurations in the position-keeping task of a real USV data-driven simulation, LUSPMPC demonstrated its advantages in learning capability, model quality, and generalization capability over traditional SSGP [24] and existing MBRL USV approach using the sparse GP [22]. Compared with SSGP, it converged to 46% fewer average position offset in the training while learning a more accurate probabilistic model of USV. It achieved 77.2% less average position offset and 74.0% higher success rate in the testing with superior generalization capability. With 100 sparse features, it significantly outperformed the SPMPC using the sparse GP with 8.7% fewer average position offset and over 70% less computational burden and, therefore, expanded the potential of probabilistic MBRL in USV systems with limited computational resources.

Our future work can be divided into two parts. On the algorithmic side, extending LUSPMPC to partially observed MDP where the unobservable disturbances are handled by hidden states instead of Gaussian noises could contribute to a superior performance of MBRL under strong disturbances following [34], [35]. On the side of engineering, designing sophisticated reward functions for specific tasks (e.g., consider USV orientation, velocity, rudder, and throttle as reward terms) and improving the optimization in MPC via predicted uncertainties reward function encourages [36], [37] are necessary to implement the proposed method to more challenging USV tasks.

REFERENCES

[1] "United nations conference on trade and development, review of maritime transport," 2018. Accessed: May 15, 2023. [Online]. Available: https://unctad.org/system/files/official-document/rmt2018_en.pdf

[2] N. Wang, S. Su, X. Pan, X. Yu, and G. Xie, "Yaw-guided trajectory tracking control of an asymmetric underactuated surface vehicle," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3502–3513, Jun. 2019.

[3] B.-O. H. Eriksen, M. Breivik, E. F. Wilthil, A. L. Flåten, and E. F. Brekke, "The branching-course model predictive control algorithm for maritime collision avoidance," *J. Field Robot.*, vol. 36, no. 7, pp. 1222–1249, 2019.

[4] N. Wang and H. R. Karimi, "Successive waypoints tracking of an underactuated surface vehicle," *IEEE Trans. Ind. Informat.*, vol. 16, no. 2, pp. 898–908, Feb. 2020.

[5] Y. Jiang, Z. Peng, D. Wang, and C. L. P. Chen, "Line-of-sight target enclosing of an underactuated autonomous surface vehicle with experiment results," *IEEE Trans. Ind. Informat.*, vol. 16, no. 2, pp. 832–841, Feb. 2020.

[6] N. Wang and H. He, "Extreme learning-based monocular visual servo of an unmanned surface vessel," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5152–5163, Aug. 2021.

[7] Z. Liu, Y. Zhang, C. Yuan, and J. Luo, "Adaptive path following control of unmanned surface vehicles considering environmental disturbances and system constraints," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 51, no. 1, pp. 339–353, Jan. 2021.

[8] N. Wang, Y. Zhang, C. K. Ahn, and Q. Xu, "Autonomous pilot of unmanned surface vehicles: Bridging path planning and tracking," *IEEE Trans. Veh. Technol.*, vol. 71, no. 3, pp. 2358–2374, Mar. 2022.

[9] R. Skulstad, G. Li, T. I. Fossen, B. Vik, and H. Zhang, "Dead reckoning of dynamically positioned ships: Using an efficient recurrent neural network," *IEEE Robot. Automat. Mag.*, vol. 26, no. 3, pp. 39–51, Sep. 2019.

[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[11] L. Zhao and M.-I. Roh, "Colregs-compliant multiship collision avoidance based on deep reinforcement learning," *Ocean Eng.*, vol. 191, 2019, Art. no. 106436.

[12] J. Woo, C. Yu, and N. Kim, "Deep reinforcement learning-based controller for path following of an unmanned surface vehicle," *Ocean Eng.*, vol. 183, pp. 155–166, 2019.

[13] Y. Zhao, X. Qi, Y. Ma, Z. Li, R. Malekian, and M. A. Sotelo, "Path following optimization for an underactuated USV using smoothly-convergent deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 10, pp. 6208–6220, Oct. 2021.

[14] N. Wang, Y. Gao, and X. Zhang, "Data-driven performance-prescribed reinforcement learning control of an unmanned surface vehicle," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5456–5467, Dec. 2021.

[15] Y. Zhao, Y. Ma, and S. Hu, "USV formation and path-following control via deep reinforcement learning with random braking," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5468–5478, Dec. 2021.

[16] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[17] D. Sarkar, M. A. Osborne, and T. A. Adcock, "Prediction of tidal currents using Bayesian machine learning," *Ocean Eng.*, vol. 158, pp. 221–231, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801818302397

[18] G. A. Hollinger, A. A. Pereira, J. Binney, T. Somers, and G. S. Sukhatme, "Learning uncertainty in ocean current predictions for safe and reliable navigation of underwater vehicles," *J. Field Robot.*, vol. 33, no. 1, pp. 47–66, 2016.

[19] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 2, pp. 408–423, Feb. 2015.

