Autonomous boat driving system using sample-efficient model predictive control-based reinforcement learning approach

Yunduan Cui¹,²,³ | Shigeki Osaki⁴ | Takamitsu Matsubara¹

¹Division of Information Science, Graduate School of Science and Technology, Nara Institute of Science and Technology, Nara, Japan
²Center for Automotive Electronics, Shenzhen Institute of Advanced Technology (SIAT), Chinese Academy of Sciences, Shenzhen, China
³SIAT Branch, Shenzhen Institute of Artificial Intelligence and Robotics for Society, Shenzhen, China
⁴Furuno Electric Co., Ltd., Nishinomiya City, Japan

Correspondence
Yunduan Cui, 1068 Xueyuan Avenue, Shenzhen University Town, Shenzhen 518055, China.
Email: cuiyunduan@gmail.com

Abstract
In this article, we propose a novel reinforcement learning (RL) approach specialized for autonomous boats: sample-efficient probabilistic model predictive control (SPMPC), to iteratively learn control policies of boats in real ocean environments without human prior knowledge. SPMPC addresses difficulties arising from large uncertainties in this challenging application and the need for rapid adaptation to dynamic environmental conditions, and the extremely high cost of exploring and sampling with a real vessel. SPMPC combines a Gaussian process model and model predictive control under a model-based RL framework to iteratively model and quickly respond to uncertain ocean environments while maintaining sample efficiency. A SPMPC system is developed with features including quadrant-based action search rule, bias compensation, and parallel computing that contribute to better control capabilities. It successfully learns to control a full-sized single-engine boat equipped with sensors measuring GPS position, speed, direction, and wind, in a real-world position holding task without models from human demonstration.

KEYWORDS
learning, marine robotics

1 | INTRODUCTION

To help address a shortage (United Nations Conference on Trade and Development, 2018) of skilled professionals in the growing marine shipping industry, there have been significant efforts towards the rapid development of autonomous surface ships, or unmanned surface vehicles (USV). These are attracting increasing attention in both industry projects (Kongsberg Maritime, 2018; Rolls-Royce, 2016) and academic research because of their potential to increase safety and efficiency, while also reducing costs and environmental impact by removing the human element from certain steps in the shipping process. A wide range of works have been conducted including collision avoidance (Eriksen, Breivik, Wilthil, Flåten, & Brekke, 2019), navigation (Elkins, Sellers, & Monach, 2010), station-staying control (Sarda, Qu, Bertaska, & vonEllenrieder, 2016), and marine sciences research (Dunbabin & Grinham, 2017). Varieties of approaches have been implemented in autonomous USV control including proportional-integral-derivative (PID) controller in dynamic positioning systems with a model vessel (Nguyen, Sørensen, & Quek, 2007) and collision avoidance with a catamaran-shaped research vessel (Naeem, Irwin, & Yang, 2012), linear quadratic controller in tracking system using a model vessel (Lefeber, Pettersen, & Nijmeijer, 2003), model predictive control (MPC) in collision avoidance using the Telemetron ASV (Eriksen et al., 2019; Hagen, Kufaloar, Brekke, & Johansen, 2018), an autopilot system using a twin hull vessel (Annamalai, Sutton, Yang, Culverhouse, & Sharma, 2015), and neural networks for formation control (Peng, Wang, Chen, Hu, & Lan, 2013) and position estimation (Skulstad, Li, Fossen, Vik, & Zhang, 2019) using USV simulations.
However, fully autonomous USV remain a more distant ambition since human intervention is still often necessary in real industrial maritime operations (United Nations Conference on Trade and Development, 2018). For example, the recent work (Eriksen et al., 2019) had excellent results in real USV collision avoidance by combining MPC with other traditional control methods. It requires a dynamical model based on human knowledge to predict USV behaviors. Furthermore, the parameters of the controller were heuristically selected for good performance. Another recent work (Skulstad et al., 2019) learned a controller for autonomous ship driving by neural networks which requires preprepared samples for supervised learning. Its implementation was therefore limited to simulation due to the expensive cost of collecting real training data.

As an integral part of contemporary machine learning, reinforcement learning (RL; Sutton & Barto, 1998) enables agents to learn an optimal or suboptimal control policies from unknown environments via trial-and-error interactions (Kober, Bagnell, & Peters, 2013), therefore presenting itself as an appealing prospect for a fully autonomous USV. A RL-based autonomous USV approach could iteratively learn control policies adaptive to different environments without prior human knowledge nor heuristic parameter tuning. Although RL has been previously explored in both autonomous ground (Vincent & Sun, 2012; Williams, Drews, Goldfain, Rehg, & Theodorou, 2018) and air (Kim, Jordan, Sastry, & Ng, 2004; Tran et al., 2015) vehicles, its application to USV remains relatively limited (Liu, Zhang, Yu, & Yuan, 2016). Some recent works tried to implement state-of-the-art deep RL algorithms to USV in path following and collision avoidance (Zhao, Roh, & Lee, 2019; Meyer, Robinson, Rasheed, & San, 2020) but were limited in simulation tasks. The lack of RL towards real USV control is possibly driven by:

1. Difficulty of predicting the uncertainties in a dynamic ocean environment, for example, frequent disturbances due to unpredictable wind and ocean current, signal noise.
2. Difficulty of quickly providing suitable control signals under such a rapidly changing and often unpredictable environment.
3. A very high sampling cost when using real USVs for data collection.

The next section summarises several existing RL studies in mitigating these difficulties.

1.1 Related works

As a potential solution to naturally model system dynamics with uncertainties, Gaussian processes (Rasmussen & Williams, 2006) represents them as Gaussian distributed random variables. Cao, Lai, and Alam (2017) proposed a controller combining a GP model with MPC for an online optimization framework suited towards quickly changing situations to handle both the prediction of uncertainties and rapid responses to changing environmental conditions. Unlike a RL based approach, it learned the GP model from precollected training data without exploration, and utilized a robust MPC controller to successfully control an unmanned quadrotor in simulation while considering state uncertainties.

Since model-based RL (Polydoros & Nalpantidis, 2017) learns policies from a trained model instead of the environment for better sample efficiency (Ghavamzadeh, Engel, & Valko, 2016) proposed both GP model based actor-critic and policy gradient algorithms and investigated them in a simplified boat steering task in simulation. A GP-based temporal difference RL approach was proposed in John, Jinkun, and Brendan (2018) and tested in an autonomous submersible navigation task in an indoor pool. PILCO (Deisenroth, Fox, & Rasmussen, 2013), a state-of-the-art model-based RL method reduces model bias by explicitly incorporating GP model uncertainty into planning and control. Assuming target dynamics are fully controllable, it learns an optimal policy by long-term planning from the initial state. However, applying PILCO to USV is difficult due to unforeseeable disturbances such as wind and current. A proper feedback control against these disturbances by re-planning is computationally demanding since a large number of parameters in a state-feedback policy are optimized, while ignoring them in the long-term planning may result in poor control performance due to accumulated model error.

Williams et al. (2018) introduced MPC into model-based RL and successfully implemented it in driving autonomous ground vehicles. This study approximates the dynamics model by means of neural networks, whose formulation makes it difficult to follow the fully Bayesian formalism for naturally considering state uncertainties. It also would require a large number of samples for model learning and hyper-parameter tuning.

As the first attempt to combine the respective benefits of GP models, MPC and model-based RL, (Kamthe & Deisenroth, 2018) extended PILCO to avoid full-horizon planning in model-based RL by introducing MPC to moderate the real-time disturbances within a closed control loop. This study successfully showed its sample efficiency in simulated cart-pole and double pendulum tasks without considering external disturbances. One possible limitation of applying this study towards challenging real-world control problems is the relatively heavy computational cost, since its optimization is executed with a dimensionality that has been expanded for a deterministic dynamical system with Lagrange parameters and state constraints under Pontryagin’s maximum principle (PMP). However, in autonomous boats where state constraints are less important, a simpler and more computationally efficient method may be feasible.

From these works, the combination of GP, MPC, and model-based RL is a potentially suitable solution towards a sample efficient learning in unpredictable environments. However, the corresponding application in real-world challenging tasks remains limited due to the gap between theory and real word implementation, for example, the computational cost, the hardware delay and so on. The motivation of this study is to fill this gap by developing a MPC and GP model-based RL approach specialized for autonomous boats in real ocean environment.

1.2 Contribution

In this article, we present a novel RL approach specialized for autonomous boats: sample-efficient probabilistic model predictive
control (SPMPC). Enjoying the sample efficiency of model-based RL, SPMPC iteratively learns a GP model of boat dynamics to increase the robustness of control against unpredictable and frequently changing noises and disturbances. Furthermore, it efficiently optimizes control signals under a close-loop MPC to reduce the heavy computation cost of the full-horizon planning in Deisenroth et al. (2013). Unlike the method in Kamthe and Deisenroth (2018), SPMPC directly optimizes the long-term cost with neither the expanded dynamics nor state constraints by separating the uncertain state and deterministic control signal during prediction for computational efficiency.

