A Fully-autonomous Framework of Unmanned Surface Vehicles in Maritime Environments using Gaussian Process Motion Planning

Jiawei Meng[1], Student Member, IEEE, Ankita Humne[2], Student Member, IEEE, Richard Bucknall[1], Member, IEEE, Brendan Englot[3], Senior Member, IEEE and Yuanchang Liu[1,*], Member, IEEE

Abstract

Unmanned surface vehicles (USVs) are of increasing importance to a growing number of sectors in the maritime industry, including offshore exploration, marine transportation and defence operations. A major factor in the growth in use and deployment of USVs is the increased operational flexibility that is offered through use of optimised motion planners that generate optimised trajectories. Unlike path planning in terrestrial environments, planning in the maritime environment is more demanding as there is need to assure mitigating action is taken against the significant, random and often unpredictable environmental influences from winds and ocean currents. With the focus of these necessary requirements as the main basis of motivation, this paper proposes a novel motion planner, denoted as Gaussian process motion planning 2 star (GPMP2*), extending the application scope of the fundamental Gaussian process-based (GP-based) motion planner, Gaussian process motion planning 2 (GPMP2), into complex maritime environments. An interpolation strategy based on Monte-Carlo stochasticity has been innovatively added to GPMP2* to produce a new algorithm named GPMP2* with Monte-Carlo stochasticity (MC-GPMP2*), which can increase the diversity of the paths generated. In parallel with algorithm design, a Robotic Operating System (ROS) based fully-autonomous framework for an advanced USV, the Wave Adaptive Modular Vessel 20 (WAM-V 20), has been proposed. The practicability

J. Meng[1], Y. Liu[1,*] and R. Bucknall[1] are with the Department of Mechanical Engineering, University College London, Torrington Place, London WC1E 7JE, UK (corresponding author: Yuanchang Liu, yuanchang.liu@ucl.ac.uk, tel: +44 (0)20 7679 7062). A. Humne[2] is with the Department of Microtechnique (Robotics), EPFL, Switzerland. B. Englot[3] is with the Department of Mechanical Engineering, Stevens Institute of Technology, Hoboken, NJ, USA.

of the proposed motion planner as well as the fully-autonomous framework have been functionally validated in a simulated inspection missions for an offshore wind farm in ROS.

Index Terms

Unmanned surface vehicles, environment characteristics, GP-based path planning, interpolation strategy, Monte-Carlo stochasticity, fully-autonomous framework

I. Introduction

The planning of trajectories in complex maritime environments plays a critical role in developing autonomous maritime platforms such as USVs. The paths generated for operations in maritime environments should not only ensure the success of a mission but, wherever and whenever possible, actively try to minimise the energy consumption during a voyage. Even with growing recognition of the importance of motion planning algorithms for USVs, two major challenges have largely hindered their progress of development including: 1) the majority of mainstream motion planning algorithms do not encompass proper consideration of the environmental impacts such as winds and surface currents and 2) among the minority of algorithms that do take these environmental characteristics into account, important metrics including the computation time and path quality are not up to that minimum standard of quality required for practical applications. These aforementioned challenges have been addressed to some extent in the past few years, but there is a need to further optimise the solutions.

Current existing mainstream motion planning algorithms can be divided into four categories: 1) grid-based algorithms [1], [2], [3], 2) sampling-based algorithms [4], [5], [6], 3) potential field algorithms [7], [8] and 4) intelligent algorithms [9], [10], [11], variations of which have been applied across different robotic domains. All the aforementioned algorithms have been developed over many years and have made an incredible contribution to robotic motion planning problems. Nevertheless, these algorithms have some drawbacks and cannot fully meet the requirements for motion planning in practical application scenarios. Grid-based algorithms require a post-processing path smoothing procedure to satisfy the non-holonomic constraints of vehicles [12]. Sampling-based and intelligent algorithms might require an extremely long computation time to ensure convergence, otherwise the distance and smoothness of the paths can not be guaranteed [13], [14]. Potential field algorithms

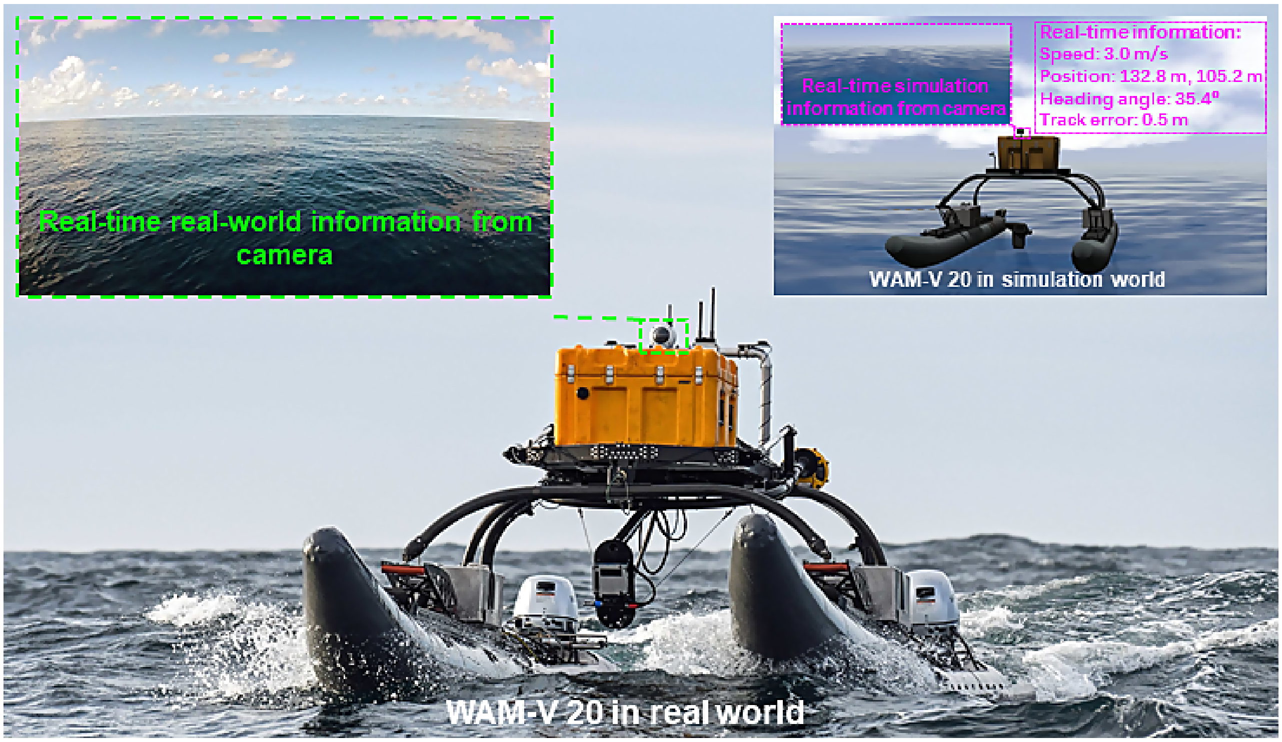

Fig. 1. A demonstration of WAM-V 20 USV in the real world and the virtual maritime scenario. The virtual maritime scenario is highly similar to the real world, where the real-time camera information, position, speed, heading angle and track error can be measured.

suffer from the limitations of local minima and require additional strategies to avoid this issue [15]. Meanwhile, these motion planning algorithms are not designed for maritime environments with time-varying ocean currents. Another perspective of categorising different motion planning algorithms can be found in [16].

To address the problems in practical application scenarios, trajectory optimisation algorithms have been proposed in recent years [17], [18], [19], [20], [21]. One of them is the GP-based motion planning algorithm [17], [21] that represents trajectories as samples from Gaussian processes in the continuous-time domain and optimises them via probabilistic inference. This novel motion planning paradigm brings two significant benefits: 1) the capability of smoothing the path in line with the planning process based on the specification of the system's dynamic models and 2) the superiority in convergence speed through the employment of a fast-updating inference tool such as a factor graph [22]. However, there are still some constraints when it comes to implementing trajectory optimisation algorithms in maritime environments, and the issues of integrating characteristics of maritime environments such ocean currents and avoiding dense obstacles remains especially challenging.

Another research bottleneck for USV development is the lack of high-fidelity environments. Fig. 1 compares a typical catamaran, the WAM-V 20 USV, in real world and virtual maritime scenarios. In this high-fidelity virtual maritime scenario, physical fidelity and visual realism with real-time execution requirements are well-balanced. In general, establishing practical experimental platforms would be expensive. By developing high-fidelity simulation environments, validating the newly proposed motion planning, control and any other algorithms can be conducted in an efficient and low-cost manner.

In fact, simulations with a sufficient level of fidelity have been gradually adopted for USV platforms. Game engines such as Unity [23] and Unreal Engine [24] can present a vivid virtual world, which might be suitable simulation platforms for motion planning and control algorithms. However, most of them do not have a dedicated support for robotics and the hardware requirements for running these game engines are usually difficult to satisfy. In 2002, an open-source simulation platform designed for supporting various indoor and outdoor robotic applications was proposed, namely the Gazebo [25]. Specifically, it delivers the following benefits that made it become the most popular simulation platform among robotic researchers: 1) it supports the use of different physics engines to simulate collision, contact and reaction forces among rigid bodies, 2) its sensor libraries are progressive due to the open source facility and 3) it supports for robotics middle-ware based upon a well-developed messaging system.

Nevertheless, most of the simulators based on the fundamental structure of Gazebo are designed for terrestrial, aerial and space robots [26], [27], [28]. To address this deficiency and provide a standard simulator for the development and testing of algorithms for USVs, the Virtual RobotX simulator (VRX) was proposed in 2019. VRX is a Gazebo-based simulator capable of simulating the behaviour of USVs in complex maritime circumstances with waves and buoyancy conditions [29]. Also, a mainstream catamaran (WAM-V 20 USV) model is provided in VRX with an easy-to-access interface to any self-designed autopilot. There is, however, a lack of a fully-autonomous navigation system in VRX, especially a system integrating both motion planning capability and autopilot.

To bridge these research gaps, this paper has specifically focused on developing a new motion planning paradigm for USVs with the main contributions summarised as follows:

- A new GP-based motion planning algorithm, named as MC-GPMP2*, has been de-

veloped by integrating a Monte-Carlo stochasticity to enable an improved collision avoidance capability.
- A fully-autonomous framework for USVs has been designed for the VRX simulator.
- Enriched high-fidelity tests have been carried out in ROS to simulate offshore wind farm operations using USVs, where the superiority of the proposed motion planning algorithms is properly demonstrated.

The rest of the paper is organised as follows. Section 2 formulates the problem and discusses the mathematical model of the conventional GP-based motion planning algorithm in various complex environments. Section 3 describes the Monte-Carlo sampling and introduces it into our motion planning algorithm. Section 4 presents the modelling and control of the WAM-V 20 USV in ROS. Section 5 demonstrates the proposed path planner's simulation results and then compares them with the results obtained from a series of mainstream motion planning algorithms. Section 6 demonstrates the practical performance of the proposed path planning algorithm and autopilot in ROS, followed by the conclusion and indications for future work in Section 7.

II. GP-based Motion Planning in Various Complex Environments

This section explains the GP-based motion planning algorithms in general and proposes a new method named the Gaussian process motion planner 2 star (GPMP2*), which will be developed and applied to motion planning for autonomous vehicles such as USVs and unmanned underwater vehicles (UUVs).

A. Problem formulation as trajectory optimisation

GP-based motion planning algorithms can be applied to solve the problem of trajectory optimisation, i.e. employing Gaussian Processes to optimise trajectories in an efficient manner. Formally, the trajectory optimisation aims to determine the best trajectory from all feasible trajectories while satisfying any user defined constraints and minimising any user prioritised costs [30], [31], [19]. By considering a trajectory as a function of continuous time t, such an optimisation process can be formulated as the standard form of an optimisation problem with continuous variables as:

$$
\begin{aligned}
\text{minimise} \quad & F[\theta(t)] \\
\text{subject to} \quad & G_i[\theta(t)] \leq 0,\ i = 1, \ldots, m_{ieq} \\
& H_i[\theta(t)] = 0,\ i = 1, \ldots, m_{eq}.
\end{aligned} \tag{1}
$$

where $\theta(t)$ is a continuous-time trajectory function mapping a specific moment t to a specific robot state θ. $F[\theta(t)]$ is an objective function to find the best trajectory by minimising the higher-order derivatives of robot states (such as velocity and acceleration) and collision costs. $G_i[\theta(t)]$ is a task-dependent inequality constraint function and $H_i[\theta(t)]$ is a task-dependent equality constraint function that contain the desired start and goal robot states with specified configurations.

As stated in [17], [32], by properly allocating the parameters of low-resolution states with relatively large sample interval Δt (defined as support states) and interpolating high-resolution states with relatively small sample interval $\Delta\tau$ (defined as interpolated states), the computational cost of Gaussian Processes can be efficiently reduced and a continuous-time trajectory function represented by a Gaussian Process can be shown as:

$$
\theta(t) \sim \mathcal{GP}(\mu(t), K(t, t')), \tag{2}
$$

where $\mu(t)$ is a vector-valued mean function and $K(t, t')$ is a matrix-valued covariance function.

For the given optimization problem, the objective function is given as:

$$
F[\theta(t)] = F_{\text{gp}}[\theta(t)] + \omega_1 F_{\text{obs}}[\theta(t)] + \omega_2 F_{\text{env}}[\theta(t)], \tag{3}
$$

where $F_{\text{gp}}[\theta(t)]$ is the GP prior cost, $F_{\text{obs}}[\theta(t)]$ is the obstacle collision cost and $F_{\text{env}}[\theta(t)]$ is the environment characteristic cost. ω_1 and ω_2 are the weight coefficients given to these costs. At this juncture we specifically highlight the inclusion of the environment cost ($F_{\text{env}}[\theta(t)]$) as it is of particular importance when considering marine vehicles. For other types of vehicles, costs can be adjusted as required.

B. Motion planning as probabilistic inference

From another perspective, GP-based motion planning algorithms can also be viewed as probabilistic inference problems, where Bayesian inference is applied to find the optimal trajectory. The detailed explanation of using Bayesian inference to solve the trajectory optimisation problem in (1) can be found in [20]. In this subsection, we summarise the work in [20] and extend it to the more general case that includes multiple planning constraints.