[20] A. Girard, C. E. Rasmussen, J. Q. Candela, and R. Murray-Smith, "Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2003, pp. 545–552.
[21] S. Kamthe and M. Deisenroth, "Data-efficient reinforcement learning with probabilistic model predictive control," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2018, pp. 1701–1710.
[22] Y. Cui, S. Osaki, and T. Matsubara, "Autonomous boat driving system using sample-efficient model predictive control-based reinforcement learning approach," *J. Field Robot.*, vol. 38, no. 3, pp. 331–354, 2021.
[23] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 1257–1264.
[24] A. Rahimi et al., "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 1177–1184.
[25] M. Lázaro-Gredilla, J. Quinonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal, "Sparse spectrum Gaussian process regression," *J. Mach. Learn. Res.*, vol. 11, pp. 1865–1881, 2010.
[26] Y. Pan, X. Yan, E. A. Theodorou, and B. Boots, "Prediction under uncertainty in sparse spectrum Gaussian processes with applications to filtering and control," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2760–2768.
[27] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4759–4770.
[28] J. Vinogradska, B. Bischoff, D. Nguyen-Tuong, and J. Peters, "Stability of controllers for Gaussian process dynamics," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 3483–3519, 2017.
[29] C. K. Williams, "Prediction with Gaussian processes: From linear regression to linear prediction and beyond," in *Learning in Graphical Models*. Berlin, Germany: Springer, 1998, pp. 599–621.
[30] F. Tobar, "Band-limited Gaussian processes: The sinc kernel," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.
[31] C. Valenzuela and F. Tobar, "Low-pass filtering as Bayesian inference," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2019, pp. 3367–3371.
[32] A. G. D. G. Matthews et al., "GPflow: A Gaussian process library using tensorflow," *J. Mach. Learn. Res.*, vol. 18, no. 40, pp. 1–6, 2017.
[33] M. J. Powell, "The BOBYQA algorithm for bound constrained optimization without derivatives,", University of Cambridge, Cambridge, U.K., Rep. DAMTP 2009/NA06, 2009.
[34] R. McAllister and C. E. Rasmussen, "Data-efficient reinforcement learning in continuous state-action Gaussian-POMDPs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2040–2049.
[35] Y. Cui, L. Peng, and H. Li, "Filtered probabilistic model predictive control-based reinforcement learning for unmanned surface vehicles," *IEEE Trans. Ind. Informat.*, vol. 18, no. 10, pp. 6950–6961, Oct. 2022.
[36] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Robust constrained learning-based NMPC enabling reliable mobile robot path tracking," *Int. J. Robot. Res.*, vol. 35, no. 13, pp. 1547–1563, 2016.
[37] C.-Y. Kuo, A. Schaarschmidt, Y. Cui, T. Asfour, and T. Matsubara, "Uncertainty-aware contact-safe model-based reinforcement learning," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 3918–3925, Apr. 2021.

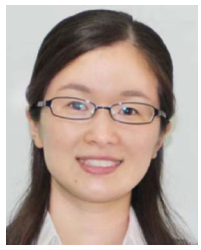
Yunduan Cui (Member, IEEE) received the B.E. degree in electronic engineering from Xidian University, Xi'an, China, in 2012, the M.E. degree in computer science from Doshisha University, Kyoto, Japan, in 2014, and the Ph.D. degree in computer science from the Nara Institute of Science and Technology, Ikoma, Japan, in 2017.

From 2017 to 2020, he was a Postdoc Researcher with the Nara Institute of Science and Technology. He is currently an Associate Professor with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Beijing, China. His research focuses on machine learning, especially reinforcement learning-driven unmanned systems.

Dr. Cui was the recipient of the Best Oral Paper Award of IEEE-RAS 16th International Conference on Humanoid Robots in 2016, Best Student Award of Nara Institute of Science and Technology in 2018, Best Paper Award of Japanese Neural Network Society (JNNS) in 2018, and the SICE International Young Authors Award for IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) in 2019.

Wenbo Shi received the B.S. degree in metallic materials engineering from the Harbin University of Science and Technology, Harbin, China, in 2018, and the M.S. degree in computer science and technology from Qingdao University, Qingdao, China, in 2022.

He is currently an Algorithm Engineer with Baidu Online Network Technology, Company, Ltd., Beijing, China. His research interests mainly include computer vision, medical image processing, and probabilistic model-based reinforcement learning.



Huan Yang (Member, IEEE) received the B.S. degree in computer science from the Heilongjiang Institute of Technology, Harbin, China, in 2007, the M.S. degree in computer science from Shandong University, Jinan, China, in 2010, and the Ph.D. degree in computer engineering from Nanyang Technological University, Singapore, in 2015.

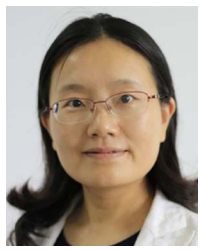
She is currently an Associate Professor with the College of Computer Science and Technology, Qingdao University, Qingdao, China. Her research interests include intelligent image processing, perception-based quality assessment, and computer vision.

Cuiping Shao received the B.S. degree in electronic science and technology from the Xi'an University of Technology, Xi'an, China, in 2009, the M.S. degree in microelectronics from Xi'an Microelectronics Technology Research Institute, Xi'an, China, in 2012, and the Ph.D. degree in computer science from the University of Chinese Academy of Sciences, Beijing, China, in 2019.

She is currently an Associate Professor with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Beijing. Her research interests include VLSI design and test, hardware security, and data fault tolerance.

Lei Peng (Member, IEEE) received the M.S. and Ph.D. degrees in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2003 and 2009, respectively.

He is currently an Associate Research Fellow with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Beijing, China. His research interests include ITS, IoV, and Big Data mining.



Huiyun Li (Senior Member, IEEE) received the M.Eng. degree in electronic engineering from Nanyang Technological University, Singapore, in 2001, and the Ph.D. degree in computer science from the University of Cambridge, Cambridge, U.K., in 2006.

She is currently a Professor with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Beijing, China, and a Professor with the Chinese University of Hong Kong, Hong Kong. Her research interests include automobile electronics and autonomous vehicles.