An instrumented autonomous boat driving system was then built, consisting of a full-sized boat equipped with a single engine and sensors for GPS position, speed, direction, and wind (Figure 1). Several features were proposed including quadrant-based action search rule, bias compensation, and parallel computing to improve the boat’s control capabilities. The proposed system was then evaluated in a real-world position holding task. Experimental results show the capability of the proposed system in terms of both robustness to disturbances and sample efficiency. To complement the real experimental results, several simulation experiments were also conducted to investigate SPMPC’s learning behaviors and control performances under more challenging conditions.

Our preliminary work was published as a conference paper in Cui, Osaki, and Matsubara, 2019. This article builds upon the preliminary work as follows:

1. Implementation of a parallel computation based communication node to alleviate control delay.
2. Addition of domain knowledge to limit the search range of optimization for improved control capability.
3. Extending the state with engine speed and rudder angle to reduce the effect of the delay between the control signal and hardware.
4. Extending the real-world experiment to a position holding task based on the target reaching task in Cui et al. (2019), and analyzing the results with more detail.

1.3 Outline

The remainder of this paper is organized as follows. Section 2 details the algorithm of SPMPC. Section 3 describes the SPMPC-based autonomous boat driving system. The real boat experiment in a position holding task is presented in Section 4. Several simulation experiments are conducted in Section 5 to investigate the RL learning behaviors, model accuracy, the effects of changing algorithm settings, and extension towards a position reaching and holding task. Finally, discussions and conclusion follow in Sections 6 and 7.

2 | APPROACH

In this section, the algorithm of SPMPC is detailed. As a model-based RL approach, SPMPC stores the knowledge of the environment in a learned GP model (Section 2.1). Figure 2 shows an example of SPMPC driving a boat. At step t after observing the current state, SPMPC can predict future states (e.g., boat velocity and direction, shown in opaque gray) with uncertainties caused by disturbances such as wind and current using the modified moment-matching approach introduced in Section 2.2. Given a long-term cost function for example, the squared Euclidean distance to the target (red cross), SPMPC then optimizes a control sequence to minimize the cost function, and employs an MPC framework as a quick feedback controller against the dynamical ocean environment which features unpredictable and unobservable disturbances (Section 2.3). By repeating this process at each step, SPMPC can control the boat to minimize the task cost while considering both the uncertainties and frequently changing dynamics in the challenging ocean environment. We finally introduce how to run SPMPC in a RL process in Section 2.4.

2.1 | Gaussian process (GP) model

The GP is a collection of random variables, any finite number of which have a joint Gaussian distribution. It is widely used as a non-parametric regression model (Rasmussen & Williams, 2006). Consider a stochastic dynamical system:

\[
x_{t+1} = f(x_t, u_t) + w_t
\]

where the unknown latent transition function is defined as \( f, x \in \mathbb{R}^D \) is the \( D \)-dimensional state, \( u \in \mathbb{R}^n \) is the \( n \)-dimensional action, \( w \sim N(0, \Sigma_w) \) is the system noise, and \( t \) is the time step. Given the training input \( \tilde{x}_t := (x_t, u_t) \) and the training target \( y_t := x_{t+1} \). The GP model is represented as \( y_t = f(w_t) \). Consider a multidimensional state \( x \in \mathbb{R}^a \), for each target dimensions \( a = 1, ..., D \), one GP regression model of the latent function \( y_t^a = f_t(x_t) + w_t \) is fully specified by its mean function \( m_t^a(.) \) and a covariance squared exponential (SE) kernel function:

\[
k_x(x_i, x_j) = \alpha_k^2 \exp\left(-\frac{1}{2} \frac{(x_i - x_j)^T \Lambda^{-1} (x_i - x_j)}{\theta^2}ight)
\]

with two hyper-parameters: \( \alpha_k^2 \) is the overall variance of \( f_t \) and \( \Lambda \) is the diagonal matrix of squared characteristic length-scales for each training inputs \( \Lambda = \text{diag}(\ell_1^2, ..., \ell_D^2) \). Note that in the dimension of \( x_{t+1} \) can be smaller than \( x_t \), here we simplify the equations by assuming they have same dimensionality \( D \). Given the training input \( \tilde{x}_t := (x_t, u_t) \), training target \( y_t := x_{t+1} \), for each target dimensions \( a = 1, ..., D \), and \( m_b^a(\tilde{x}_t) = 0 \) as a prior, the hyper-parameters are commonly learned via evidence maximization (MacKay, 2003; Rasmussen & Williams, 2006). After learning the hyper-parameters, given a new input \( \tilde{x}_n \), the mean and variance of the GP’s posterior becomes:
where \( k_{aa} = k_a(\tilde{x}_a, \tilde{x}_a), k_{sa} = k_a(\tilde{x}_a, \tilde{x}_n), k_{sa}^* = k_a(\tilde{x}_n, \tilde{x}_a) \) is the corresponding element in \( K^a \) and \( \beta_a = (K^a + \alpha_a^2 I)^{-1}y^a \). \( \tilde{x}_n \) is the training input set and \( y^a = [y_1^a, ..., y_N^a] \) is the training target set for the corresponding dimension.

### 2.2 Efficient Gaussian processes prediction with uncertainties

In general, considering the model uncertainties in a long-term prediction using GP is difficult since the prediction with uncertain input \( \tilde{x}_n \sim N(\tilde{\mu}, \tilde{\Sigma}) \) at each step follows:

\[
p(\hat{f}(\tilde{x}_n)|\tilde{\mu}, \tilde{\Sigma}) = \int p(\hat{f}(\tilde{x}_n)|\tilde{x}_n)p(\tilde{x}_n|\tilde{\mu}, \tilde{\Sigma})d\tilde{x}_n
\]

which is a non-Gaussian predictive distribution and cannot be computed analytically. Approximating such an intractable marginalization of model input by traditional methods such as Monte-Carlo sampling is computationally demanding, especially for each candidate of the control sequence during optimization. As one solution, analytic movement matching (Deisenroth, Huber, & Hanebeck, 2009; Girard, Rasmussen, Candela, & Murray-Smith, 2003) approximates the non-Gaussian predictive distribution in Equation (5) by a Gaussian distribution that possesses the same mean and variance, and can therefore be expressed analytically.

Figure 3 shows the principle of moment matching, the upper left figure shows GP prediction with eight sample points, the variance is shown as gray area. Define a Gaussian input \( \bar{x}_n \sim N(\mu, \Sigma) \) as the blue curve in the lower left figure, the ground truth of distribution \( p(\hat{f}(\tilde{x}_n)|\mu, \Sigma) \) and the Gaussian distribution approximated by moment-matching are shown as green and red curves in the upper right figure, respectively. Please note that the distribution in upper right figure is along the Y axis of the upper left figure since it is about \( \hat{f}(\tilde{x}_n) \) when observing \( \tilde{x}_n \).

In this study, we propose a modified moment-matching to efficiently optimize the deterministic control sequence by separating the uncertain state and deterministic control in the prediction:

\[
[\mu_{t+1}, \Sigma_{t+1}] = h(\mu_t, \Sigma, u_t).
\]

By assuming the state and control signal are independent, the SE covariance function in Equation (2) can be separated as:
Defining \( k_0(u_0) = k_0(U, u_0) \) and \( k_0(x_0) = k_0(X, x_0) \), the mean and covariance related to Equations (3) and (4) therefore follows:

\[
m_0(x_0, u_0) = (k_0(u_0) \times k_0(x_0))^{\dagger}(K_0 + \alpha_n^2 I)^{-1}y_0
\]

\[
= (k_0(u_0) \times k_0(x_0))^{\dagger} \hat{y}_0
\]

\[
\sigma_0^2(x_0, u_0) = (k_0(u_0) \times k_0(x_0))^{\dagger}(K_0 + \alpha_n^2 I)^{-1}(k_0(x_0) \times k_0(u_0))
\]

Introducing Equations (8) and (9) to Equation (5), we obtain the exact analytical expression of analytic movement matching with deterministic action \( u_\ast \) given the mean and variance of the states:

\[
h(\mu, \Sigma, u_\ast) = \int p(\hat{f}(x_\ast)|x_\ast)p(x_\ast|\mu, \Sigma, u_\ast)dx_\ast.
\]

The expression is detailed in Appendix A.1.