By exploiting the sparsity of the underlying problem, probabilistic inference, such as Bayesian inference, is effective for solving optimisation problems and the optimisation problem in (1) can be converted into the following Bayesian inference:

$$\theta^* = \arg\max_{\theta}\ p(\theta) l(\theta; e), \tag{4}$$

where $p(\theta)$ represents the GP prior that encourages smoothness of trajectory and $l(\theta; e)$ represents a likelihood. More specifically, the GP prior distribution is given in terms of the mean μ and covariance K:

$$p(\theta) \propto \exp\left\{-\frac{1}{2}||\theta - \mu||_K^2\right\}, \tag{5}$$

whereupon the GP prior cost in (3) is given as the negative natural logarithm of the prior distribution:

$$F_{\text{gp}}[\theta(t)] = F_{\text{gp}}[\theta] = \frac{1}{2}||\theta - \mu||_K^2. \tag{6}$$

The likelihood in the above Bayesian inference can be viewed as a combination of different categories of likelihoods such as the obstacle collision likelihood and the environment characteristic likelihood, thereby it is named as the combined likelihood. Moreover, it is worth noting that the distribution of the combined likelihood can be written into a product of the distributions from all the subcategory likelihoods using the features of the exponential distribution:

$$l(\theta; e) = \underbrace{\exp\left\{-\frac{1}{2}||g_1(\theta)||_{\Sigma_{\text{obs}}}^2\right\}}_{l(\theta; e_{\text{obs}})} \cdot \underbrace{\exp\left\{-\frac{1}{2}||g_2(\theta)||_{\Sigma_{\text{env}}}^2\right\}}_{l(\theta; e_{\text{env}})} \tag{7}$$

$$= \exp\left\{-\frac{1}{2}||g_1(\theta)||_{\Sigma_{\text{obs}}}^2 - \frac{1}{2}||g_2(\theta)||_{\Sigma_{\text{env}}}^2\right\} \tag{8}$$

Σ_{obs} and Σ_{env} are the diagonal covariance matrices with regard to collision and environmental characteristics:

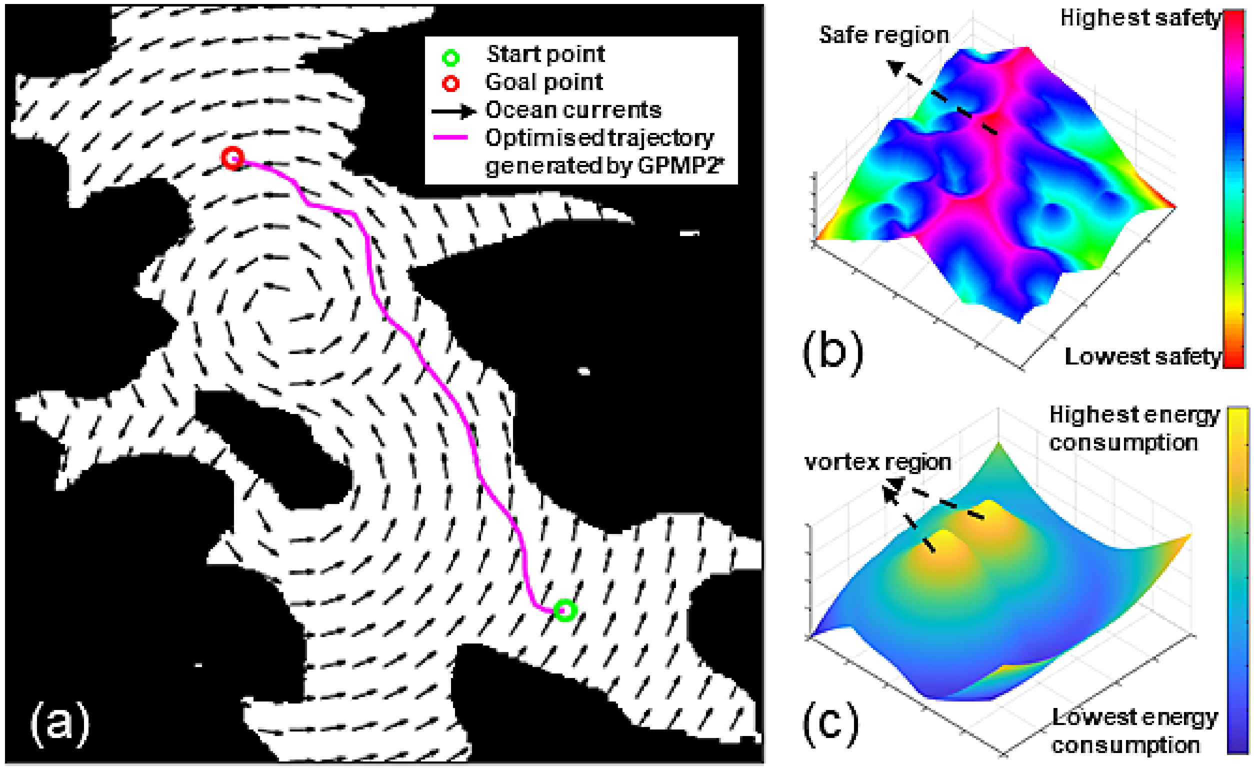

Fig. 2. An example of the proposed GPMP2* motion planning algorithm: (a) demonstrates the optimised trajectory generated by GPMP2*, (b) demonstrates the signed distance field generated by the obstacle collision likelihood function and (c) demonstrates the environment characteristic field generated by the environment characteristic likelihood function.

$$\Sigma_{\text{obs(env)}} = \text{diag}[\sigma_{\text{obs(env)}}], \tag{9}$$

where σ_{obs} and σ_{env} are the weighting coefficients with regard to collision and environment characteristics. $g_1(\theta)$ and $g_2(\theta)$ are defined as a vector-valued obstacle cost function and a vector-valued environment characteristic cost function. More specifically, the definition of $g_1(\theta)$ is given as:

$$g_1(\theta_i) = [c(d(x(\theta_i, S_j)))]_{1 \leq j \leq M}, \tag{10}$$

where $c(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is the workspace cost function that penalises the set of points $B \subset \mathbb{R}^n$ on the robot body when they are in or around an obstacle, $d(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is the signed distance function that calculates the signed distance of the point, x is the forward kinematics function, S_j is the sphere on the robot model and M is the number of spheres that represents the robot model. An example of a constructed signed distance field is graphically shown in Fig. 2 (b), and the obstacle collision cost in (3) can be given as:

$$F_{\text{obs}}[\theta(t)] = \int_{t_0}^{t_N} \int_B c(x(\theta(t), u))||\dot{x}(\theta(t), u)||dudt. \tag{11}$$

where u represents the known system control input. Also, the definition of $g_2(\theta)$ is given as:

$$g_2(\theta_i) = [e(x(\theta_i, S_j))]_{1 \le j \le M}, \tag{12}$$

where $e(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is the environment compensation function that integrates the relevant environment characteristics such as surface wind and ocean currents on the set of points $B \subset \mathbb{R}^n$ on the robot body.

The environment compensation function is defined as a metric calculated using an anisotropic fast marching algorithm as stated in [33], [32]. Such a metric can measure the energy consumption rate at each pixel so that trajectories can be generated to avoid high energy consumption regions (the bright regions in Fig. 2 (c)). The environment information is simulated as a vortex function in this work but any real-time statistical data can also be extracted and used as stated in [13]. The environment characteristic cost in (3) can therefore be given as:

$$F_{\text{env}}[\theta(t)] = \int_{t_0}^{t_N} \int_B e(x(\theta(t), u))||\dot{x}(\theta(t), u)||dudt, \tag{13}$$

where u represents the known system control input.

Now we can rewrite the Bayesian inference in (4) into the following form on the basis of the information provided by (5) and (8):

$$\theta^* = \arg\max_{\theta}\ p(\theta)l(\theta; e) \tag{14}$$

$$= \arg\max_{\theta}\ \{-\log(p(\theta)l(\theta; e))\} \tag{15}$$

$$= \arg\max_{\theta}\ \left\{\frac{1}{2}||\theta - \mu||_K^2 + \frac{1}{2}||g_1(\theta)||^2_{\Sigma_{\text{obs}}} + \frac{1}{2}||g_2(\theta)||^2_{\Sigma_{\text{env}}}\right\}. \tag{16}$$

Similarly, we can rewrite the objective function in (3) into the following form on the basis of the information provided by (6), (11) and (13):

$$\begin{aligned} F[\theta(t)] = \frac{1}{2}||\theta - \mu||_K^2 + \lambda_1 \int_{t_0}^{t_N} \int_B c(x(\theta(t), u))||\dot{x}(\theta(t), u)||dudt \\ + \lambda_2 \int_{t_0}^{t_N} \int_B e(x(\theta(t), u))||\dot{x}(\theta(t), u)||dudt, \end{aligned} \tag{17}$$

where λ_1 and λ_2 correspond to σ_{obs} and σ_{env}, respectively.

A notable advantage of the proposed motion planning algorithm is that when multiple environment constraints need to be considered simultaneously, these constraints can be formulated as various subclass environment characteristic likelihoods. By taking advantage of the features of exponential distributions, subclass likelihoods can be further integrated into a superclass environment characteristic likelihood to enable a fast summation of constraints. To gain a more intuitive understanding of the feasibility of the proposed motion planning algorithm, Fig. 2 demonstrates an example of how the GP-based motion planner can be used to avoid obstacles as well as vortexes. Also, the proposed method can be used in either 2-dimensional (2D) or 3-dimensional (3D) environments. Overall, any type of GP-based motion planning method that incorporates the characteristics of the environment through adding corresponding likelihood to probabilistic inference can be viewed as GPMP2*.

III. GP-based Motion Planning with Incremental Optimisation Characteristics

This section provides detail of the proposed MC-GPMP2* algorithm. The Monte-Carlo sampling based, obstacle space estimation is first introduced and this is then followed by the details of sampling point interpolation and incremental inference.

A. Obstacle space estimation using Monte-Carlo sampling

Monte-Carlo sampling is a highly efficient statistical method to determine the approximate solution of many quantitative numerical problems. It can reduce the computation time when there is a relatively high complexity in sampling space [34], [35]. In this work, this sampling method is used to estimate the ratio of the obstacle space to the entire sampling space, especially when the obstacle space has a relatively irregular shape. Algorithm 1 demonstrates the specific procedure of the estimation, where the random sample point is generated from a continuous uniform distribution:

Algorithm 1: Obstacle Space Estimation using the Monte-Carlo Sampling (MC-EstimateObstacleSpace)

Input: 3-dimensional sampling space $\mathcal{X}_{x,y,z}$ and the total number of samples N_{spl}

for $i = 1, 2, \ldots, N_{\text{spl}}$ do

  Generate a random sample point inside the 3-dimensional sampling space $\text{X}_{\text{rand}} \leftarrow \text{Sample}(\mathcal{X}_{x,y,z})$;

  if $(\text{CollisionFree}(\text{X}_{\text{rand}}, \mathcal{X}_{x,y,z}) == \text{TRUE})$ then

    Accept the random sample point X_{rand} by increasing the accepted sample number $N_{\text{ac}} = N_{\text{ac}} + 1$

  end

  if $(\text{CollisionFree}(\text{X}_{\text{rand}}, \mathcal{X}_{x,y,z}) == \text{FALSE})$ then

    Reject the random sample point X_{rand} by maintaining the previous accepted sample number $N_{\text{ac}} = N_{\text{ac}}$

  end

end

Compute the ratio of the obstacle space to the entire sampling space through $P_{\text{obs}} = \frac{N_{\text{ac}}}{N_{\text{spl}}}$

Output: Obstacle space proportion P_{obs}

Notes: The pixels inside the obstacle space are '1' and the pixels outside the obstacle space are '0'. $\text{CollisionFree}(\text{X}_{\text{rand}}, \mathcal{X}_{x,y,z}$ is a function to check whether the random generated node X_{rand} is inside the obstacle space or not.

$$(x, y, z) \sim \mathcal{U}(a, b), \tag{18}$$

where $a = (a_1, a_2, a_3)^T$ is a vector-valued lower bound function representing the lower bounds of the 3-dimensional sampling space, $b = (b_1, b_2, b_3)^T$ is a vector-valued upper bound function indicating the upper bounds of the sampling space. The probability density function of the continuous uniform distribution at any point $(x, y, z) \in \mathbb{R}^3$ inside the sampling space can be given as:

$$f(\lceil x \rceil, \lceil y \rceil, \lceil z \rceil) = \frac{1}{\prod_{i=1}^{3}(b_i - a_i)}, \tag{19}$$

where $\lceil \cdot \rceil$ is the ceiling function used to round-up to the nearest integer and the volume of the entire sampling space can be represented by $\prod_{i=1}^{3}(b_i - a_i)$.

In Algorithm 1, based on the law of large numbers [36], [37], [38], the accuracy of the estimation of obstacle space gradually increases as the number of samples increases. Its convergence rate is $\mathcal{O}(\frac{1}{\sqrt{N}})$, which means that quadrupling the total number of samples reduces the algorithm's error by half, regardless of the dimensions of the sampling space [39]. Therefore, Algorithm 1 can provide a reasonably accurate result once the total number of samples exceeds a specific threshold. When the total number of samples in Algorithm 1 equals the total number of pixels on the map, the sampling algorithm becomes a traversal algorithm and generates a result with an accuracy that can be considered absolute.

In general, using GP-based motion planning in conjunction with Algorithm 1 to construct a modified motion planning algorithm can offer two notable advantages:

- Shortening the execution time of GP-based motion planning, especially for high-dimensional problems;
- Shortening the path length and improving the path quality by enhancing the diversity of the generated trajectory.

B. Monte-Carlo based GP interpolation

As mentioned in Section. II, apart from the support states, a major benefit of using GPs is the facility to query the planned state at any moment of interest. In addition, according to [20], trajectories generated by a GP-based motion planner can be fine-tuned by increasing the number of states. Therefore, to facilitate obstacle avoidance, in this paper, it is proposed to interpolate additional states between two support states according to the obstacle estimation of Monte-Carlo.

Similar to previous research [20], [19], [17], [21], a linear time-varying stochastic differential equation (LTV-SDE) is adopted to represent the motion model as:

$$\dot{\theta}(t) = A(t)\theta(t) + u(t) + F(t)w(t). \tag{20}$$

where $A(t)$ and $F(t)$ are time-varying matrices of the system, $u(t)$ is the control input and $w(t)$ is the white process noise represented as:

$$w(t) \sim \mathcal{GP}(0, Q_c\delta(t,t')), \tag{21}$$

where Q_c is the power-spectral density matrix and $\delta(t,t')$ is the Dirac delta function. Based on (20), a queried/interpolated state $\theta(\tau)$ at $\tau \in [t_i, t_{i+1}]$ is a function only of its neighboring state as (the detailed proof of this is presented in [20], [19], [17], [21]):

Algorithm 2: Building Factor Graph with the Monte-Carlo Stochasticity (MC-BuildFactorGraph)

```
Input: Total number of sub-searching regions N
Add Prior Factor
for i = 1, 2, ..., N do
    Add Obstacle Factor and Environment Factor
    Compute total number of sample points in the low-resolution region N_j:
    P_obs ← MC-EstimateObstacleSpace
    N_j = λ · P_obs
    for j = 1, 2, ..., N_j do
        Add GP Prior Factor, Interpolated Obstacle Factor and Interpolated Environment Factor
    end
end
Add Prior Factor
Output: Factor graph G_mc
Notes: λ represents a self-defined scaling term.
```

$$\theta(\tau) = \widetilde{\mu}(\tau) + \Lambda(\tau)(\theta_i - \widetilde{\mu}_i) + \Psi(\tau)(\theta_{i+1} - \widetilde{\mu}_{i+1}), \tag{22}$$

where

$$\begin{aligned} \Lambda(\tau) &= \Phi(\tau, t_i) - \Psi(\tau)\Phi(t_{i+1}, t_i), \\ \Psi(\tau) &= Q_{i,\tau}\Phi(t_{i+1}, \tau)^T Q_{i,i+1}^{-1} \qquad , \end{aligned} \tag{23}$$

where $\Phi(*,*)$ is the state transition matrix and $Q_{a,b}$ is:

$$Q_{a,b} = \int_{t_a}^{t_b} \Phi(b,s)F(s)Q_c F(s)^T \Phi(b,s)^T \mathrm{ds}. \tag{24}$$

In general, (22) can be used to interpolate a series of dense states to facilitate the generation of collision-free trajectories while keeping a relatively small number of support states to maintain low computational cost. As stated previously, the strategy of interpolating dense states is lacking in the previous research [20], [19], [17], [21]. Therefore, it is proposed that the number of interpolated states should be determined by the proportion of the obstacle space relative to the whole region space (P_{obs}), and such a proportion can be quickly estimated by Monte-Carlo sampling as described in Algorithm 1. In addition, the Monte-Carlo stochasticity adds a variation to the number of interpolated states ($N_j = \frac{t_{i+1}-t_i}{\tau} = \lambda \cdot P_{\text{obs}}$) to test the optimal number of interpolated states incrementally, where λ

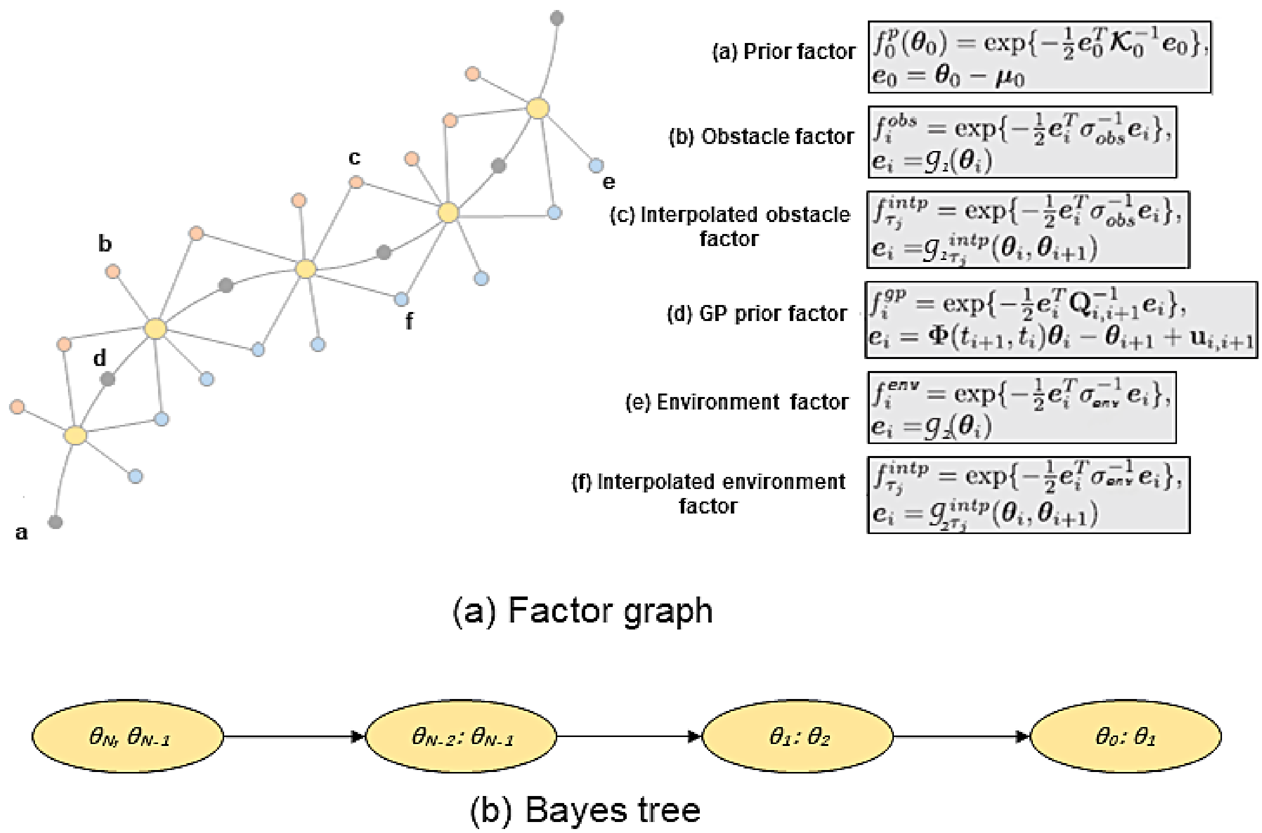

Fig. 3. A demonstration of the factor graph and Bayes tree in our problem: (a) illustrates the factor graph, containing six categories of factors including Prior factor, GP prior factor, Obstacle factor, Interpolated obstacle factor, Environment factor and Interpolated environment factor, (b) illustrates the Bayes tree, indicating the conditional dependencies between various states.

is a self-defined scaling term and P_{obs} is computed by Algorithm 1. More specifically, P_{obs} tends to increase as the volume of obstacles within a specific region increases, leading to a growth in the number of interpolated states of this region N_j. Interpolated states with relatively high densities can improve the performance of the motion planning algorithm on avoiding obstacles as well as smoothen the generated trajectory.

C. Probabilistic inference using the factor graph

Given the Markovian structure of the trajectory enabled by the linear time-varying stochastic differential equation (LVT-SDE) and the sparsity of the underlying problem, the posterior distribution (or the optimised trajectory) can be converted into a factor graph to perform inference incrementally. More specifically, the factor graph is a bipartite graph that can express any inference in a more intuitive graphical manner. It is bipartite as there are only two categories of nodes existing in the graph, i.e. variable nodes and factor nodes [22]. The factorisation of the posterior in our problem is formulated as:

$$p(\theta|e) \propto \prod_{m=1}^{M} f_m(\Theta_m), \tag{25}$$

where f_m are factors on variable subset Θ_m.

Then the factor graph can be converted into a Bayes tree based on the variable elimination process [40], [41], [42], [43]. The Bayes tree in our problem, as converted by (25), is:

$$p(\Theta) = \prod_{j} p(\theta_j|S_j), \tag{26}$$

where θ_j are the states and S_j denotes the separator for state θ_j which is comprised of the nodes in the intersection of the state θ_j and its parent.

To gain a more intuitive understanding regarding the factor graph, a comprehensive structure illustrating how the different factors are integrated as well as converted into a Bayes tree for our problem is demonstrated in Fig. 3. Furthermore, the specific process of building a factor graph with the Monte-Carlo stochasticity is detailed in Algorithm 2.

D. Incremental optimising motion planning based on GPs

The Gaussian process motion planner 2 star with the Monte-Carlo stochasticity (MC-GPMP2*) is proposed in this subsection by integrating the aforementioned information. The pseudo-code of the proposed motion planner is detailed in Algorithm 3 with key information explained as:

- First, the start state θ_0, goal state θ_N and replanning iteration N_{replan} are required as inputs.
- Next, the signed distance field is computed based on the obstacle cost function (as described in (10)) and the environment characteristic field is computed based on the environment characteristic cost function (as described in (12)), to construct the combined likelihood (as described in (8)).
- A factor graph with Monte-Carlo stochasticity is built based on the MC-EstimateObstacleSpace (Algorithm 1) and the MC-BuildFactorGraph (Algorithm 2) and then the optimal path θ^* is inferred based on the Levenberg-Marquardt algorithm [44].

Algorithm 3: Gaussian Process Motion Planner 2 star with the Monte-Carlo Stochasticity (MC-GPMP2*)

Input: Start state θ_0, goal state θ_N and replanning iteration N_{replan}

Precompute Signed distance field (SDF) and environment characteristic field (ECF)

for $i = 1, 2, \dots, N_{\text{replan}}$ do

  $G_{mc} \leftarrow$ MC-BuildFactorGraph

  $\theta^*(i) \leftarrow \text{LM}(\theta_0,\ \theta_N,\ G_{\text{mc}},\ SDF,\ ECF)$

  $\theta^* \leftarrow \theta^*(i)$

  if $\{L[\theta^*(i-1)] <= L[\theta^*(i)]\}$ or $\{\text{CollisionFree}[\theta^*(i)] == \text{FALSE}\}$ then

    $\theta^* = \theta^*(i-1)$

  end

end

Output: Optimal path θ^*

Notes: SDF is calculated by inputting the motion planning space into the workspace cost function. ECF is calculated by inputting the motion planning space into the environment compensation function. LM($\cdot$) represents the Levenberg-Marquardt algorithm. L($\cdot$) represents the function to measure the total length of the generated path.

- The previous step is repeated several times based on the number of replanning iterations N_{replan} required to optimise the path θ^*.

To better understand the functionality of the proposed motion planner, a comparison of the paths generated by MC-GPMP2* and GPMP2 is presented in Fig. 4. MC-GPMP2* generates relatively diversified paths when there are a relatively small number of sample points. Conversely, MC-GPMP2* generates a path with a high level of similarity compared with GPMP2 when there is a relatively larger number of sample points.

IV. WAM-V 20 USV modeling and control in ROS

In this section, detail will be provided regarding the proposed fully-autonomous navigation framework to navigate and control WAM-V 20 USVs in ROS. Overall, the proposed framework includes three major components: 1) motion planner, generating an optimised path according to obstacles and environment characteristics, 2) navigation refinement system to generate the USV heading angles needed to accurately track the paths and 3) autopilot adjusting the angle of deflection of rudders and the rotational speed of USVs to match the desired values.

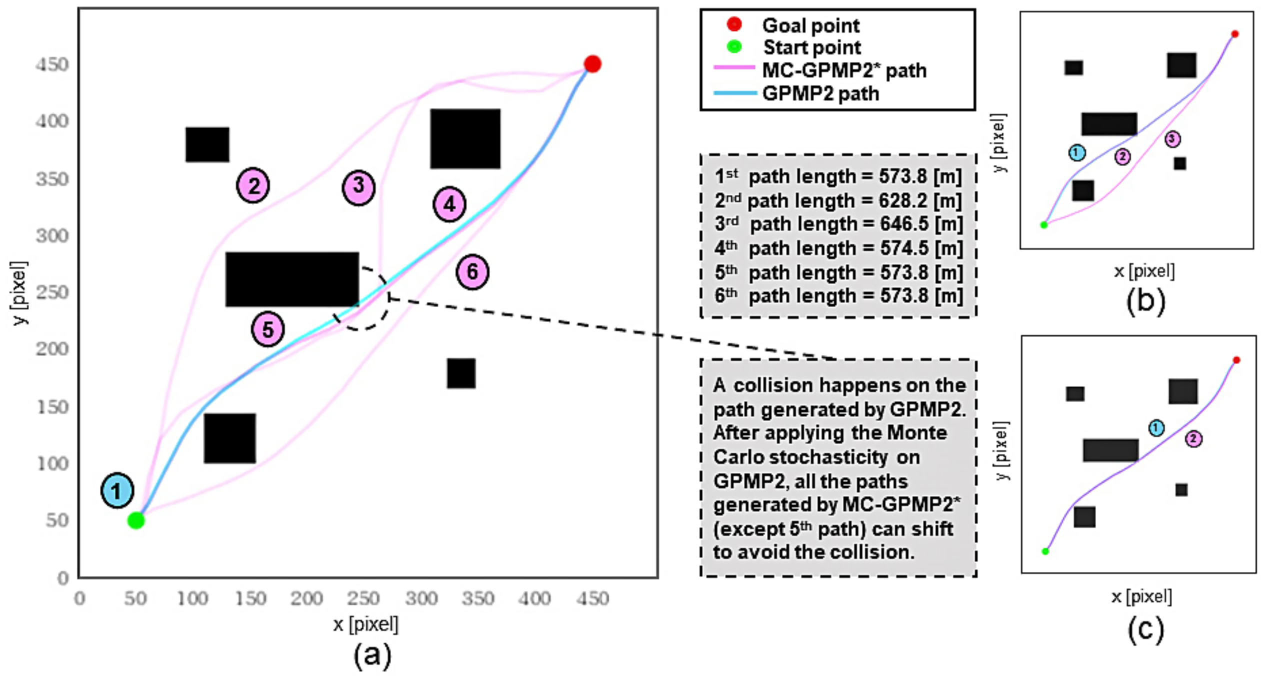

Fig. 4. A comparison of the paths generated by GPMP2 and MC-GPMP2*. GPMP2 generates a single solution, while MC-GPMP2* extends the form of this solution by adding randomness to the sampling process making the paths generated by MC-GPMP2* diversified. This characteristic provides extra solutions to a specified motion planning problem, i.e. increasing the probability of approaching a better path. From (a) - (c), the number of sample points increases gradually and the diversity of the paths generated by MC-GPMP2* decreases accordingly.

A. Mathematical modeling for WAM-V 20 USV

The specifications of the catamaran that will be used are listed in Table I. This catamaran consists of a wave-adaptive structure and two air cushions with thrusters mounted at the back end of each cushion. The thrusters rotate around the Z axis simultaneously to supply different-oriented propulsion within the E-N plane as shown in Fig. 5.

The spatial position state of the catamaran $\overrightarrow{X}$ is considered to be its 2D position (E, N), heading angle ψ, sway velocity v, surge velocity u, yaw rate r, angle of deflection of rudders δ_r and rotational speed of thrusters ω_t as illustrated in Fig. 5. Hence the mathematical model of the USV is expressed as:

$$\dot{\overrightarrow{X}} = \begin{bmatrix} \dot{E} \\ \dot{N} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} u\cos\psi - v\sin\psi \\ u\sin\psi + v\cos\psi \\ r \end{bmatrix}, \tag{27}$$

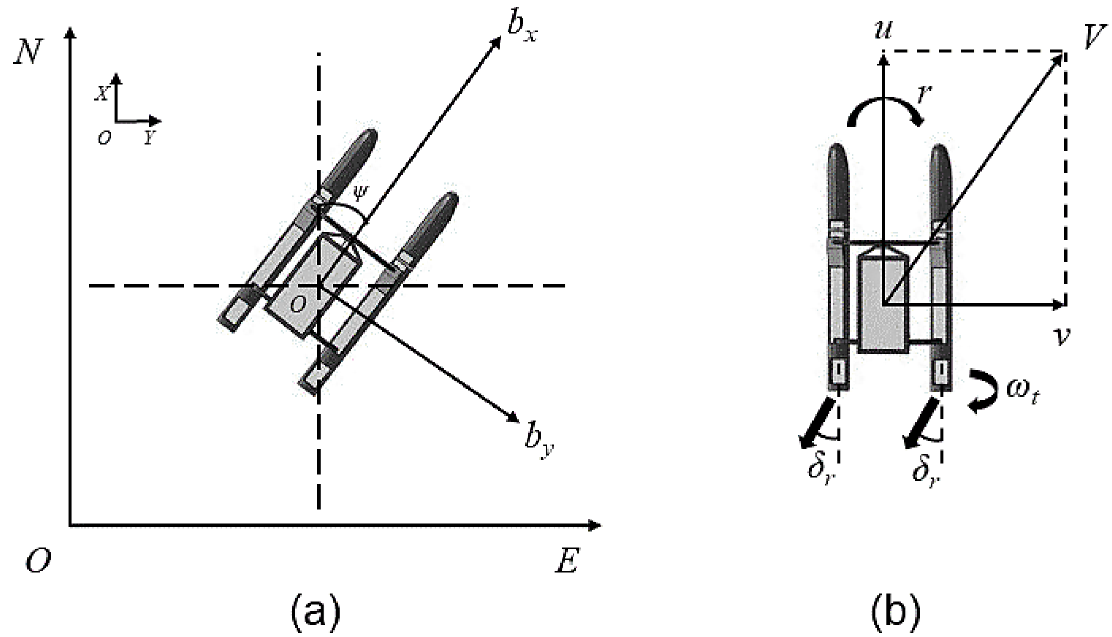

Fig. 5. Schematic depictions of the used catamaran: (a) shows the north-east-down reference frame N $= \{E, N\}$ and the frame attached to the USV platform B $= \{b_x, b_y\}$ and (b) shows the motion diagram of the USV, where r is Z-axis angular velocity (or the USV yaw rate), V is the net velocity of the USV, u is the component of the net velocity on due north (or the USV surge velocity), v is the component of the net velocity on due east (or the USV sway velocity), δ_r is the angle of deflection of rudders and ω_t is the rotational speed of the thrusters.