### 2.3 | MPC with multiple steps prediction

Now we are able to efficiently predict the future state of the GP model with uncertain input state and deterministic control via Equation (10). In SPMPC, a MPC controller is employed to provide quick feedback to the constantly changing environment. Defining a one step cost function \( l(\cdot) \) for a specific task, for example, squared Euclidean distance for target reaching and position holding, SPMPC optimizes a \( H \) steps control sequence \( u_\ast, \ldots, u_{\ast,H-1} \) to minimize the expected long-term cost:

\[
[u_\ast, \ldots, u_{\ast,H-1}] = \arg \min_{u_0, \ldots, u_{H-1}} \sum_{i=1}^{H-1} \gamma^{i-1} \mathbb{E}[l(x_i, u_i)]
\]

s. t. \( u_i \in \mathcal{U} \)

where the \( H \) steps states are predicted via Equation (10) with consideration of the uncertainties, \( \gamma \in [0, 1) \) is the discount parameter that encourages optimization to focus on more recent states. The multistep prediction of \( x_i \) is calculated via Equations (6) and (10) given an initial variance \( \Sigma_0 \). \( \mathcal{U} \) is the constrained space of control signals. Any constrained nonlinear optimization method can be applied to search the optimal control sequence; in this study we utilized a single-shooting sequential quadratic programming (SQP; Nocedal & Wright, 2006) implemented by MATLAB optimization toolbox. After optimizing the control sequence on the fly, SPMPC only executes the first control signal \( u_\ast \) to the system and then moves to step \( t + 1 \), where it repeats the process of observation and optimization to form an implicit closed-loop controller that minimizes the cost function while considering both real-time disturbances and uncertainties/errors in GP prediction.

### 2.4 | RL process of SPMPC

In this study, SPMPC employs a RL process that iteratively improves the control performance by explorations and therefore adapts to different environments without human prior knowledge. Starting from initializing the GP model via preprepared samples (e.g., samples generated by random control signals), SPMPC runs multiple iterations or rollouts to generate new samples and update its GP model to gain new model knowledge of interactions with the environment. The whole RL process follows Algorithm 1. In the \( j \)th step of iteration \( i \), the current state is first observed as \( x_j \). The control signal sequence \( [u_1^\ast, u_2^\ast, \ldots, u_{H-1}^\ast] \) is optimized by Equation (11). SPMPC only executes the first control signal \( u_1^\ast \), and ObserveState yields the corresponding target \( y_j \). The state \( x_j \), control \( u_1^\ast \) and target \( y_j = x_{j+1} \) are stored in the sample set. The GP model is updated after each iteration.

---

**FIGURE 3** An example of moment matching. Upper left: the Gaussian process prediction results with right sample points. Lower left: the input Gaussian distribution. Upper right: the real distribution in Equation (5) (green), and the approximated Gaussian distribution from moment matching (red) [Color figure can be viewed at wileyonlinelibrary.com]
3 | SPMPC AUTONOMOUS BOAT DRIVING SYSTEM

In this section, we introduce an autonomous boat driving system based on SPMPC. As shown in Figure 4, the system consists of two main parts: the USV (Section 3.1) and SPMPC (Section 3.2). Three additional modules are developed for better control capability in real-world operation: (1) bias compensation alleviates the effect of control signal during optimization (Section 3.3), (2) a parallel communication node transfers state and control signal between two subsystems with less control delay by using parallel computation (Section 3.4), (3) a quadrant-based action search rule (Section 3.5) efficiently optimizes control signals while avoiding excessive signals.

The data flow in each control cycle is presented as green (state) and blue (control signal) arrows in Figure 4: the current state $s_t$ is observed by the sensors equipped on the USV system and subject to bias compensation to estimate $\hat{s}_t$, which is the state affected by the previous control signal $u_{t-1}^*$. Then the action range for optimization is limited by the quadrant-based action search rule (Section 3.5) efficiently controls the boat while avoiding excessive signals. The SPMPC system will optimize the control sequence $u_t^*, \ldots, u_{t+H-1}^*$ based on multistep prediction, and sends the control signal of the first step back to the USV system following Section 2. The parallel communication node processes the communication and bias compensation/optimization on different cores of CPU to reduce the control delay.

3.1 USV system

As shown in Figure 1, the boat used in this study is a Nissan JoyFisher 25 (length: 7.93 m, width: 2.63 m, height: 2.54 m) fitted with a single steerable SUZUKI DF150AP outboard engine and two sensors: a Furuno SC-30 GPS position/direction/speed sensor and a Furuno WS200 wind sensor. Tables 1 and 2 present all observed states and control signals in this system. The observed states include the vessel’s position, velocity, and direction, the relative wind speed, and direction, and the real value of the engine throttle and rudder angle. Note that the vessel was not equipped with a water current sensor. Therefore both the unobservable ocean current and the observable but unpredictable wind will strongly affect the navigation of the proposed system in real settings. Here these disturbances are alleviated by the following SPMPC system which predicts the vessel dynamics in multiple steps with consideration of environmental uncertainties.

3.2 SPMPC system

The SPMPC system for autonomous boat is designed based on the RL process introduced in Section 2.3. Following Table 2, the states of the GP model is $x = [X, Y, v, \psi, \eta, \delta, v_w \cdot \cos(\phi_w), v_w \cdot \sin(\phi_w)]$, where the relative wind speed and direction are represented by a 2D vector,
the control \( u = \{ \delta, S, \tau \} \), where \( S = 1, -1, 0 \) for shift values F, R, and N, respectively. The target is defined as \( y = [X, Y, \psi, \eta, \delta] \). Building upon our previous work (Cui et al., 2019), the state \( x \) and target \( y \) are extended with the real engine speed and rudder angle to further consider the delay between the control signals and the actual status of the boat’s hardware. In the multistep prediction in Equations (6) and (10), we assume the wind states do not change during prediction by fixing \( v_{rw} \) and \( \psi_{rw} \) from the first step since the wind is difficult to predict in real ocean environment.

### 3.3 Bias compensation

Following the ideal situation described in Algorithm 1, at each time step \( t \) SPMPC first observes the state \( x_t \), then optimizes the control sequence \( [u^*_t, ..., u^*_{t+H-1}] \) and applies \( u^*_t \) to the system before observing the target \( y_t \). However, in the real-world implementation SPMPC optimizes the control sequence while continuously sending \( u^*_{t-1} \) to the system. The observed state \( x_t \) is therefore biased by \( u^*_{t-1} \) during the optimization and may worsen the controller’s performance, especially when the optimization time is lengthy. Figure 5 shows one example. At state \( x_t \), SPMPC optimized a control sequence \( [u^*_t, ..., u^*_{t+H-1}] \) to drive the boat to the target following the dashed green arrow. \( [u^*_t, ..., u^*_{t+H-1}] \) was actually implemented to state \( x_{\text{bias}} \).

### TABLE 1 Unmanned surface vehicle specifications

| Component                  | Description                                      |
|----------------------------|--------------------------------------------------|
| Vessel hull                | Nissan JoyFisher 25                              |
| Length                     | 7.93 m                                           |
| Width                      | 2.63 m                                           |
| Height                     | 2.54 m                                           |
| Weight                     | 1500 kg (empty vessel)                           |
| Propulsion system          | Single steerable outboard engine SUZUKI DF150AP  |
| GPS/speed/direction sensor | FURUNO SC-30                                     |
| Wind sensor                | FURUNO WS200                                     |
| Processing platform        | Intel Core i7-6600U CPU and 16 GB memory          |

### TABLE 2 The observed states and control parameters of the proposed ASV system

| Variable | Description                  | Range             |
|----------|------------------------------|-------------------|
| \( X \)  | The position in X axis      | GPS sensor        |
| \( Y \)  | The position in Y axis      | GPS sensor        |
| \( v_s \) | Boat speed                  | Direction sensor  |
| \( \psi_s \) | Boat direction (heading) | Direction sensor  |
| \( v_{rw} \) | Relative wind speed to boat speed | Wind sensor |
| \( \psi_{rw} \) | Relative wind direction to boat direction | Wind sensor |
| \( \eta \)  | Real engine speed           | Engine            |
| \( \delta \) | Real rudder angle           | Rudder            |
| \( \delta_o \) | The steering angle         | \([-30^\circ, 30^\circ]\) |
| \( \tau \)  | The throttle value of engine | \([0\%, 100\%]\)    |
| \( \hat{R} \) | The shift of engine        | F, R, N           |
which is biased by previous control signals and environmental disturbances during the optimization period $\Delta t$. Therefore the actual boat path will not reach the target following the green arrow. To mitigate this bias, a compensation was introduced in our previous work (Cui et al., 2019) to estimate the biased position $[X_{bias}, Y_{bias}]$ in $x_t$ using a simplified model:

$$
\begin{align*}
X_{bias} &= X + v_t \cdot \sin(\phi_t) \cdot \Delta t \\
Y_{bias} &= Y + v_t \cdot \cos(\phi_t) \cdot \Delta t
\end{align*}
$$

(12)

where $\Delta t$ is the time of executing the previous control signal $u_{t-1}$. The state with biased position will be the input of multistep optimization. Although this model could improve the control performance according to the results seen in our previous work, it remains limited since the boat’s velocity $v_t$ and direction $\phi_t$ are fixed following Equation (12), that is, neither acceleration nor turning can be predicted in such a bias compensation.