TABLE I

WAM-V 20 USV specifications [45].

Vehicle Length	Vehicle Width	Vehicle Weight	Maximum Speed
6 [m]	3 [m]	320 [kg]	10 [m/s]

where

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} E_v \sin\psi + N_v \cos\psi \\ E_v \cos\psi + N_v \sin\psi \end{bmatrix}, \tag{28}$$

where E_v is the velocity component of V along due east and N_v is the velocity component of V along due north.

B. Navigation refinement system

The navigation refinement system can provide a timely adjustments for the USV while tracking the desired path based upon the Light-of-sight (LOS) algorithm [46]. The system

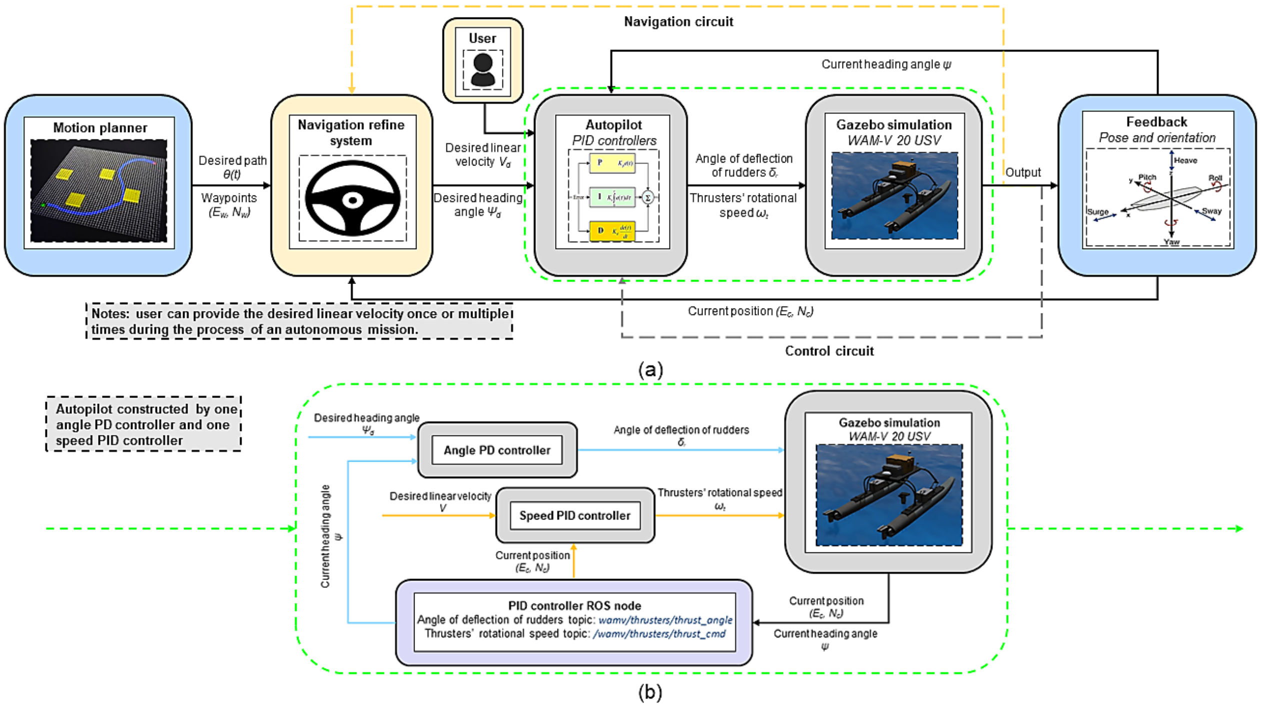

Fig. 6. (a) Overall structure of the proposed fully-autonomous USV framework (b) Detailed structure of the two controllers used in the proposed autopilot. The operator needs only to specify the target goal point and the catamaran's linear speed before the framework starts. This framework then generates a collision-free and smooth path between the current position of the catamaran and any goal point present in the Gazebo environment and makes the catamaran follow this generated path automatically without requiring any operator interaction.

uses the position of the next waypoint and the current position of the USV to determine the required update to the heading angle of the USV. Given the path generated by the motion planner as:

$$\theta(t) = [(E_{w_1}, N_{w_1}), ...(E_{w_n}, N_{w_n})], \tag{29}$$

where (E_{w_1}, N_{w_1}) is the first waypoint on the desired path, (E_{w_i}, N_{w_i}) is the ith waypoint on the desired path, (E_{w_n}, N_{w_n}) is the last waypoint on the desired path and the path generated by the motion planner $\theta(t)$ is a function of time. Hence at a certain moment t, the position of the next desired waypoint (E_w, N_w) can be found.

The reference frame in the Gazebo virtual world is expressed as G $= \{X, Y\}$. As can be seen in Fig. 5, the direction of the X axis in G coincides with the direction of N axis in N and the direction of the Y axis in G coincides with the direction of E axis in N. Furthermore, the rotational angle in N belongs to $(0, 2\pi]$ and the rotational angle in G belongs to $(-\pi, \pi]$. The rotational angles in the G and N reference frames need to be made uniform prior to obtaining the current position of the USV in the Gazebo virtual world. By inputting the position of the next waypoint and the current position of the USV into the navigation refinement system, the next desired heading angle of the USV can be obtained.

C. Autopilot

To track the desired path accurately and smoothly, it is necessary to build a high-performance control mechanism to minimise the deviation between the planned path and the actual path. Based on the mechanical structure of the selected USV, two separate controllers need to be designed as: 1) an angle controller responsible for adjusting the angle of deflection of rudders (or the USV's yaw speed) and 2) a speed controller responsible for adjusting the rotational speed of thrusters (or the USV's linear speed within E-N plane). The overall structure of the proposed fully-autonomous USV navigation control system is detailed in Fig. 6 (a), while the communicating and interfacing arrangement of the controllers used in the proposed autopilot is detailed in Fig. 6 (b). Proportional–integral–derivative (PID) control is used for designing the two controllers as it has been widely adopted in previous practical USV applications [47], [48], [49]. Other types of controllers, such as back-stepping [50], [51], [52] and finite-time path-following [53], can be modified to be used as long as correct ROS messages are communicated. More details regarding the fine-tuned autopilot can be found in the open source library at: https://github.com/jiaweimeng/wam-v-autopilot

1) *Angle PD controller:* It is a PD controller with tuned parameters (P = 1.5 and D = 12.5). This PD controller is used to adjust the USV's rudder angle to match the desired rudder angle according to the waypoints on the desired path. Compared with the standard PID controller, we excluded the integration term as we discovered no explicit steady-state error between the current and the desired rudder angles after turning.

To follow an arbitrary smooth path, the desired rudder angle is one of the controller inputs used to calculate the orientation error:

$$e_{\Delta\psi} = \psi - \psi_d \tag{30}$$

where ψ is the USV's current rudder angle, ψ_d is the desired rudder angle and the ranges of ψ and ψ_d are $(-\pi, \pi]$.

Based on a real-time acquired orientation error, an angle PD controller can then be constructed in the continuous-time domain:

$$\delta_r = k_p \left[e_{\Delta\psi_t}\right] + k_d \left[\frac{d(e_{\Delta\psi_t})}{dt}\right], \tag{31}$$

where k_p and k_d are the PD gains, δ_r is the angle of deflection, $[e_{\Delta\psi_t}]$ is the proportional error and $[\frac{d(e_{\Delta\psi_t})}{dt}]$ is the differential error.

Due to the entire fully-autonomous USV system is built in discrete-time domain, it can then be expressed as:

$$\delta_r = k_p[e_{\Delta\psi_i}] + k_d[(e_{\Delta\psi_i} - e_{\Delta\psi_{i-1}})], \tag{32}$$

where k_p and k_d are the PD gains, δ_r is the angle of deflection, $[e_{\Delta\psi_i}]$ and $[(e_{\Delta\psi_i} - e_{\Delta\psi_{i-1}})]$ are the corresponding proportional error and differential error in the discrete-time domain, respectively.

2) Speed PID controller: It is a PID controller with tuned parameters (P = 2.5, I = 0.05 and D = 1.7). This PID controller is used to adjust the thrusters' rotational speed, hence to match the actual linear velocity of the USV with the desired value according to the user's requirement.

To maintain the actual velocity of the USV just at the level of the desired linear velocity or the user-specified velocity, the actual velocity of the USV is one of the controller inputs used to calculate the velocity error:

$$e_{\Delta V} = V - V_d \tag{33}$$

where V is the actual linear velocity of the USV, V_d is the desired linear velocity and the ranges of them will be described in Section. VI.

Nevertheless, the actual linear velocity of the USV cannot be obtained straightforwardly from the Gazebo simulation environment. Thus we need to measure it through the following equation:

$$V = \frac{\sqrt{(N_c - N_p)^2 + (E_c - E_p)^2}}{\Delta T}, \tag{34}$$

where (E_c, N_c) and (E_p, N_p) are the current position and the previous position of the USV obtained straightforwardly from the Gazebo simulation environment between one system interval period ΔT, respectively.

TABLE II

Specification of the parameters used in the motion planning algorithms.

Map [pixel]	GP-based Motion Planning					A*	RRT*
	ϵ	σ_{obs}	σ_e	$T_{\max}$	N	l	l
500x500	20	0.05	0.005	2.0	5	10.0	10.0
1000x1000	20	0.05	0.005	4.0	10	10.0	10.0
2000x2000	20	0.05	0.005	8.0	20	10.0	10.0

Notes: These parameters are empirically determined as values provide a good trade-off between collision avoidance and energy consumption reduction.

Based on a real-time acquired velocity error, a speed PID controller can then be constructed in the continuous-time domain:

$$\omega_t = k_p\left[e_{\Delta V_t}\right] + k_i\left[\int_0^{t_c}(e_{\Delta V_t})\,dt\right] + k_d\left[\frac{d(e_{\Delta V_t})}{dt}\right], \tag{35}$$

where k_p, k_i and k_d are the PID gains, ω_t is the thrusters' rotational speed of the USV, $[e_{\Delta V_t}]$ is the proportional error, $[\int_0^{T_c}(e_{\Delta V_t})\,dt]$ is the integral error, $[\frac{d(e_{\Delta V_t})}{dt}]$ is the differential error and t_c is the present moment.

Due to the entire fully-autonomous USV system is built based on the discrete-time domain, it can then be expressed as:

$$\omega_t = k_p[e_{\Delta V_i}] + k_i\left[\sum_{n=0}^{T_i} e_{\Delta V_t}\right] + k_d\left[(e_{\Delta V_i} - e_{\Delta V_{i-1}})\right], \tag{36}$$

where k_p, k_i and k_d are the PID gains, ω_t is the thrusters' rotational speed of the USV, $[e_{\Delta\psi_i}]$, $[\sum_{n=0}^{T_i} e_{\Delta V_t}]$ and $[(e_{\Delta\psi_i} - e_{\Delta\psi_{i-1}})]$ are the corresponding proportional error, integral error and differential error in the discrete-time domain, respectively.

V. Simulations and discussions

This section demonstrates the performance of the proposed motion planning algorithm on the basis of comparisons against three simulation benchmarks.

A. Simulation details

Three simulation benchmarks have been conducted to evaluate the proposed MC-GPMP2*. First, the incremental optimisation process of the proposed method was subjected to

TABLE III

Specification of the used hardware platform.

Name of the Device	Description	Quantity
Processor	2.6-GHz Intel Core i7-6700HQ	8
RAM	8 GB	1

qualitative tests. Then the proposed method was quantitatively compared with other state-of-the-art motion planning algorithms including GPMP2 [20], A* (or A star) [2], RRT* (or rapidly-exploring random tree star) [5] and AFM (or anisotropic fast marching method) [54] in different environments both with and without environment characteristics (ocean currents). In all the simulations, GP-based methods were always initialised with a constant-velocity straight-line trajectory. Table II details the specifications of the parameters used in the motion planning algorithms. The specific parameters of GP-based motion planning, A* and RRT* in the following simulations in various resolutions are clarified. In Table II, ϵ indicates the safety distance [pixel], σ_{obs} indicates the obstacle cost weight, σ_e indicates the energy cost weight, $T_{\max}$ indicates the total sampling time [s], N indicates the low-resolution region number in Algorithm 2 and l indicates the step size [pixel]. In the following simulations, one pixel in the map equals one meter in the corresponding motion planning problem. Table III is a specification of the hardware platform used.