In this study we update the bias compensation by employing the GP model learned in GPMPC to predict the biased state via an additional GP regression with one step prediction:

$$
\hat{x}_t = \hat{f}(x_t, u^*_t)
$$

(13)

which naturally considers the full dynamics of the boat including position, velocity, and direction. Please note it is a normal GP regression without uncertainties in input since only one step prediction is required. Although at the early stages of SPMPC the initialized GP model may have large error compared with the model in Equation (12), we postulate the GP model could iteratively capture the boat’s dynamics through the RL process and eventually give a reliable estimate of the biased state.

### 3.4 Parallelized communications node

The role of the communication node in Figure 4 is transferring states read from sensors and control signals optimized by SPMPC. It has been improved to utilize a parallel computation structure to further improve overall control capabilities. As shown in Figure 6a, our previous workflow (Cui et al., 2019) of communication and bias compensation is on a single CPU. At time step $t$, the communication node first sends a control signal $u_{t-1}$ to the USV system, then reads the state $x_t$ from sensors and passes it to SPMPC. It then pauses until the bias optimization and optimization are finished for the current step. Therefore the execution time of the control signal $\Delta t$ would contain both communication process, bias compensation, and optimization; all of which combined is too long a period to robustly control the boat.

To reduce $\Delta t$, we develop a parallel computation structure to process the communication and bias compensation/optimization across different CPU cores. As shown in Figure 6b, at time step $t$, CPU 1 is set to receive the state $x_t$ and send control signal $u_{t-1}$. Unlike the previous structure that directly uses $x_t$ in bias compensation and optimization of the current step, $[x_{t|t}]$ is stored for step $t+1$. In parallel, CPU 2 predicts the biased state $\hat{x}_{t-1} = \hat{f}(x_{t-1}, u^*_{t-1})$ and searches for the optimal control signal $u^*_t$, where $x_{t-1}$ and $u^*_{t-1}$ were stored in step $t-1$ (we define $x_0$ as initial state, $u^*_0$ is a zero vector).
3.5 Quadrant-based action search rule

According to the results from our previous work’s real-world target reaching task (Cui et al., 2019), it is more challenging to hold the boat’s position near the target than reaching the target. This is due to the fact that a control signal which is optimized for the reaching task often gives control signals which are excessive for maintaining a fixed position, and as a result the boat overshoots the target position. Here we develop a quadrant-based action search rule to limited the range of potential control signals when the vessel is close to the target to accomplish the position holding task. Its main idea is to encourage the boat to search in a reasonable range of engine throttle and rudder angle while avoiding excessive control signals relative to the target’s distance and direction to the boat.

The details of the quadrant-based action search rule are presented in Figure 7. Depending on the relative position of the vessel to the target (red dot), the vessel is binned into one of four quadrants. Then the range of the engine throttle and rudder angle is limited according to the relative direction between the vessel’s head and target to encourage the vessel to efficiently change its direction towards the target. A threshold circle with a set radius is also defined; if the vessel is inside the circle, the engine throttle is further limited to 50% to avoid excessive control signals. Conversely, if the vessel is out of the circle, larger engine throttle is encouraged to quickly drive the vessel back to the target. The output of the quadrant-based action search rule is the ranges of control signals $\delta_o$ and $\tau$ that guide the optimization in a smaller action space.

4 REAL EXPERIMENT

4.1 Real experiment setup

In this section, we describe the real-world experiment of applying the proposed autonomous boat driving system described in Section 3 to a position holding task. System hardware for these experiments was provided by Furuno Electric Co., Ltd. The communication node that reads states from the sensors/boat and sends control signal to the rudder/engine is implemented in LabView and runs on a laptop with an Intel Core i7-6600U CPU and 16 GB memory. The SPMPC system is implemented in MATLAB on another laptop with an Intel Core i7-8700k CPU and 32 GB memory. Communication between laptops is handled over TCP/IP.

For this task, we define the initial position of the boat as $p_{\text{target}} = [0, 0]$, with initial orientation close to 0°. Max throttle is limited to 40%, that is, the max speed of engine is 2000-40% = 800 rpm, to ensure safety during the experiment, especially when generating initial samples via random control signals. The objective is to remain as close as possible to the initial position against environmental disturbances. The cost function used in

**FIGURE 7** Overview of the quadrant-based action search rule. In each subfigure, the boat is drawn in blue, and the green circular sector shows the range of heading direction for corresponding situation. The movement directions encouraged by the rule are shown as black arrows. The red circular sector shows the rudder range limited by the rule. The percentage range represents the percentage of engine throttle limited by the rule inside/outside the threshold circle, ± values for shift R and F, respectively. [Color figure can be viewed at wileyonlinelibrary.com]
SPMPC is defined as the squared Euclidean distance between the predicted mean of boat position in the $s$th step $\mathbf{p}_s = [X_s, Y_s]$ and $\mathbf{p}_{\text{target}}$:

$$l(p_s) = \frac{1}{2} \left| \mathbf{p}_s - \mathbf{p}_{\text{target}} \right|^2. \quad (14)$$

The setting of SPMPC including the definition of state $\mathbf{x}$, control signal $\mathbf{u}^*$ and target $\mathbf{y}$, and the assumption of fixed wind all follow Section 3.2. Sparse GP with $N_{\text{sparse}} = 50$ pseudo-inputs was applied for efficient calculation. For each step of SPMPC, the control signal is executed for 3 s to sufficiently operate the hardware, while the optimization time is limited to be within 2 s. SPMPC settings were $H = 3$ steps, that is, SPMPC predicted the boat’s trajectories in approx. 9 s. The discount parameter is set to $\gamma = 0.95$, and the initial variance of each state’s components were set to 0.1.

The full-scale experiments were conducted in Ashiya-hama, Hyogo Prefecture, Japan (Figure 8) on June 6, 2019 and July 30, 2019 with separate areas for training and testing. At each trial, $N_{\text{initial}}$ rollouts with random actions were first conducted to generate $N_{\text{initial}}L_{\text{rollout}}$ samples for initializing GP model, then $N_{\text{trial}}$ iterations were applied to the RL process to generate another $N_{\text{trial}}L_{\text{rollout}}$ samples in $N_{\text{trial}}$ iterations. After learning, control performance was evaluated by multiple test rollouts. All relevant tuning parameters in SPMPC are listed in Table 3.

Training was conducted at 11:00, then testing time followed at around 14:00. Please note that although the training and testing areas are not far to each other, their local disturbances (wind, current, etc.) were very different according to the weather information shown in Figure 9. Therefore, these conditions can be used for evaluating the generalization ability of SPMPC.

4.2 | Real experiment results

In this section, we present the results of the real experiments. Two experiments were conducted in this part. The first experiment investigated the performance of the proposed SPMPC system with all plugins (Section 4.2.1). The second investigated whether the additional states of the engine and rudder contribute to better performance (Section 4.2.2).

4.2.1 | Evaluation of SPMPC system

The first experiment was conducted on June 6, 2019 to investigate the control performances of the proposed system with/without the quadrant-based action search rule. The average weather over Ashiya-hama on the day was cloudy, ocean current speed 0.0–0.2 m/s and direction 45°, wave height 0.3–0.5 m, wind speed 4 m/s and direction 135°. Both wind and current experienced by the boat continuously changed as it moved over the water.

### Table 3 Parameters in sample-efficient probabilistic model predictive control

| Parameter | Description |
|-----------|-------------|
| $N_{\text{initial}}$ | number of rollouts in random sampling process |
| $N_{\text{trial}}$ | number of rollouts/iterations in RL process |
| $N_{\text{sparse}}$ | number of pseudo-inputs of sparse GP |
| $L_{\text{rollout}}$ | length of rollout |
| $H$ | length of prediction in MPC |
Setting the length of rollout $L_{\text{rollout}} = 25$ steps, both autonomous systems with and without enabling the rule for position holding initialized their GP model from 250 samples generated via driving $N_{\text{init}} = 10$ rollouts with random actions. The GP models were then updated via a $N_{\text{train}} = 10$ iterations RL process using SPMPC. The learned GP models were evaluated by three rollouts with length 50 steps. As each step is 3 s, the duration of each trial is about 75 s in training and 150 s in testing. The policies of operating zero actions and random actions with the quadrant-based action search rule were tested as comparisons.

The average distance to the initial position $[0, 0]$ over three test rollouts are compared in Figure 10, and the corresponding example trajectories are shown in Figure 11. The policy with zero actions could not keep the boat in place since strong winds and currents moved the boat far away. The policy with random actions and the rule enabled had a close average position offset to SPMPC without enabling rule according to Figure 10, however its trajectory shows that it also could not keep the boat near its initial position. The random actions limited by the rule helped the boat to drive against the disturbances in the first several steps, but quickly failed after. On the other hand, the SPMPC without the rule performed better with behaviors clearly attempting to drive the boat back to the initial position with an average position offset of approx. 11 m. The best performance came from SPMPC with the rule, successfully holding the position with an average offset of $4.63 \pm 1.36$ m.

We further compare the states and control signals over the example trajectory of SPMPC with/without the rule in Figure 12. By limiting the
action range for optimization by this rule, excessive control signals for both rudder and throttle were alleviated, which resulted in smoother changes in both boat velocity and direction and finally contributed to better control capability in the position holding task.

4.2.2 Evaluation of SPMPC system with additional states

The second experiment was conducted on July 30, 2019 to investigate whether adding states of real engine speed ($\eta_r$) and real rudder angle ($\delta_r$) to the state (and target) of GP model, that is, $x_t = [X_t, Y_t, \dot{X}_t, \dot{Y}_t, \psi_{rw}, \psi_{rw} \sin(\psi_{rw}), \psi_{rw} \cos(\psi_{rw})]$, contribute to improving control performance. For a more challenging task, we extend the length of test rollout to 150 steps while keeping $L_{rollout} = 25$ steps (about 75 s) in the training process. After training, the learned GP models were evaluated by three rollouts with length 150 steps, about 450 s. The average weather over Ashiya-hama area on that day was cloudy, water current speed $\sim 0.0 - 0.2$ m/s north, wave height $0.3$ m, and wind speed $5$ m/s southwest.

The average position offset to the initial position $[0, 0]$ over three test rollouts are compared in Figure 13, the corresponding example trajectories are shown in Figure 14. This result indicates that the additional information of the real engine speed and rudder angle resulted in a better performance in the position holding task. According to the trajectories of states and control signals shown in Figure 15, by adding the real rudder value as state SPMPC would control the rudder more frequently than before. The observation of real engine speed helped the system to detect the delay between the control signal and the hardware which improved control over acceleration without frequently changing the throttle or resulting in excessive velocities.

To further investigate the prediction behavior of SPMPC during this experiment the boat trajectory, SPMPC prediction, and corresponding optimization heat map in the 8, 41, 87, and 125 steps of the example test rollout using SPMPC with the quadrant-based action search rule and additional states are studied in Figure 16.

In each frame at the top of Figure 16, SPMPC was in a dynamical environment. The arrows on background demonstrate the current wind direction as detected by sensor. At each step, it received the current state processed by bias compensation, then optimized the
**FIGURE 12** An example of state trajectories in the first experiment. The data is recorded on the USV system, one time step is about 100 ms. SPMPC, sample-efficient probabilistic model predictive control [Color figure can be viewed at wileyonlinelibrary.com]

**FIGURE 13** The average position offset of Experiment 2. Error bars show the SD. SPMPC, sample-efficient probabilistic model predictive control [Color figure can be viewed at wileyonlinelibrary.com]

**FIGURE 14** Example of real trajectory during a test of experiment 2. The blue circle indicates the threshold circle of the quadrant-based action search rule with radius 7 m. The size of boat is drawn to half scale for clear representation of the trajectory. SPMPC, sample-efficient probabilistic model predictive control [Color figure can be viewed at wileyonlinelibrary.com]
control signal trajectory to minimize the cost function based on multistep prediction. The three step prediction of the boat’s position under the optimized control signal trajectory with corresponding SD are shown in black point and ellipse. To further visualize the movement of the boat over a short period, the boat states in the previous and next five steps are drawn in gradient colors. To represent the SPMPC’s decisions, a heatmap shows the cost function’s value over the first step control signals. The optimization range defined by the rule for position holding is shown in opaque white. The optimization range defined by the rule for position holding is shown in opaque

FIGURE 15  One example of states’ trajectories in test of Experiment 2. The data is recorded on the unmanned surface vehicle system. One time step is about 100 ms [Color figure can be viewed at wileyonlinelibrary.com]
The optimized control signals are shown in the yellow circle. Please note that although the cost function is affected by three steps control signal trajectory, only the first step’s signals are variables in these heatmaps while the others as the signals optimized by SPMPC to fit the 2D plot.

At step 8, affected by the unobservable current and unpredictable wind, SPMPC accurately predicted the future state close to the real trajectory. The heatmaps of SPMPC with and without variance are similar: SPMPC believed driving forward will result in a high cost and decided to return to the initial position with $-30^\circ$ rudder and 40% throttle according to the heatmap. Combining the heatmap with the optimization range defined by the rule for position holding is shown in opaque white. The optimized control signals are shown in the yellow circle [Color figure can be viewed at wileyonlinelibrary.com]

**FIGURE 16** Short samples of prediction and state (upper) and control signals’ trajectories (lower) during the second experiment. One step prediction of sample-efficient probabilistic model predictive control in the upper figure is about 3 s, one time step in the lower figure is about 100 ms. For the figure of boat trajectory, the red marker at the center shows the initial position, while the threshold circle of the quadrant-based action search rule introduced in Section 3.5 is shown in blue. The arrows in the background represent wind direction as detected by sensor. The boat’s current state (position and direction) is represented in green with a black border, while the shapes in blue to red gradient represent the boat state in the previous and next five steps, respectively. For the heatmap, the optimization range defined by the rule for position holding is shown in opaque white. The optimized control signals are shown in the yellow circle [Color figure can be viewed at wileyonlinelibrary.com]
conditions, as it smoothly alters the boat velocity and direction towards the initial position.

At step 41, the boat tried to move backwards toward the initial position. From 5 steps ahead when the boat velocity reached a peak, SPMPC predicted the potential position offset and shifted gear to reverse. At the step 41, since the boat velocity was changed from plus to minus, SPMPC immediately switched the rudder’s direction from 15° to −6° which corrected the disturbance from the wind. Although the real trajectory biased more along the Y axis compared with the prediction due to other unobservable disturbances such as the ocean current, the optimized control signal successfully controlled the boat to hold position.

At step 87, the boat was accelerating forwards. The predicted trajectory is biased by disturbances but still remained close to the real trajectory. One interesting observation is the difference in heatmaps of SPMPC with and without variance. Without considering the variance in multiple step prediction, SPMPC could not accurately predict the boat state and preferred to go further backward which is clearly not an optimal solution. On the other hand, SPMPC with variance could correctly detect the potential cost of driving backward and decided to drive towards the initial position.

At step 125, SPMPC kept the boat near the initial position by carefully switching the shift and rudder to maintain its position at a relatively low velocity. Similar to step 87, whether considering the variance in SPMPC resulted in different optimized actions. Without considering variance, it appears that SPMPC could not predict the effect of wind correctly, and as a result preferred to further accelerate the boat with 25% throttle to against the wind. On the other hand, with consideration of variance, SPMPC decided to slow down the boat and utilized the wind to keep its position.

5.1 | Simulation experiment setup

The simulation used in this section was jointly developed by NAIST and Furuno Electric Co., Ltd. It approximates the boat dynamics in the ocean with disturbances including wind and current, based on expert knowledge and real driving data. A position holding task similar to the real experiment was conducted. The system settings and experimental settings follow Sections 3 and 4, respectively. In simulation the velocity and direction of both wind and current are parameters that change between each step. One time step for SPMPC in this simulation is about 2 s. Please see Appendix A.2 for more details.

5.2 | Simulation experiment results

5.2.1 | Convergence test

The first simulation experiment is to investigate the RL behaviors of SPMPC. Here the MPC prediction horizon \( H = 3 \) steps, that is, SPMPC predicts the boat’s trajectories in 6 s. The length of rollouts is \( L_{\text{rollout}} = 50 \) steps, about 100 s in simulation. The RL processes have \( N_{\text{trial}} = 10 \) iterations. The discount factor is set to \( \gamma = 0.95 \), sparse GP (Neselson & Ghahramani, 2004) with \( N_{\text{sparse}} = 50 \) pseudo-inputs are utilized for efficient calculation, and SPMPC with 100 and 500 samples for initializing the GP model were compared. At the beginning of each experiment, the initial parameters of the environment were set as uniform distributions: \( v_w = U(0, 8) \) m/s, \( \psi_w = U(−180, 180) \), \( v_c = U(0, 0.5) \) m/s and \( \psi_c = U(−180, 180) \). At each step both wind and current slightly change to simulate a real oceanic environment following Equation (A14) in Appendix A.2. After one iteration of learning, the learned GP model was tested by 10 independent rollouts with same initial environmental parameters of the current experiment and length \( L_{\text{rollout}} = 100 \) steps, about 200 s. The final result was from averaging over 10 experiments. A comparative baseline is conducted by applying two PID controllers for rudder angle and engine throttle. The PID error is defined as the angle between boat head and the target, and the distance from boat to the target. Setting the control frequency to 20 Hz, both two PID controllers’ parameters were manually tuned to \( P = 1.0, I = 0.1, D = 0.1 \) based on its average performance on 1000 trials with the same random environmental parameters as above.

Figure 17a,b show the average position offset from the initial position over all tests of SPMPC and the baseline (PID controllers), respectively without and with the quadrant-based action search rule. The upper and lower boundaries of the opaque regions show the worst and best average position offset over all tests. Without the quadrant-based action search rule, the PID controller had an average position offset of about 13 m. SPMPC outperformed the baseline and iteratively reduced the position offset from 12 to 8 m. On the other hand, more initial samples for the GP model did not contribute to a faster learning. With the quadrant-based action search rule, all approaches improved in performance. The PID controller achieved an average position offset near 10 m. SPMPC iteratively reduced the
average position offset to about 5 m, which significantly outperformed the baseline. Also, more initial samples for GP model accelerated the learning process. Please note the worse performance of SPMPC in simulation compared to the real-world experiment is due to the more challenging environmental parameters set in simulation. In the real experiments wind velocity never exceeded 5 m/s, and current velocity never exceeded 0.2 m/s.