B. System dynamics model

Applying a constant-velocity motion model minimises acceleration along the trajectory, thus reducing energy consumption and increasing the smoothness of the generated path. The system dynamics of the robot platform is represented with the double integrator linear system with additional white noise on acceleration. The trajectory is then generated by (20) with the following specific parameters:

$$A = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}, x(t) = \begin{bmatrix} r(t) \\ v(t) \end{bmatrix}, F(t) = \begin{bmatrix} 0 \\ I \end{bmatrix}, u(t) = 0, \tag{37}$$

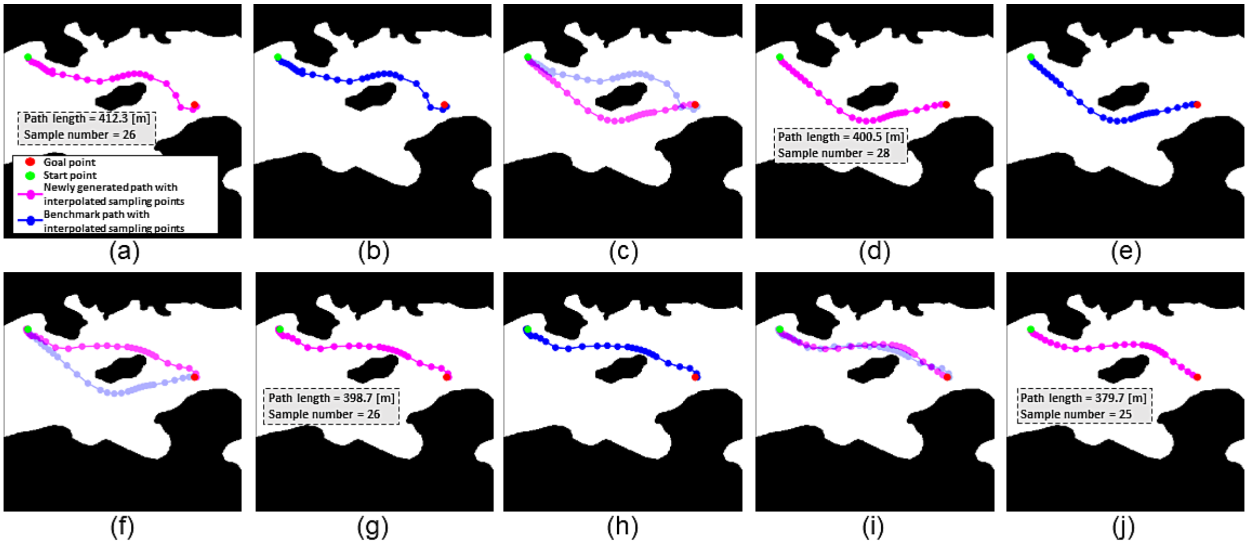

Fig. 7. A demonstration of the incremental optimisation process of MC-GPMP2* in replanning problems: (a) a new path is generated, (b) this newly generated path turns into a benchmark path, (c) this benchmark path is compared with another newly generated path and the latter will be accepted if 1) it is shorter than the former and 2) it does not collide with any obstacle and (d) the newly generated path is accepted. Similar to (a) - (d), (d) - (g) and (g) - (j) repeat the process to achieve incremental optimisation in replanning problems. Overall, the path length generated by MC-GPMP2* decreases from 412.3 [m] to 379.7 [m] within 5 replanning iterations.

where $r = (x, y)^T$ is the position vector, $v = (v_x, v_y)$ is the velocity vector and given $\Delta t_i = t_{i+1} - t_i$,

$$\Phi(t,s) = \begin{bmatrix} I & (t-s)I \\ 0 & I \end{bmatrix}, Q_{i,i+1} = \begin{bmatrix} \frac{1}{3}\Delta t_i^3 Q_C & \frac{1}{2}\Delta t_i^2 Q_C \\ \frac{1}{2}\Delta t_i^2 Q_C & \Delta t_i Q_C \end{bmatrix}, \tag{38}$$

This prior is centred around a zero-acceleration trajectory (or a straight-line segment) [20]. During the optimisation process, the cost function can make the trajectory deviate from the straight-line segment to construct an optimised trajectory.

C. Incremental optimisation process of the proposed method

In this subsection, we demonstrate the incremental optimisation process of the proposed GPMP2* when trajectory replanning is taking place in a coastal region. We explicitly reveal how the Monte-Carlo sampling can adaptively vary the number of sampling points to generate an optimised trajectory. As shown in Fig. 7, by having 5 support states, a new path with 26 sampling points is generated as shown in Fig. 7 (a) with the path length being 412.3 [m]. The number of sampling points between each support state are 7, 4, 3, 8 and 4, respectively. By using this path as a benchmark (Fig. 7 (b)), a new path with 28 sampling points is generated as shown in Fig. 7 (c) with the length being 400.5 [m] and the sampling points between each support state being 7, 4, 3, 10 and 4, respectively. A comparison between the benchmark path and this new path is then conducted. The new path will be accepted if 1) it is shorter than the benchmark path and 2) it does not intersect with any obstacle. Such iterative comparisons will continue until no more new paths are generated and an optimal trajectory can then be selected, which in this case is that shown in Fig. 7 (j).

To summarise, GP-based motion planning generates a trajectory from a stochastic process, where the pattern of the trajectory is determined by the sampling points. Within the conventional GP-based motion planning, such as GPMP2, although an option to adjust the number of sample points is provided, there is a lack of strategy to achieve the optimal number of sample points, forcing most GP-based motion planning algorithms to require manual tuning of the number of sample points. Monte Carlo stochasticity can be added to GP-based motion planning algorithms to achieve an adaptive tuning process by doing the following strategy: within a region with a small number of support states, more states can be interpolated based upon the number of obstacles, i.e. a larger number of sample points would need to be interpolated to deal with a number of densely packed obstacles while reducing the number of points for less densely packed obstacles. By following such a strategy, sampling points can be adjusted and interpolated more effectively and efficiently.

D. Benchmark without environment characteristics

In this subsection, we conduct a comparative study showing the improvement of MC-GPMP2* against the mainstream motion planning algorithms such as GPMP2, A* and RRT*. Various simulation environments are adopted including: 1) a no-obstacle environment, 2) a single-obstacle environment, 3) a multi-obstacle environment, 4) a narrow-passage environment and 5) a coastal environment without any environment characteristics. Note that within the MC-GPMP2*, a relatively large number of sampling points is used to guarantee the generation of optimised trajectories.

The simulation results are shown in Fig. 8 (a) - (e). Note that only the results from the 500 * 500 pixel maps are illustrated as different resolutions mainly affects the computation time rather than the generated trajectories. A quantitative assessment of different algorithms is

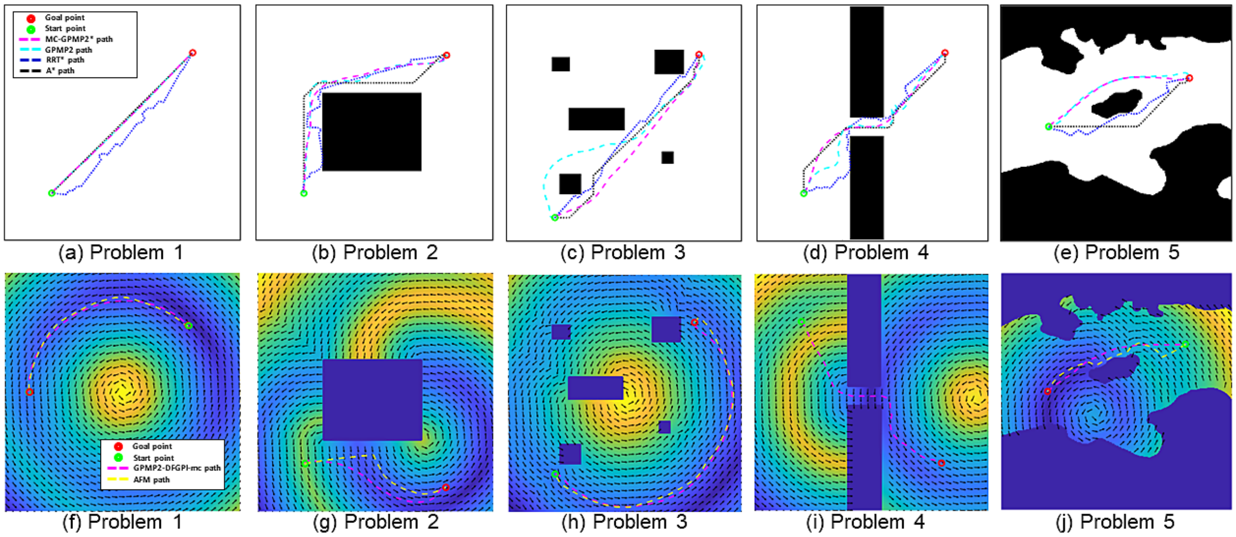

Fig. 8. Comparisons of the paths generated by various motion planning algorithms in different scenarios with and without environment characteristics from 500 * 500 pixel maps: (a) and (f) demonstrate non-obstacle scenario (problem 1), (b) and (g) demonstrate single-obstacle scenario (problem 2), (c) and (h) demonstrate multi-obstacle scenario (problem 3), (d) and (i) demonstrate narrow-passage scenario (problem 4), (e) and (j) demonstrate coastal scenario (problem 5). Furthermore, (a) - (e) have no ocean currents but (f) - (j) have ocean currents.

shown in Table IV, where main evaluation metrics such as execution time and path length are compared.

From Fig. 8 (a) - (e), MC-GPMP2* illustrates a distinct advantage regarding the average path length and path smoothness compared with GPMP2, A* and RRT*. In no-obstacle environments (problem 1), MC-GPMP2*, GPMP2 and A* each generate a straight-line path that connects the start point and goal point simply by the shortest distance. However, RRT* generates a winding path with the longest path length and lowest path smoothness. In single-obstacle environments (problem 2), the paths generated by MC-GPMP2* and GPMP2 are of relatively short length and relatively high smoothness. Compared with the GPMP2 path, the MC-GPMP2* path shows further improvement in both the path length and smoothness. This is a benefit of the proposed interpolation strategy. On the other hand, both the paths generated by A* and RRT* are as smooth and might not be smooth enough to satisfy the system dynamics model of the USV. In multi-obstacle environments (problem 3), MC-GPMP2* and GPMP2 paths demonstrate better path smoothness based upon a comparison with the A* and RRT* paths. However, the GPMP2 path tends to avoid the

TABLE IV

A comparison of MC-GPMP2*, GPMP2, A* and RRT* on average execution time (T) and path length (L) in 15 path planning problems without ocean currents. The improvement on average execution time (T_I) with the Monte Carlo stochasticity was also measured in each path planning problem. The experiment on each path planning problem was tested 5 times to calculate the average value.

Map [pixel]	Problem	MC-GPMP2*			GPMP2		A*		RRT*	
		T [ms]	L [m]	T_I [ms]	T [ms]	L [m]	T [ms]	L [m]	T [ms]	L [m]
500*500	1	202.1	500.8	58.0	283.3	500.8	1847.1	494.9	3261.4	562.1
	2	154.3	607.2	39.9	208.5	618.5	21804.8	617.9	4185.8	656.5
	3	175.7	521.9	49.6	236.5	532.7	15204.6	529.8	3723.1	577.5
	4	208.8	480.2	67.8	292.3	491.2	13430.5	476.9	3633.3	610.2
	5	187.5	234.6	17.3	224.5	245.1	6834.5	283.1	2595.8	322.5
1000*1000	1	214.6	992.1	69.2	306.3	992.1	4307.4	989.9	7343.5	1126.4
	2	207.5	1245.5	49.3	277.9	1267.1	-	-	10388.7	1360.2
	3	214.9	1104.7	57.4	286.3	1119.3	-	-	6736.4	1137.1
	4	230.5	1107.2	80.5	339.2	1120.1	-	-	8366.2	1406.3
	5	233.8	546.8	31.1	287.3	562.7	17434.5	549.7	4823.3	562.6
2000*2000	1	363.1	1981.5	82.6	471.1	1981.5	5748.3	1979.9	15216.2	2275.5
	2	340.9	2432.9	61.3	427.2	2499.6	-	-	22966.2	2665.7
	3	369.8	2201.5	79.6	463.9	2222.7	-	-	19636.8	2273.7
	4	358.5	2028.8	105.8	497.9	2045.5	-	-	18707.3	2383.2
	5	406.6	1178.7	51.6	491.5	1195.3	-	-	11049.9	1452.2

Notes: ”-” means the motion planning algorithm is not applicable in this map as its execution time is more than 30 [s], which is meaningless in practical situations. The proposed method is marked in light green. The shortest execution time (T) and the shortest path length (L) in each problem are marked in light blue. Meanwhile, the improvement on average execution time (T_I) with Monte-Carlo stochasticity in each problem is marked in light yellow. Without Monte-Carlo stochasticity, MC-GPMP2* uses a traversal algorithm to estimate the obstacle space. In this benchmark, all the motion planning algorithms only run once, which means the replanning processes of them are excluded. For instance, the re-wiring process of the tree branches of RRT* will be terminated once a feasible path has been found.

first obstacle sweeping out around the left hand side, leading to a significant increase in the path length. With the proposed interpolation strategy, MC-GPMP2* generates an option that would avoid the first obstacle from the bottom side and this results in a decrease on the path length. Similar to multi-obstacle environments (problem 3), MC-GPMP2* produces a path option which presents a further improvement on length and smoothness compared with the GPMP2 path in narrow-passage environments (problem 4). This is because most of the sampling points of MC-GPMP2* were sampled around the narrow passage to improve

TABLE V

A comparison of MC-GPMP2* and AFM on average energy consumption rate (P), execution time (T) and path length (L) in 15 path planning problems with ocean currents. The improvement on average execution time (T_I) with the Monte Carlo stochasticity was also measured in each path planning problem. The experiment on each path planning problem was tested 5 times to calculate the average value.

Map [pixel]	Problem	MC-GPMP2*				AFM		
		P [%]	T [ms]	L [m]	T_I [ms]	P [%]	T [ms]	L [m]
500*500	1	11.8	684.2	327.2	59.4	10.5	909.6	344.6
	2	4.2	608.0	154.5	47.3	2.1	943.6	167.4
	3	15.8	682.5	463.6	77.6	4.9	815.7	505.3
	4	2.0	646.8	168.6	115.8	-	-	-
	5	5.3	609.2	236.7	51.2	2.7	715.6	258.8
1000*1000	1	12.1	2401.5	658.9	103.1	10.5	2559.8	684.8
	2	4.2	2195.1	309.1	107.4	2.0	2306.5	336.5
	3	14.1	2514.7	946.9	143.6	4.9	2731.2	1014.6
	4	2.0	2181.4	355.6	197.0	-	-	-
	5	5.0	2112.3	480.8	67.4	2.7	2265.3	517.2
2000*2000	1	11.6	10821.3	1306.8	166.9	10.3	11193.4	1364.3
	2	2.2	9263.9	407.9	208.2	1.0	9536.5	442.3
	3	14.9	11504.6	1821.8	222.8	4.9	11576.5	2027.8
	4	2.1	9724.3	730.3	252.5	-	-	-
	5	5.3	9006.2	954.9	155.4	2.7	9245.6	1034.0

Notes: ”-” means the motion planning algorithm is not applicable in this map as its execution time is more than 30 [s], which is meaningless in practical situations. The energy consumption rate (P) caused by ocean currents is computed based upon the metric proposed in AFM [33] as explained in (12). The proposed method is marked in light green. The shortest execution time (T) and the shortest path length (L) in each problem are marked in light blue. Meanwhile, the improvement on average execution time (T_I) with Monte-Carlo stochasticity in each problem is marked in light yellow. Without Monte-Carlo stochasticity, MC-GPMP2* uses traversal algorithm to estimate obstacle space. The replanning process of MC-GPMP2* is excluded in this benchmark.

the option for success of the mission and shorten the length of the path apart from the narrow passage itself. In coastal environments, MC-GPMP2* demonstrates the highest path smoothness and the best obstacle avoidance performance as would be expected.

From Table IV, MC-GPMP2* demonstates an obvious benefit on average execution time and path length over GPMP2, A* and RRT* in maps across a range of resolutions. In most of the large-scale motion planning problems with 1000 * 1000 pixel and 2000 * 2000 pixel maps, A* failed to deliver a feasible solution. This is because the motion planning strategy of A* led to a significant increase in complexity in large-scale motion planning problems. Although RRT* could consistently deliver a feasible solution in all the motion planning problems, the average execution time, path length and path smoothness were not satisfactory as the randomness of its sampling points is too high in the configuration space. Compared with GPMP2*, the interpolation strategy of MC-GPMP2* led to a notable improvement in average execution time and path length simultaneously.