We further investigated the prediction capabilities of GP model during the RL process. Figure 18 presents the average boat position errors between the 1, 2, and 3 steps predictions (i.e., 2, 4, and 6 s) of GP model and the true values, in "SPMPC with rule, 500 initial test" of Figure 17b. The opaque areas indicate SD. The results clearly show that SPMPC improved the prediction capability over iterations. It is reasonable that longer step prediction resulted in a larger error due to the increased effects from uncertain disturbances.

We also investigated the effect of *N*<sub>rollout</sub> and *L*<sub>rollout</sub> in the RL process in Figure 19 using the setting of "SPMPC with rule, 500 initial samples." These results indicate that a suitable balance between *N*<sub>rollout</sub> and *L*<sub>rollout</sub> is necessary for SPMPC to perform well. A limited *N*<sub>rollout</sub> resulted in poor generalization ability due to less diversity in experienced environmental settings. An insufficient *L*<sub>rollout</sub> can also prevent SPMPC from learning boat dynamics subject to longer-term disturbances.

All results in this subsection empirically confirm that SPMPC’s RL behavior that updated the model over numerous iterations can successfully learn the position holding task in simulation, and outperform the baseline. Furthermore the proposed quadrant-based action search rule greatly improved the control performance of SPMPC.

### 5.2.2 Control performance test

The second test is to evaluate whether the prediction length and the uncertainties of predicted state contribute to better control results, especially in challenging ocean environments. Six configurations of SPMPC with prediction length *H* = [1, 3, 5] steps (predict...
trajectories in 2, 6, 10 s, variance on/off in GP prediction were tested with three levels of initial environmental settings, to simulate increasingly challenging ocean environments:

1. **Level 1**, \( \psi = 37^\circ + U(-30, 30)^\circ, v_c = 100^\circ, v_w = 2.0 \text{ m/s}, v_c = 0.25 \text{ m/s} \)

2. **Level 2**, \( \psi = 37^\circ + U(-30, 30)^\circ, v_c = 100^\circ, v_w = 4.0 \text{ m/s}, v_c = 0.5 \text{ m/s} \)

3. **Level 3**, \( \psi = 37^\circ + U(-30, 30)^\circ, v_c = 100^\circ, v_w = 8.0 \text{ m/s}, v_c = 1.0 \text{ m/s} \)

Following the setting in Section 5.2.1, SPMPC was initialized with 500 random samples, the RL process took 10 iteration with length \( L_{\text{Rollout}} = 50 \text{ steps, about 100 s} \). The final GP model was tested with 25 rollouts with length \( L_{\text{Rollout}} = 100 \text{ steps, about 200 s} \). The final results were averaged over 25 independent experiments.

The results of both average position offset and optimization time are shown as a box plot in Figure 20. In a relatively mild environment (wind velocity 2 m/s, current velocity 0.25 m/s), SPMPC with one step prediction stayed the boat with median offsets 16.70 m (with variance) and 12.71 m (without variance), while SPMPC with three steps prediction but no variances had a median offset 6.3 m. The SPMPC with both three steps prediction and variances had a better performance of 5.1 m. With five steps prediction, SPMPC had the best performance in median offset: 4.17 m with variance and 1.99 m without variance. In level 2 with double wind and current velocities, SPMPC with three steps prediction and variances stayed the boat with a median position offset 8.13 m, while SPMPC with five steps prediction and variances had a close median position offset 8.09 m. All other configurations had median position offsets larger than 10 m. At level 3 with wind velocity at 8 m/s and current velocity at 1 m/s, SPMPC with five steps prediction and variances outperformed the other configurations with a median offset: 10.04 m. Looking at the average optimization time, although five steps prediction consistently yielded better control performance, the median optimization time per step is around 2 to 4 s. On the other hand, three steps prediction took about 0.4 s in each step. Since the SPMPC in real experiment allows for up to 2 s for optimization, we believe the three steps prediction is an acceptable compromise between computation time and control performance.

These results indicate the importance of multistep prediction with uncertain inputs in SPMPC, especially in particularly challenging environmental conditions. With increased prediction steps, SPMPC had better control performances and higher calculation cost. The uncertainties of predicted state also contributed to better results under strong disturbances.
5.2.3 | Rule and cost function test

In Section 3.5, a quadrant-based action search rule is proposed to limit the range of control signals for the position holding task. One alternative solution to limit excessive control signals in MPC is directly adding a penalty term of control signal in Equation (14) following:

\[ l(p_s, u_s) = \frac{1}{2} ||p_s - p_{\text{target}}||^2 + \alpha ||u_s||^2 \quad (15) \]

where \( p_s \) and \( u_s \) are the position and control signal in the \( s \)th step, \( p_{\text{target}} \) is the target position, \( \alpha \) is the weight of the penalty term.

The third test is to evaluate the control performance of the quadrant-based action search rule compared with the penalty term in cost function defined in Equation (15). Five configurations of SPMPC include using cost function Equation (15) with \( \alpha = [1, 10, 100] \) and using cost function Equation (14) with/without the rule were tested. To uphold the boat’s range of control inputs, only the throttle signal with range \([-100\%, 100\%]\) is limited in Equation (14). Following the settings in Section 5.2.2, we set \( H = 3 \) steps with variance on in the level 1 environment.

The results of average position offset are shown in Figure 21 where the SPMPC with the proposed action search rule outperformed all other configurations while the SPMPC using Equation (15) with different \( \alpha \) could not learn this task. We further compared the position offset and throttle signal of selected configurations in an example rollout in Figure 22. For SPMPC with the rule enabled, the throttle limitation was \([-50\%, 50\%]\) at the beginning when the boat was close to the initial position. It was temporarily removed near the 20-th step since the boat moved bit far, which enabled throttle signals with sufficient power to drive the boat back. On the other hand, the SPMPC using Equation (15) with \( \alpha = 1 \) had a degraded control performance with drastically changing throttle signals as the penalty term contributed less to the cost function. Setting \( \alpha = 10 \), SPMPC had a poor control capability because the agent could not contiguously output sufficient throttle under such a strong limitation of the control signal. When \( \alpha = 100 \), SPMPC almost stopped responding to the environment to avoid the high control signal cost in Equation (15).

These results indicate the proposed quadrant-based action search rule is a suitable approach for SPMPC to limit excessive control signals in position-keeping task. It dynamically determines a search range of control while fairly considering all candidates in optimization. As a comparison, adding a corresponding penalty term to the cost function could not balance the control capability and excessive control signals. The static parameter \( \alpha \) in the penalty term encouraged lower control signals that are usually insufficient to against the uncertain disturbances while expanding it in a dynamic way is difficult.

5.2.4 | Sparse GP test

In this test, the effect of sparse GP’s parameter in SPMPC was investigated. Following the setting in Section 5.2.2, we set \( H = 3 \) steps with variance on in the level 2 environment. Sparse GP models with \( N_{\text{pseudo}} = 10, 20, 50, 100, 500 \) were evaluated in both average position offset and optimization time. According to the results in Figure 23, setting the pseudo-inputs to 10 decreased optimization time to about 0.2 s with a very large position offset of 43.39 m. On the other hand, 500 pseudo-inputs slightly improved the median position offset to 8.05 m with an unacceptable optimization time of 17.4 s. As a compromise we selected 50 pseudo-inputs in all other experiments in both real-world and simulation since it balanced the control performance and optimization time (median position offset 8.13 m, and optimization time 0.46 s).

5.2.5 | Target reaching and position holding task

In the last test, we explored the potential of applying SPMPC with the quadrant-based action search rule to a reaching-target and position-stay task as the first step to combining the real-world experiments in our previous work (Cui et al., 2019) and this article. Starting from the initial position \([0, 0]\) m, the task is to drive the boat to \([100, 100]\) m and hold this position. The cost function is the

![FIGURE 21](https://wileyonlinelibrary.com) Average position offset over 25 tests with different configurations of cost function. SPMPC, sample-efficient probabilistic model predictive control [Color figure can be viewed at wileyonlinelibrary.com]
squared Euclidean distance to the target position. We set a threshold of 7 m to the target; within this threshold, the quadrant-based action search rule is used to limit the optimization.