To summarise, MC-GPMP2* can generate a path within the shortest execution time, with highest smoothness and near-optimal path length in almost all the cases and achieve better performance with respect to obstacle avoidance compared with other mainstream motion planning algorithms including GPMP2, A* and RRT*.

E. Benchmark with environment characteristic

In this subsection, we conduct another comparative study showing the improvement of MC-GPMP2* over AFM in the same simulation environments with a supplementary environment characteristic resulting from an ocean current field. The ocean current field is generated by the energy consumption metric proposed in AFM [33].

Simulation results related to this benchmark are illustrated in Fig. 8 (f) - (j). Similar to the previous benchmark, only the results from the 500 * 500 pixel maps are shown. The quantitative assessment of MC-GPMP2* and AFM is shown in Table V, where main evaluation metrics such as energy consumption rate, execution time and path length are compared.

From Fig. 8 (f) - (j), MC-GPMP2* has an obvious advantage regarding the average execution time and path length compared with AFM. Comparatively, AFM has a considerable advantage regarding its average energy consumption rate as it continuously tracks the ocean currents. Nevertheless, this could lead to AFM falling into a local minimum when an obstacle is blocking the continuous ocean currents, such as the motion planning problems in narrow-passage environments (problem 4). MC-GPMP2* generates a path under the interaction of two different fields, namely the signed distance field and the energy consumption field. To be more precise, the signed distance field and the energy consumption field can be obtained by inputting the map in the signed distance function in (10) and the metric that can measure the energy consumption rate at each pixel in (12), respectively. Moreover, the energy consumption field can prevent the occurrence of local minima when avoiding obstacles in the signed distance field. In other words, once MC-GPMP2* has fallen into a local minimum in the signed distance field, the energy consumption field would take it out of that local minimum.

From Table V, the proposed method demonstrates a notable advantage on average execution time and path length over another mainstream method (AFM) in different-resolution maps. For both methods, the energy consumption field is computed based upon the energy consumption metric stated in (12). The energy consumption field is more likely to be historical data recorded by relevant meteorological institutions. As a result, the computation time for generating the simulated energy consumption field can be saved when applying the proposed method in practical cases. This would lead to a remarkable reduction in the execution time as the proportion of the time cost on generating the energy consumption field exceeds 96 [%] in the 2000 * 2000 pixel maps.

To summarise, MC-GPMP2* can consider various environment characteristics during the motion planning process. The path generated by MC-GPMP2* would fit these characteristics as much as possible. Compared with other mainstream motion planning algorithms such as AFM, when the path planning has to adjust for the influence of ocean currents, MC-GPMP2 can generate a path with the shortest execution time, highest smoothness, near-optimal path length and a better performance on obstacle avoidance.

In both the benchmark tests with and without environment characteristics: the Monte-Carlo sampling algorithm can converge much earlier than the traversal algorithm, thereby reducing the time cost of a motion planner with sample points. In these benchmark tests, we only demonstrate the improvement of average execution time in 2D motion planning problems. But in high-dimensional motion planning problems, such as the motion planning problems for multiple degrees of freedom robotic arms, the Monte-Carlo sampling holds the potential to reduce a significant time cost since its convergence rate is independent of the dimension of the configuration space. Hence it solves the problem of dimensional explosion in GP-based motion planning algorithms to some extent.

VI. Implementation in ROS

This section demonstrates the performance of the proposed autonomous navigation system for WAM-V 20 USVs. Two different motion planning algorithms, i.e. RRT* and the proposed

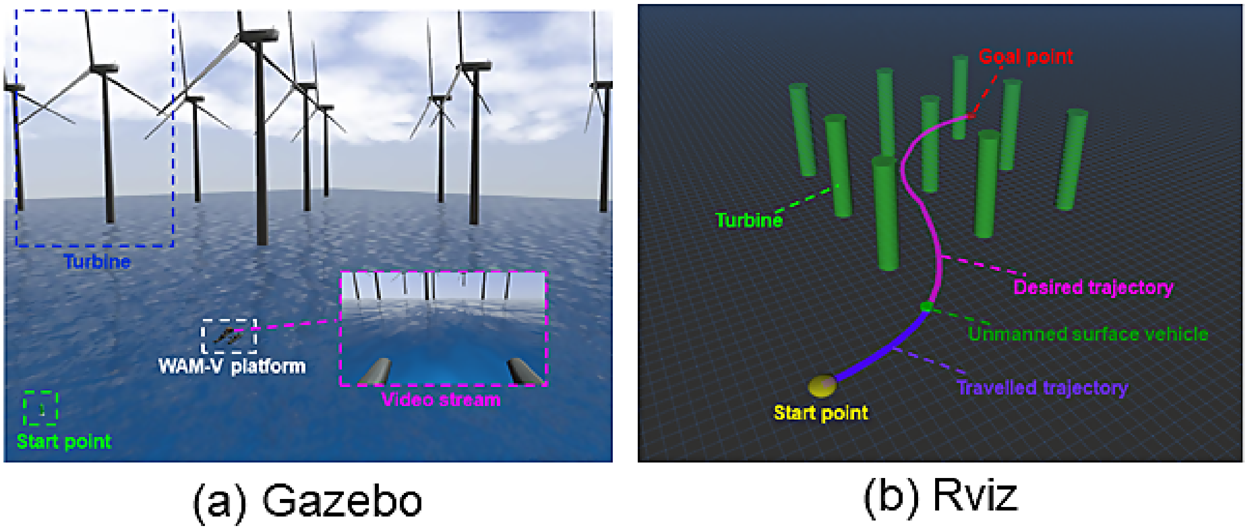

Fig. 9. ROS simulation environment: (a) demonstrates the Gazebo virtual world, where the green dash line block represents the start point, the white dash line block represents the selected platform, the blue dash line block represents the wind turbine (obstacle) and the purple dash line block represents the captured video information from the camera mounted at the front end of the platform and (b) demonstrates the corresponding motion planning problem solved by MC-GPMP2* in Rviz, where the start point is represented in yellow, the goal point is represented in red, the obstacles are represented in green, the desired path is represented in purple and the travelled path is presented in dark blue. In the Gazebo virtual world, wind and wave fields can be adjusted by changing the corresponding parameters to create a realistic simulation environment.

MC-GPMP2*, are implemented and compared. An offshore wind farm inspection mission is simulated in ROS to show the practicability of the proposed work.

A. Simulation details

The detailed information of the environment used in the ROS simulation is detailed in Fig. 9, where (a) shows the offshore wind farm in Gazebo with the inclusion of a series of physical properties such as sunlight, wind, ocean currents, gravity and buoyancy, (b) provides a simulation overview of the configuration space of the corresponding motion planning problem in Rviz. A green buoy and a red buoy are placed inside the simulation environment to indicate the start point and the goal point for the route proposed for the WAM-V 20 USV to navigate. The platform was equipped with a camera to better observe the surrounding environment and record videos. The footage from the camera was streamed to and displayed on the Rviz interface through the WAM-V Camera node (/wam-v/sensors/cameras/front-camera/image-raw). The virtual onboard camera gives this

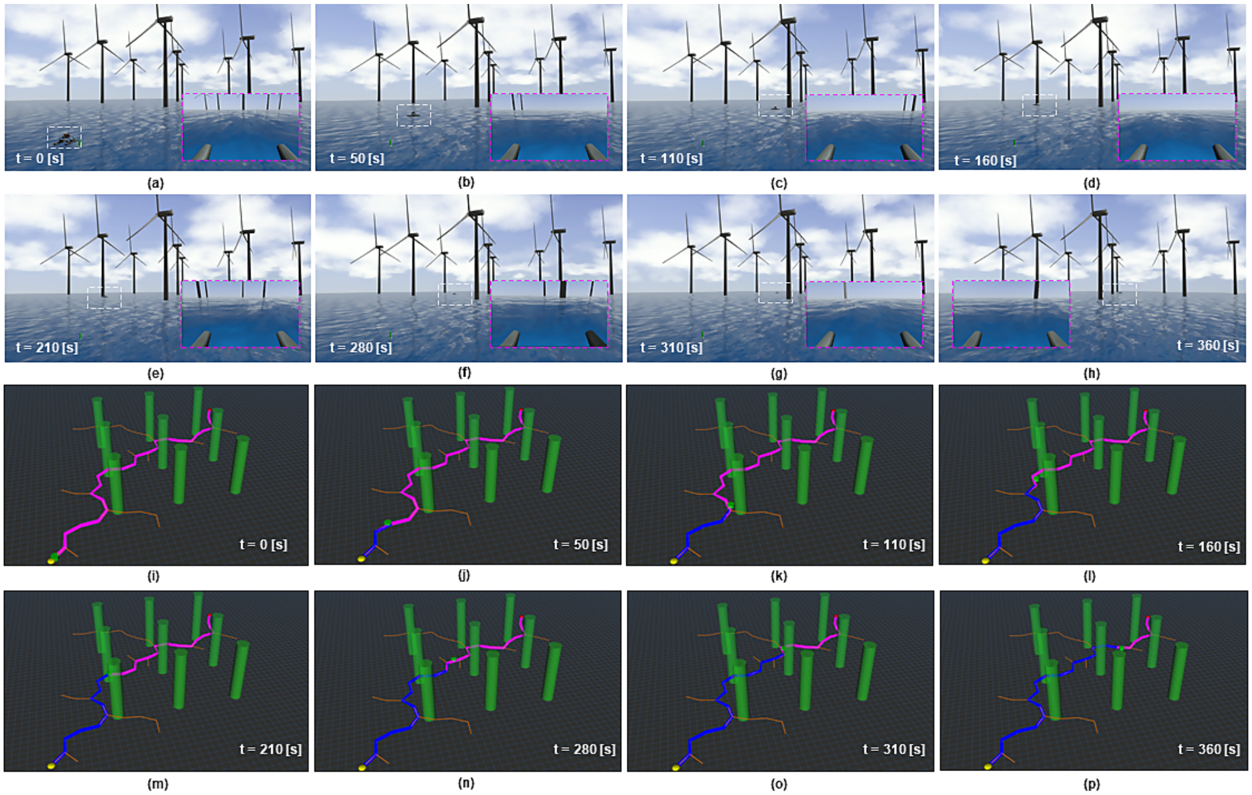

Fig. 10. The storyboards of the inspection mission in an offshore wind power generation scenario based on a path generated by RRT*: From (a) to (h), the images demonstrate the location of the platform and the video stream from the camera mounted at the front end of the platform when the time equals 0 [s], 50 [s], 110 [s], 160 [s], 210 [s], 280 [s], 310 [s] and 360 [s], respectively. From (i) to (h), the images demonstrate the corresponding motion planning problem in Rviz when the time equals 0 [s], 50 [s], 110 [s], 160 [s], 210 [s], 280 [s], 310 [s] and 360 [s], respectively.

work the potential to combine with previous research done by our research group on using onboard cameras for object detection and segmentation in maritime environments [55], [56].

During the inspection mission in Gazebo, the USV transited through the wind turbine area to drive away any fish boats entering this area to reduce risk of collision and damage to the wind turbines. Figs. 10 and 12 demonstrate the storyboards of the inspection mission from both the first-person and third-person perspectives in the Gazebo as well as the motion planning problem solved by the corresponding motion planning algorithms in Rviz.

In the ROS simulation, the inspection mission is designed based on the following steps:

- Simulation information is inputted as: 1) a Green Buoy Model State node (/gazebo/model-states/green_buoy) reads the location of the green buoy which represents the start point, 2) a Red Buoy Model State node (/gazebo/model-states/red_buoy) reads the

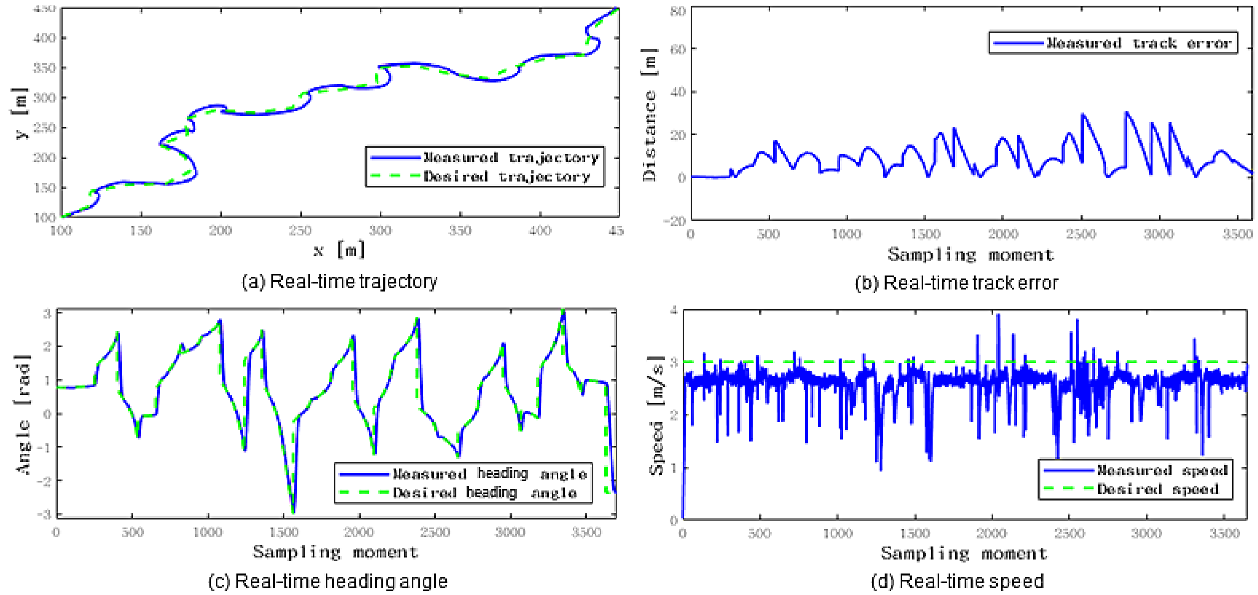

Fig. 11. Performance analysis of the inspection mission in an offshore wind power generation scenario based on a path generated by RRT*: (a) compares the desired and measured trajectories, (b) demonstrates the real-time track error, (c) compares the desired and measured heading angles and (d) compares the desired and measured speeds.

location of the red buoy which represents the goal point, 3) a WAM-V Model State node (/gazebo/model-states/wam-v) reads the current pose of the WAM-V 20 USV, 4) the Wind Turbines Model State node (/gazebo/model-states/turbines) reads the locations of the wind turbines, 5) the Ocean Currents State node (/gazebo/model-states/ocean-currents) reads the information regarding the ocean currents and 6) a WAM-V Thrusters State node (/wam-v/thrusters) reads the angle of deflection of rudders δ_r and thrusters' rotational speed ω_t in the Gazebo.

- This information is then transmitted and used to generate start point, goal point and obstacles in the Rviz. The motion planning algorithm then generates a desired path $\theta(t)$ with a series of waypoints (E_w, N_w) based on the information in Rviz.
- The waypoints (E_w, N_w) are transmitted to Gazebo and the platform begins tracking the planned path $\theta(t)$ according to the desired heading ψ_d and the desired linear velocity V_d.
- The autopilot calculates and regulates the angle of deflection of rudders δ_r and thrusters' rotational speed ω_t of the platform in Gazebo in real-time according to the desired heading angle ψ_d and linear velocity V_d. The autopilot makes the platform fulfill the

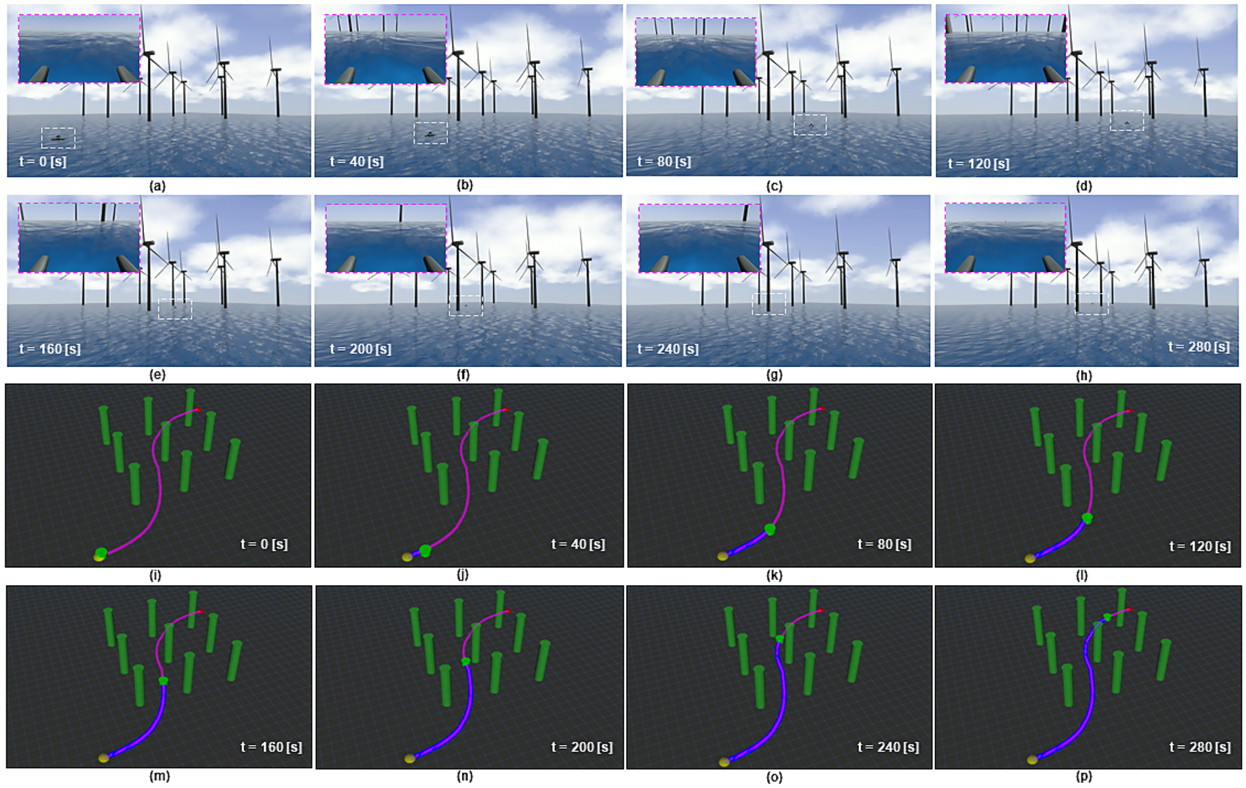

Fig. 12. The storyboards of the inspection mission in an offshore wind power generation scenario based on a path generated by MC-GPMP2*: From (a) to (h), the images demonstrate the location of the platform and the video stream from the camera mounted at the front end of the platform when the time equals 0 [s], 40 [s], 80 [s], 120 [s], 160 [s] and 200 [s], 240 [s] and 280 [s], respectively. From (i) to (p), the images demonstrate the corresponding motion planning problem in Rviz when the time equals 0 [s], 40 [s], 80 [s], 120 [s], 160 [s], 200 [s], 240 [s] and 280 [s], respectively.

motion constraint such as the pose and orientation of the desired path $\theta(t)$. It is worth noting that due to vehicle inertia, the USV would keep moving forward after reaching the target waypoint (E_{w_i}, N_{w_i}). In order to minimise the effects of inertia, the platform is considered to have reached the target waypoint (E_{w_i}, N_{w_i}) once it is inside a certain range (7 [m] in this case) of the waypoint.

- The platform enters the standby mode once it reaches the last target waypoint (E_{w_n}, N_{w_n}) of the desired path $\theta(t)$.

B. Performance analysis

Performance analysis of the proposed autonomous navigation systems using different motion planning algorithms is detailed in Figs. 11 and 13. In general, motion planning

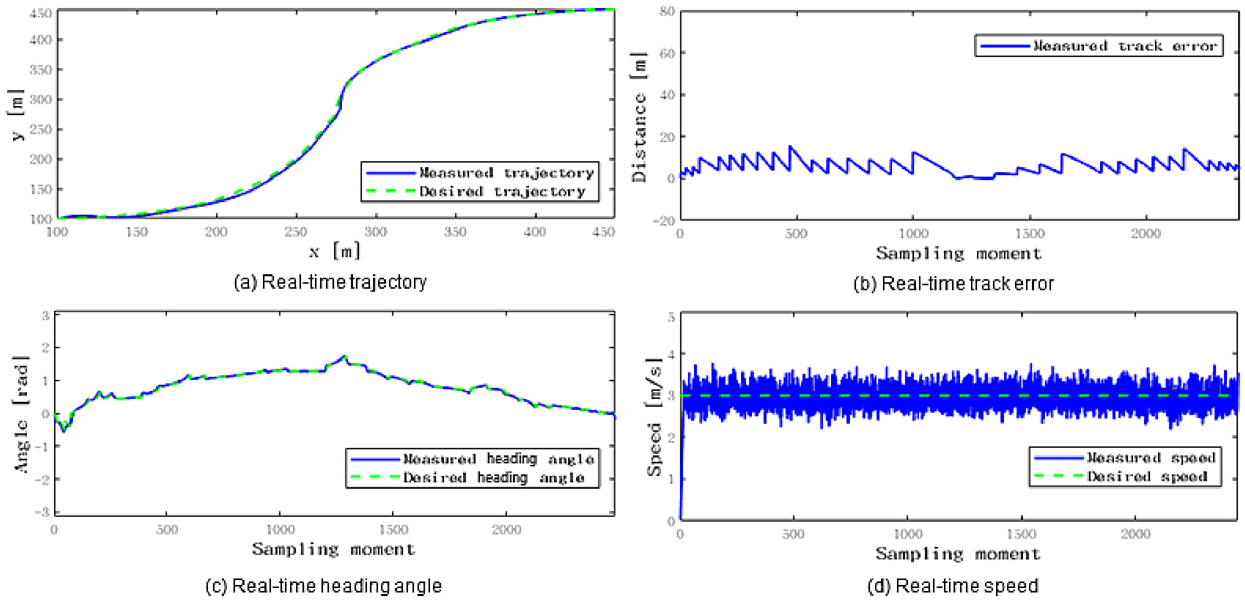

Fig. 13. Performance analysis of the inspection mission in an offshore wind power generation scenario based on a path generated by MC-GPMP2*: (a) compares the desired and measured trajectories, (b) demonstrates the real-time track error, (c) compares the desired and measured heading angles and (d) compares the desired and measured speeds.

algorithms (such as RRT* and MC-GPMP2*) can be integrated into the proposed navigation system, within which a good trajectory tracking performance is achieved. Noticeably, as shown in Fig. 10 and Fig. 12, MC-GPMP2* generates much smoother path than RRT*, which leads to reduced tracking error as shown in Figs. 11 (b) and 13 (b).

Improved path smoothness can also lead to less severe control inputs and potentially improve the stability of the USV. For example, by comparing the heading angles and speeds in Figs. 11 (c), (d) and Figs. 13 (c) and (d), a more gradual variation, especially in heading angle, is experienced by following the trajectory provided by MC-GPMP2* as opposed to the dramatic change between positive and negative maximum values for RRT* trajectories. Such a benefit makes the proposed MC-GPMP* a more viable solution for USVs, especially when operating in constrained areas requiring refined motion planning capability.

VII. Conclusion and future work

This paper introduced an improved version of the conventional GP-based motion planning algorithm (GPMP2) by further discussing the form of the likelihood function in probabilistic inference. The improved version GPMP2* extends the application scope of GPMP2 from environments with only obstacles into complex environments with a variety of environment characteristics. Further, a novel fast GP interpolation strategy with Monte-Carlo stochasticity has been added into GPMP2*, constructing another improved version named MC-GPMP2*. MC-GPMP2* can enhance the diversity in the generated path while reducing the time cost of manually tuning sampling points. Then a fully-autonomous framework has been proposed for a mainstream catamaran (WAM-V 20 USV). This framework contains an interface for any motion planner and an efficient, open-source autopilot. In four different simulations, we first demonstrated the path diversity of MC-GPMP2* and its incremental optimisation process in replanning problems. The performance of MC-GPMP2* was then compared with other mainstream motion planning algorithms such as GPMP2, A* and RRT* across a range of environments with obstacles. MC-GPMP2* generated paths with the shortest execution time, highest path smoothness and shortest path lengths in almost all cases. A competitive study was then conducted between MC-GPMP2* and a mainstream motion planning algorithm in environments with ocean currents (AFM). The results demonstrated that MC-GPMP2* delivers a better performance compared with AFM in execution time, path length and path smoothness in all the cases. Finally, we compared the performances of MC-GPMP2* and RRT* in an inspection mission based on WAM-V 20 USV and the proposed framework in a high-fidelity virtual world. The results further reinforced the remarkable performance of MC-GPMP2* in practical autonomous missions as well as reflected the accuracy and effectiveness of the proposed USV navigation and control framework.

In terms of future work, proposed areas of focus are: 1) validating the improvement of MC-GPMP2* over other mainstream algorithms in high-dimensional environments, such as the motion planning circumstances of UUVs or robotic arms, 2) enriching the autopilot repository by adding other mainstream controllers such as back-stepping and finite-time path following, 3) automatically tuning the weight coefficients ω_1 and ω_2 in the objective function in (3) by using learning-based algorithms, 4) developing another motion planner that can use multiple USVs simultaneously to inspect the offshore wind power generation scenario and 5) using a digital twin for the navigation of USV so that simulation and real environment both can be benchmarked against each other.

References

[1] E. W. Dijkstra et al., "A note on two problems in connexion with graphs," Numerische mathematik, vol. 1, no. 1, pp. 269–271, 1959.

[2] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," IEEE transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968.

[3] A. Stentz, "Optimal and efficient path planning for partially known environments," in Intelligent unmanned ground vehicles. Springer, 1997, pp. 203–220.

[4] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," IEEE transactions on Robotics and Automation, vol. 12, no. 4, pp. 566–580, 1996.

[5] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[6] J. Meng, V. M. Pawar, S. Kay, and A. Li, "Uav path planning system based on 3d informed rrt for dynamic obstacle avoidance," in 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2018, pp. 1653–1658.

[7] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in Autonomous robot vehicles. Springer, 1986, pp. 396–404.

[8] C. Petres, Y. Pailhas, Y. Petillot, and D. Lane, "Underwater path planing using fast marching algorithms," in Europe Oceans 2005, vol. 2. IEEE, 2005, pp. 814–819.

[9] M. Dorigo, A. Colorni, and V. Maniezzo, "Distributed optimization by ant colonies," 1991.

[10] D. Whitley, "A genetic algorithm tutorial," Statistics and computing, vol. 4, no. 2, pp. 65–85, 1994.

[11] S. MahmoudZadeh, D. M. Powers, and A. M. Yazdani, "A novel efficient task-assign route planning method for auv guidance in a dynamic cluttered environment," in 2016 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2016, pp. 678–684.

[12] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, and D. Lane, "Path planning for autonomous underwater vehicles," IEEE Transactions on Robotics, vol. 23, no. 2, pp. 331–341, 2007.

[13] T. Lolla, P. Haley Jr, and P. Lermusiaux, "Path planning in multi-scale ocean flows: Coordination and dynamic obstacles," Ocean Modelling, vol. 94, pp. 46–66, 2015.

[14] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," 1988.

[15] S. Garrido, L. Moreno, and D. Blanco, "Exploration of a cluttered environment using voronoi transform and fast marching," Robotics and Autonomous Systems, vol. 56, no. 12, pp. 1069–1081, 2008.

[16] Y. Singh, S. Sharma, D. Hatton, and R. Sutton, "Optimal path planning of unmanned surface vehicles," 2018.

[17] M. Mukadam, X. Yan, and B. Boots, "Gaussian process motion planning," in 2016 IEEE international conference on robotics and automation (ICRA). IEEE, 2016, pp. 9–15.

[18] T. D. Barfoot, C. H. Tong, and S. Särkkä, "Batch continuous-time trajectory estimation as exactly sparse gaussian process regression." in Robotics: Science and Systems, vol. 10. Citeseer, 2014.

[19] X. Yan, V. Indelman, and B. Boots, "Incremental sparse GP regression for continuous-time trajectory estimation and mapping," Robotics and Autonomous Systems, vol. 87, pp. 120–132, 2017.

[20] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using gaussian processes and factor graphs." in Robotics: Science and Systems, vol. 12, 2016, p. 4.

[21] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, “Continuous-time gaussian process motion planning via probabilistic inference,” The International Journal of Robotics Research, vol. 37, no. 11, pp. 1319–1340, 2018.

[22] F. Dellaert, “Factor graphs and gtsam: A hands-on introduction,” Georgia Institute of Technology, Tech. Rep., 2012.

[23] Unity-Game Engine, 2021. [Online]. Available: https://unity.com/

[24] Unreal Engine, 2021. [Online]. Available: https://www.unrealengine.com/

[25] Gazebo, 2021. [Online]. Available: http://gazebosim.org/

[26] C. E. Agüero, N. Koenig, I. Chen, H. Boyer, S. Peters, J. Hsu, B. Gerkey, S. Paepcke, J. L. Rivero, J. Manzo et al., “Inside the virtual robotics challenge: Simulating real-time robotic disaster response,” IEEE Transactions on Automation Science and Engineering, vol. 12, no. 2, pp. 494–506, 2015.

[27] M. Allan, U. Wong, P. M. Furlong, A. Rogg, S. McMichael, T. Welsh, I. Chen, S. Peters, B. Gerkey, M. Quigley et al., “Planetary rover simulation for lunar exploration missions,” in 2019 IEEE Aerospace Conference. IEEE, 2019, pp. 1–19.

[28] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, “Rotors—a modular gazebo mav simulator framework,” in Robot operating system (ROS). Springer, 2016, pp. 595–625.