Setting the initial environment parameters to $w_s = 3 \text{ m/s}$, $c_s = 0.1 \text{ m/s}$, $w_s = 172^\circ$ and $c_d = 65^\circ$, SPMPC initialized the GP model with 500 samples, and then explored 10 iterations with rollout length $L_{\text{rollout}} = 100$ steps. The learned GP model was evaluated by 10 independent rollouts with $L_{\text{rollout}} = 100$ steps. The duration of both training and testing is about 200 s in simulation. The average position offset from steps 51–100 of SPMPC with and without the quadrant-based

![Figure 22](image1.png) **FIGURE 22** Trajectories of position offset and throttle signal of different configurations in one example rollout. One time step is about 2 s. SPMPC, sample-efficient probabilistic model predictive control [Color figure can be viewed at wileyonlinelibrary.com]

![Figure 23](image2.png) **FIGURE 23** The effect of sparse GP’s pseudo-inputs on control performance and optimization time. GP, Gaussian process [Color figure can be viewed at wileyonlinelibrary.com]
action search rule is compared in Figure 24. Their test trajectories were shown in Figure 25. These results clearly show that SPMPC with the quadrant-based action search rule is capable of learning this task with high sample efficiency and robustness against the disturbances in a changing ocean environment.

5.3 | Simulation experiment summary

In this section, the proposed autonomous boat driving system was evaluated in simulation with different settings in algorithm and environment. These results empirically confirmed that: (1) SPMPC iteratively improved both the control performances and model prediction capability through the RL processes; (2) the multistep prediction with uncertain input played an important role in SPMPC especially in environments with large disturbances; (3) the optimal selection of sparse GP pseudo-input to balance the performance and calculation cost; (4) the possibility of applying SPMPC in a task combining target reaching and position holding.

6 | DISCUSSIONS

Since the proposed SPMPC is a general RL approach that iteratively learns the model of vessel in ocean environment without requiring prior model knowledge, it can therefore be easily implemented to a wide range of USVs with different engines and sensors. In this study, SPMPC was evaluated in position-keeping task and in both real word experiment and/or simulation. It is straightforward to extend the application of SPMPC to more complicated scenarios. For example, extending the cost function in Equation (14) to further keeping the direction of boat in position-keeping task, or introducing boat’s rotational motions to the cost function to improve the driving Suitability. Furthermore, with additional radar devices, environmental sensors and other pattern recognition technologies, the proposed method could be a potential solution to other challenging tasks such as collision avoidance and auto-docking.

For future work, in regard to implementation, a real-world task combining goal reaching and position holding will be conducted based on the simulation results introduced in Section 5.2.4. A current sensor will be added to the boat to detect the state of wave, the software will be further updated by moving to C++ and CUDA for higher control frequency and better optimization ability. It would also be of interest to directly learn expert driving skills by building a GP model based on human demonstrations instead of random generated samples, to potentially achieve more human-like autonomous driving.

Algorithmically, since multiple GP models are trained for each target dimension with a SE kernel in this study, both multi-dimensional output GP (Álvarez, Luengo, Titsias, & Lawrence, 2010) and advanced kernel function approximation approaches such as (Le, Sarlós, & Smola, 2013) would improve computational efficiency.

FIGURE 24  The average position offset of the reaching-target and position holding task in simulation. Error bars show the SD. SPMPC, sample-efficient probabilistic model predictive control [Color figure can be viewed at wileyonlinelibrary.com]

FIGURE 25  Example of all 10 test trajectories in the reaching-target and position holding task. The red marker represents the initial position, and the blue circle indicates the zone (radius 7 m) in which the quadrant-based action search rule is active. SPMPC, sample-efficient probabilistic model predictive control [Color figure can be viewed at wileyonlinelibrary.com]
Another interesting topic is introducing the current progresses in safe RL (Berkenkamp, Turchetta, Schoellig, & Krause, 2017; Chow, Ghavamzadeh, Janson, & Pavone, 2017) to SPMPC to safely learn the model via exploration with less risky situation would improve its capabilities of more challenging tasks in real ocean environments. Following many GP-based RL methods (Deisenroth et al., 2013; Kamthe & Deisenroth, 2018) that are commonly verified empirically, we do not focus on the theoretical analysis of performance and converge guarantee of SPMPC in this study, and keep it open for future investigation.

7 | CONCLUSION

In this study SPMPC, a RL approach specialized for autonomous boats, is proposed to iteratively learn boat driving policy in a real ocean environment without prior human knowledge. SPMPC combines GP model prediction with uncertain inputs and the MPC in a model-based RL framework, to address the difficulties in the control of autonomous boats including strong and unpredictable disturbances in the ocean environment and the high sampling cost of RL with real vessels. A corresponding autonomous boat driving system is built based on SPMPC with additional modules including bias compensation, a parallelized communication node and a quadrant-based action search rule to address issues such as control delay and excessive control signal in real-world applications. The proposed system has been validated in a position holding task with a real full-sized boat operating in the open ocean. The results show the proposed system quickly learned a controller to hold the boat’s position against the disturbances, demonstrating SPMPC’s sample efficiency. Several simulation tests have also been conducted as supplementary validation, further evaluating the performance of the proposed method in different environmental and algorithmic settings and its potential to be applied towards more complicated tasks.

ORCID

Yunduan Cui  http://orcid.org/0000-0001-5539-4260

REFERENCES

Álvarez, M., Luengo, D., Titsias, M., & Lawrence, N. (2010). Efficient multiloutput gaussian processes through variational inducing kernels. International Conference on Artificial Intelligence and Statistics (pp. 25–32). Annamalai, A. S., Sutton, R., Yang, C., Culverhouse, P., & Sharma, S. (2015). Robust adaptive control of an uninhabited surface vehicle. Journal of Intelligent & Robotic Systems, 78(2), 319–338. Berkenkamp, F., Turchetta, M., Schoellig, A., & Krause, A. (2017). Safe model-based reinforcement learning with stability guarantees. Advances in Neural Information Processing Systems (NIPS), 908–918. Candela, J. Q., Girard, A., Larsen, J., & Rasmussen, C. E. (2003). Propagation of uncertainty in bayesian kernel models-application to multiple-step ahead forecasting. Proceedings of 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2003). IEEE, volume 2, II-701. Cao, G., Lai, E. M.-K., & Alam, F. (2017). Gaussian process model predictive control of an unmanned quadrotor. Journal of Intelligent & Robotic Systems, 88(1), 147–162. Chow, Y., Ghavamzadeh, M., Janson, L., & Pavone, M. (2017). Risk-constrained reinforcement learning with percentile risk criteria. The Journal of Machine Learning Research, 18(1), 6070–6120.

Cui, Y., Osaki, S., & Matsubara, T. (2019). Reinforcement learning boat autopilot: A sample-efficient and model predictive control based approach. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 2868–2875). Deisenroth, M. P., Fox, D., & Rasmussen, C. E. (2013). Gaussian processes for data-efficient learning in robotics and control. IEEE Transactions on Pattern Analysis & Machine Intelligence, 37(2), 408–423.

Deisenroth, M. P., Huber, M. F., & Hanebeck, U. D. (2009). Analytic moment-based Gaussian process process filtering. The Annual International Conference on Machine Learning (pp. 225–232). Dunabin, M., & Grinham, A. (2017). Quantifying spatiotemporal greenhouse gas emissions using autonomous surface vehicles. Journal of Field Robotics, 34(1), 151–169.

Elkins, L., Sellers, D., & Monach, W. R. (2010). The autonomous maritime navigation (amn) project: Field tests, autonomous and cooperative behaviors, data fusion, sensors, and vehicles. Journal of Field Robotics, 27(6), 790–818.

Erikson, B. O. H., Breivik, M., Wilthil, E. F., FlAten, A. L., & Brekke, E. F. (2019). The branching-course model predictive control algorithm for maritime collision avoidance. Journal of Field Robotics, 36(7), 1222–1249. Ghavamzadeh, M., Engel, Y., & Valko, M. (2016). Bayesian policy gradient and actor-critic algorithms. The Journal of Machine Learning Research, 17(1), 2319–2371.

Girard, A., Rasmussen, C. E., Candela, J. Q., & Murray-Smith, R. (2003). Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. Advances in Neural Information Processing Systems (NIPS), (pp. 545–552).

Hagen, I. B., Kufolaur, D. K. M., Brekke, E. F., & Johansen, T. A. (2018). Mpc-based collision avoidance strategy for existing marine vessel guidance system. 2018 IEEE International Conference on Robotics and Automation (ICRA) (pp. 7618–7623). John, M., Jinkun, W., & Brendan, E. (2018) Sparse gaussian process temporal difference learning for marine robot navigation. 2018 Conference on Robot Learning (CoRL).

Kamthe, S., & Deisenroth, M. (2018) Data-efficient reinforcement learning with probabilistic model predictive control. International Conference on Artificial Intelligence and Statistics (pp. 1701–1710). Kim, H. J., Jordaan, M. I., Saistry, S., & Ng, A. Y. (2004). Autonomous helicopter flight via reinforcement learning. Advances in Neural Information Processing Systems (NIPS) (pp. 799–806).

Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. The International Journal of Robotics Research, 32(11), 1238–1274.

Kongsberg Maritime (2018). Autonomous ship project.Key facts about yara birkeland. Retrieved from https://www.km.kongsberg.com/ks/web/nokbg0240.nsf/AllWeb/4B8113BB707A50A4FC125811D00407045

Kuss, M. (2006). Gaussian process models for robust regression, classification, and reinforcement learning (PhD thesis). Technische Universität.

Le, Q., Sarlós, T., & Smola, A. (2013) Fastfood-computing hilbert space expansions in loglinear time. International Conference on Machine Learning (pp. 244–252).

Lefeber, E., Pettersen, K. Y., & Nijmeijer, H. (2003). Tracking control of an underactuated ship. IEEE Transactions on Control Systems Technology, 11(1), 52–61.

Liu, Z., Zhang, Y., Yu, X., & Yuan, C. (2016). Unmanned surface vehicles: An overview of developments and challenges. Annual Reviews in Control, 41, 71–93.

MacKay, D. J. (2003). Information theory, inference and learning algorithms. Cambridge: Cambridge University Press.

Meyer, E., Robinson, H., Rasheed, A., & San, O. (2020). Taming an autonomous surface vehicle for path following and collision avoidance using deep reinforcement learning. IEEE Access, 8, 41466–41481.
APPENDIX A

Analytic movement matching with deterministic action

Introducing Equations (8) and (9) to Equation (5), we consider its exact analytical expression with deterministic action $u_*$:

$$h(\mu, \Sigma, u_*) = p(\tilde{f}(x_0, u_0) | x_0, u_0) \approx \int p(\tilde{f}(x_0, u_0) | x_0, u_0)p(x_0 | \mu, \Sigma)dx_0.$$  \hspace{1cm}(A1)

The mean of $h(\mu, \Sigma, u_*)$ in each input dimension $a$ is calculated following the derivations in Candela et al. (2003) and Kuss (2006):

$$\mu_{ua} = \int m_{ua}(x_0, u_0)p(x_0 | \mu, \Sigma)dx_0 = \beta_u^T l_0.$$  \hspace{1cm}(A2)

For target dimension $a, b = 1, \ldots, D$, the predicted variance $\Sigma_{aba}$ and covariance $\Sigma_{aba}, a \neq b$ of $h(\mu, \Sigma, u_*)$ follow:

$$\Sigma_{aba} = \EE [\sigma_{ua}^2(x_0, u_0)] + \EE [m_{ua}^2(x_0, u_0)] - \mu_{ua}^2 = \beta_u^T \Lambda \beta_u + \alpha^2 - \text{tr}[(K^2 + \sigma_{ua}^2)^{-1} L] - \mu_{ua}^2.$$  \hspace{1cm}(A3)

$$\Sigma_{aba} = \EE [m_{ua}(x_0, u_0)m_{ub}(x_0, u_0)] - \mu_{uab}^2,$$  \hspace{1cm}(A4)

where $\Lambda$ is the diagonal matrix of training inputs’ length scales in kernel $k_0(x_0, x_0)$.

The vectors $l_0$ and matrices $\Lambda, Q$ have elements:

$$l_0 = k_0(u_0, u_0) \int k_0(x_0, x_0)p(x_0 | \mu, \Sigma)dx_0 = k_0(u_0, u_0) \alpha^2_0 \| \Sigma A_0^{-1} + I \|_F^2 \times \exp \left( -\frac{1}{2} (x_0 - \mu)^T (\Sigma + A_0)^{-1} (x_0 - \mu) \right).$$  \hspace{1cm}(A5)

$$Q_{ij} = \alpha_2^2 \alpha_0^2 k_0(u_j, u_i) k_0(u_i, \mu) 1_{2d} \| A_0^{-1} + I \|_F^2 \times \exp \left( (z_j - \mu)^T (\Sigma + \frac{1}{2} A_0^{-1}) \Sigma A_0^{-1} (z_j - \mu) \right),$$  \hspace{1cm}(A6)

$$Q_{ij} = \alpha_0^2 \alpha_0^2 k_0(u_j, u_i) (A_0^{-1} + A_0^{-1}) \Sigma + I \|_F^2 \times \exp \left( -\frac{1}{2} (x_i - x_j)^T (A_0 + A_0^{-1})^{-1} (x_i - x_j) \right) \times \exp \left( -\frac{1}{2} (z_j - \mu)^T R^{-1} (z_j - \mu) \right).$$  \hspace{1cm}(A7)

Boat model in simulation

In this section, the simulation used in Section 5 is detailed. Following Table 2, the change of boat heading $\Delta \phi$ at each step is calculated as:

$$z_0' = \tilde{A}_0 (A_0 + A_0^{-1})^{-1} x_0 + A_0 (A_0 + A_0^{-1})^{-1} x_0,$$  \hspace{1cm}(A8)

$$R = (A_0^{-1} + A_0^{-1})^{-1} + \Sigma.$$  \hspace{1cm}(A9)
\[
\Delta \psi_i = t \cdot (\omega_i + \omega_w)
\]
\[
\omega_i = \frac{t \cdot K_i \tau}{t + T_i} \cdot \frac{t}{t + T_\delta} \cdot \delta_i
\]
\[
\omega_w = \frac{\alpha_{Kw} \xi t}{t + T_{\text{rw}}} \cdot \psi_{\text{rw}}
\]
\[
\alpha = \begin{cases} 
0, & \text{if } \psi_{\text{rw}} = 0 \text{ or } \pm \pi \\
-\cos \left(\psi_{\text{rw}} - \frac{\pi}{4}\right), & \text{if } \psi_{\text{rw}} > 0 \\
\cos \left(\psi_{\text{rw}} + \frac{\pi}{4}\right), & \text{if } \psi_{\text{rw}} < 0
\end{cases}
\]
\[
\tau = \frac{t \cdot K_r}{t + T_r} \cdot \frac{t \cdot K_e}{t + T_e} \cdot \tau.
\]

\[
\Delta X = v_i \cdot t \cdot \sin(\psi_i) + \beta \cdot v_w \cdot t \cdot \sin(\psi_w) + v_i \cdot t \cdot \sin(\psi_i)
\]
\[
\Delta Y = v_i \cdot t \cdot \cos(\psi_i) + \beta \cdot v_w \cdot t \cdot \cos(\psi_w) + v_i \cdot t \cdot \cos(\psi_i).
\]

The model of wind and current follows:
\[
v_w = v_{w0} + N(\mu_{v_w}, \sigma_{v_w}^2)
\]
\[
\psi_w = \psi_{w0} + N(\mu_{\psi_w}, \sigma_{\psi_w}^2)
\]
\[
v_c = v_{c0} + N(\mu_{v_c}, \sigma_{v_c}^2)
\]
\[
\psi_c = \psi_{c0} + N(\mu_{\psi_c}, \sigma_{\psi_c}^2)
\]

where \(v_{w0}, \psi_{w0}, v_{c0}, \psi_{c0}\) are the initial wind/current speed and direction set at the beginning of each rollout. The values of all parameters are listed in Table A1.

### Table A1: The observed states and control parameters of the proposed ASV system

| Parameter | Description | Value |
|-----------|-------------|-------|
| \(t\)    | Time unit per step (s) | 0.05 |
| \(T_\delta\) | Time constant for command to real rudder | 2.1 |
| \(K_r\) (Forward) | Gain constant for rudder | 0.0086 |
| \(T_r\) (Forward) | Time constant for rudder | 3.6 |
| \(K_r\) (Backward) | Gain constant for rudder | 0.0087 |
| \(T_r\) (Backward) | Time constant for relative wind | 2.5 |
| \(K_{\text{rw}}\) | Gain constant for relative wind | 0.95 |
| \(T_{\text{rw}}\) | Time constant for relative wind | 2.0 |
| \(K_f\) (Forward) | Gain constant for throttle | 65.0 |
| \(T_f\) (Forward) | Time constant for throttle | 1.5 |
| \(K_f\) (Backward) | Gain constant for throttle | 50.0 |
| \(T_f\) (Backward) | Time constant for throttle | 1.5 |
| \(K_s\) (Forward) | Gain constant for engine | 0.00087 |
| \(T_s\) (Forward) | Time constant for engine | 2.2 |
| \(K_s\) (Backward) | Gain constant for engine | 0.00033 |
| \(T_s\) (Backward) | Time constant for engine | 3.5 |
| \(\beta\) | Wind constant | 0.06 |
| \(\mu_{v_w}\) | Mean of wind speed (m/s) | 0.0 |
| \(\sigma_{v_w}\) | SD of wind speed (m/s) | 0.01 |
| \(\mu_{\psi_w}\) | Mean of wind direction (°) | 0.0 |
| \(\sigma_{\psi_w}\) | SD of wind direction (°) | 1.0 |
| \(\mu_{v_c}\) | Mean of current speed (m/s) | 0.0 |
| \(\sigma_{v_c}\) | SD of current speed (m/s) | 0.01 |
| \(\mu_{\psi_c}\) | Mean of current direction (°) | 0.0 |
| \(\sigma_{\psi_c}\) | SD of current direction (°) | 1.0 |