[29] VRX Simulator, 2021. [Online]. Available: https://github.com/osrf/vrx

[30] J. Ko and D. Fox, “Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models,” Autonomous Robots, vol. 27, no. 1, pp. 75–90, 2009.

[31] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, “Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs,” in 2015 IEEE international conference on robotics and automation (ICRA). IEEE, 2015, pp. 3067–3074.

[32] J. Meng, Y. Liu, R. Bucknall, W. Guo, and Z. Ji, “Anisotropic gpmp2: a fast continuous-time gaussian processes based motion planner for unmanned surface vehicles in environments with ocean currents,” IEEE Transactions on Automation Science and Engineering, 2022.

[33] R. Song, Y. Liu, and R. Bucknall, “A multi-layered fast marching method for unmanned surface vehicle path planning in a time-variant maritime environment,” Ocean Engineering, vol. 129, pp. 301–317, 2017.

[34] N. Metropolis and S. Ulam, “The monte carlo method,” Journal of the American statistical association, vol. 44, no. 247, pp. 335–341, 1949.

[35] R. Eckhardt, “Stan ulam, john von neumann, and the monte carlo method,” Los Alamos Science, vol. 15, no. 30, pp. 131–136, 1987.

[36] Definition of law of large numbers on wikipedia, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Law_of_large_numbers

[37] G. Grimmett and D. Stirzaker, Probability and random processes. Oxford university press, 2020.

[38] R. Durrett, Probability: theory and examples. Cambridge university press, 2019, vol. 49.

[39] S. Asmussen and P. W. Glynn, Stochastic simulation: algorithms and analysis. Springer Science & Business Media, 2007, vol. 57.

[40] F. Dellaert and M. Kaess, “Square root sam: Simultaneous localization and mapping via square root information smoothing,” The International Journal of Robotics Research, vol. 25, no. 12, pp. 1181–1203, 2006.

[41] M. Kaess, A. Ranganathan, and F. Dellaert, “isam: Incremental smoothing and mapping,” IEEE Transactions on Robotics, vol. 24, no. 6, pp. 1365–1378, 2008.

[42] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping using the bayes tree," The International Journal of Robotics Research, vol. 31, no. 2, pp. 216–235, 2012.

[43] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in 2011 IEEE International Conference on Robotics and Automation. IEEE, 2011, pp. 3281–3288.

[44] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," Quarterly of applied mathematics, vol. 2, no. 2, pp. 164–168, 1944.

[45] WAM-V 20, 2020. [Online]. Available: http://www.wam-v.com/wam-v-20-asv

[46] T. I. Fossen, M. Breivik, and R. Skjetne, "Line-of-sight path following of underactuated marine craft," IFAC proceedings volumes, vol. 36, no. 21, pp. 211–216, 2003.

[47] Y. Liu, R. Bucknall, and X. Zhang, "The fast marching method based intelligent navigation of an unmanned surface vehicle," Ocean Engineering, vol. 142, pp. 363–376, 2017.

[48] R. Song, Y. Liu, and R. Bucknall, "Smoothed a* algorithm for practical unmanned surface vehicle path planning," Applied Ocean Research, vol. 83, pp. 9–20, 2019.

[49] W. B. Klinger, I. Bertaska, J. Alvarez, and K. D. von Ellenrieder, "Controller design challenges for waterjet propelled unmanned surface vehicles with uncertain drag and mass properties," in 2013 OCEANS-San Diego. IEEE, 2013, pp. 1–7.

[50] W. Zhou, Y. Wang, C. K. Ahn, J. Cheng, and C. Chen, "Adaptive fuzzy backstepping-based formation control of unmanned surface vehicles with unknown model nonlinearity and actuator saturation," IEEE Transactions on Vehicular Technology, vol. 69, no. 12, pp. 14 749–14 764, 2020.

[51] W. B. Klinger, I. R. Bertaska, and K. D. von Ellenrieder, "Experimental testing of an adaptive controller for usvs with uncertain displacement and drag," in 2014 Oceans-St. John's. IEEE, 2014, pp. 1–10.

[52] W. B. Klinger, I. R. Bertaska, K. D. von Ellenrieder, and M. R. Dhanak, "Control of an unmanned surface vehicle with uncertain displacement and drag," IEEE Journal of Oceanic Engineering, vol. 42, no. 2, pp. 458–476, 2016.

[53] N. Wang, Z. Sun, Y. Jiao, and G. Han, "Surge-heading guidance-based finite-time path following of underactuated marine vehicles," IEEE Transactions on vehicular Technology, vol. 68, no. 9, pp. 8523–8532, 2019.

[54] Q. Lin, "Enhancement, extraction, and visualization of 3d volume data," Ph.D. dissertation, Linköping University Electronic Press, 2003.

[55] L. Yao, D. Kanoulas, Z. Ji, and Y. Liu, "Shorelinenet: an efficient deep learning approach for shoreline semantic segmentation for unmanned surface vehicles," in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2021, pp. 5403–5409.

[56] X. Chen, Y. Liu, and K. Achuthan, "Wodis: Water obstacle detection network based on image segmentation for autonomous surface vehicles in maritime environments," IEEE Transactions on Instrumentation and Measurement, vol. 70, pp. 1–13, 2021.

[57] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," The International Journal of Robotics Research, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[58] T. I. Fossen, "Guidance and control of ocean vehicles," University of Trondheim, Norway, Printed by John Wiley & Sons, Chichester, England, ISBN: 0 471 94113 1, Doctors Thesis, 1999.

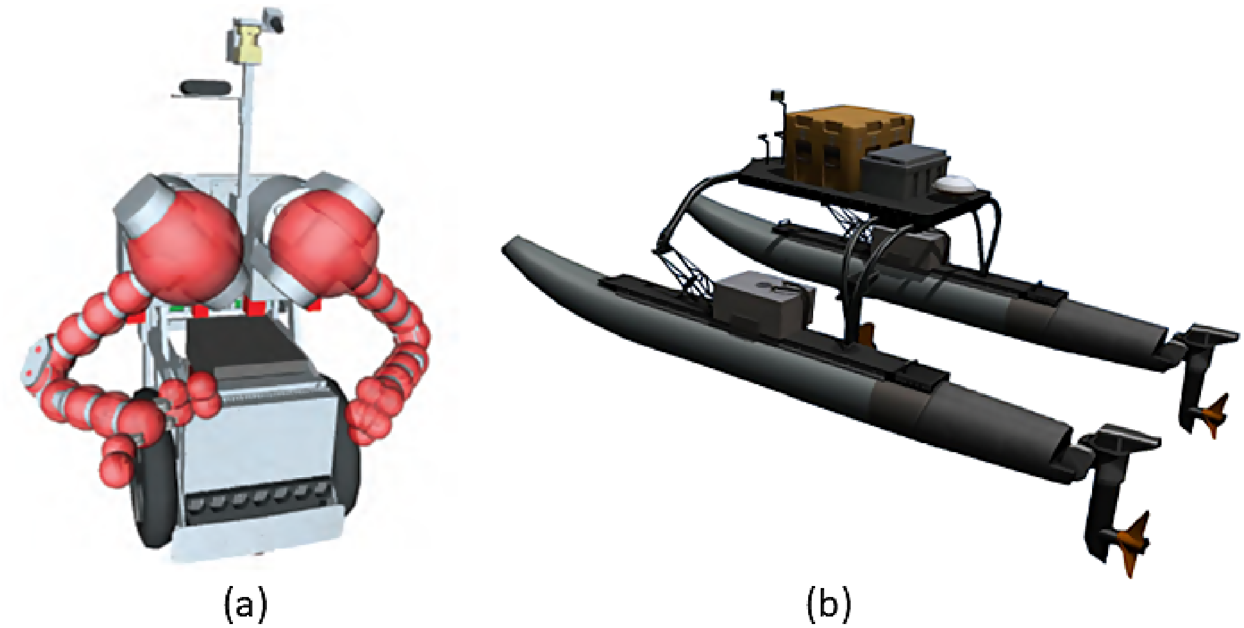

Fig. 14. A demonstration of two different robot platforms in our problem: (a) illustrates the Herb robot [57] and (b) illustrates the WAM-V 20 USV [29].

[59] ——, "Nonlinear modelling and control of underwater vehicles," Ph.D. dissertation, Universitetet i Trondheim (Norway), 1991.

[60] T. I. Fossen and O.-E. Fjellstad, "Nonlinear modelling of marine vehicles in 6 degrees of freedom," Mathematical Modelling of Systems, vol. 1, no. 1, pp. 17–27, 1995.

[61] B. Bingham, C. Agüero, M. McCarrin, J. Klamo, J. Malia, K. Allen, T. Lum, M. Rawson, and R. Waqar, "Toward maritime robotic simulation in gazebo," in OCEANS 2019 MTS/IEEE SEATTLE. IEEE, 2019, pp. 1–10.

Appendix A
Modeling the robots and obstacles in MC-GPMP2*

Fig. 14 provides a more intuitive perspective about the representation of different robot models in (10). To make the optimisation problem tractable, we simply view: 1) the robot model of the robotic arm as a series of spheres over the links and 2) the robot model of the catamaran as a rigid body. Consequently, (10) can be simplified as follows when we apply the catamaran as the robot platform in our motion planning problem:

$$g_1(\theta_i) = [c(d(\theta_i))], \tag{39}$$

where $c(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is the workspace cost function that penalises the set of points $B \subset \mathbb{R}^n$ on the robot body when they are in or around an obstacle and $d(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is the signed distance function that calculates the signed distance of the point. Further, the signed distance function $d(\cdot)$ is defined by the following equations:

$$d(\cdot) = D(\cdot) - \overline{D}(\cdot), \tag{40}$$

where $D(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is the Euclidean distance transforms function and is also named as distance field. $\overline{D}(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is the complement of distance field $D(\cdot)$. Based upon the definition in (40), $d(\cdot)$ allows us to easily distinguish if a point is inside or outside of the obstacles. More specifically, the signed distance function: 1) generates a positive result if the point is located inside the obstacles, 2) equals to zero if the point is located on the boundaries of the obstacles and 3) generates a negative result if the point is located outside the obstacles.

Appendix B
Density of interpolated states in GPMP2

Based upon the information in Section. III-B in our previous research [32], we know that GPMP2 only interested in the collision-free event ($l(\theta; c_i = 0)$). This indicates that the waypoints generated by GPMP2 are always located outside the obstacle areas to obey this rule. Whereas, the density of the interpolated states can influence the length of the line segment between two neighbour waypoints. In general, a longer line segment can increase the possibility of overlapping with obstacles as demonstrated in Fig. 15. To address this problem, we propose MC-GPMP2* to increase the diversity of the generated paths as well as select an appropriate number of interpolated states to ensure all the line segments between the neighbour waypoints do not overlap with any obstacle.

Appendix C
USV dynamic model in ROS environment

As mentioned earlier in the introduction part of this article, we propose a fully-autonomous framework for USVs based upon the VRX simulator that is originally designed in [29]. In this simulator, Fossen's six degrees of freedom robot-like vectorial model for marine craft [58], [59], [60] has been applied in Gazebo to express the dynamic model of the USV:

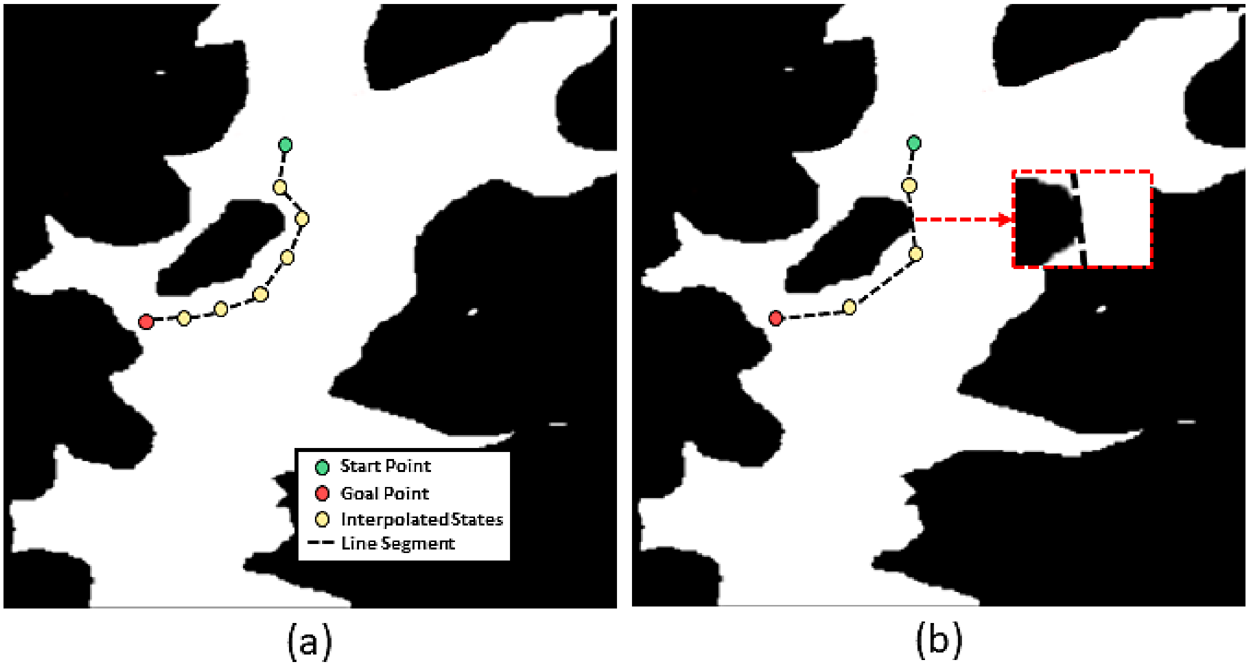

Fig. 15. The effect of the density of interpolated states in GPMP2: (a) illustrates the trajectory generated with relatively high density on the interpolated states and (b) illustrates the trajectory generated with relatively low density on the interpolated states. In (b), the line segment between the second and third waypoints overlaps with the obstacle at the centre (a zoom-in view is provided in the red square region).

$$\underbrace{M_{RB}\dot{v} + C_{RB}(v)v}_{\text{rigid body forces}} + \underbrace{M_A\dot{v}_r + C_A(v_r)v_r + D(v_r)v_r}_{\text{hydrodynamic forces}} \tag{41}$$

$$+ \underbrace{g(\eta)}_{\text{hydrostatic forces}} \tag{42}$$

$$= \tau_{\text{propulsion}} + \tau_{\text{wind}} + \tau_{\text{waves}}, \tag{43}$$

where

$$\eta = [x, y, z, \phi, \theta, \psi]^T \tag{44}$$

$$v = [u, v, \omega, p, q, r]^T, \tag{45}$$

are position and velocity vectors respectively for surge, sway, heave, roll, pitch and yaw. To be more specific, the total velocity (v) is the sum of an irrational water current velocity (v_c) and the vessel velocity relative to the fluid (v_r). The forces and moments due to propulsion (or the control input), wind and waves are represented as $\tau_{\text{propulsion}}$, τ_{wind} and τ_{waves}. Generally, the hydrodynamic forces, hydrostatic and wave forces, wind forces and propulsion forces function on the USV simultaneously in Gazebo [61].