Received 11 April 2023, accepted 6 May 2023, date of publication 11 May 2023, date of current version 19 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3275433

RESEARCH ARTICLE

Trajectory-Free Motion Planning of an Unmanned Surface Vehicle Based on MPC and Sparse Neighborhood Graph

SİMAY ATASOY[1,2], (Member, IEEE), OSMAN KAAN KARAGÖZ[1,3], (Member, IEEE), AND MUSTAFA MERT ANKARALI[1,3], (Member, IEEE)
[1]Department of Electrical and Electronics Engineering, Middle East Technical University, 06800 Ankara, Turkey
[2]ASELSAN Research Center, 06200 Ankara, Turkey
[3]Center for Robotics and AI, Middle East Technical University, 06800 Ankara, Turkey

Corresponding author: Simay Atasoy (simay.atasoy@metu.edu.tr)

This work was supported in part by Aselsan Inc., and in part by the Scientific and Technological Research Council of Turkey (TUBITAK) 1001 Program under Project 118E195 (initial phases) and Project 122E249 (final phases).

ABSTRACT Unmanned Surface Vehicles (USV) have gained significant attention in military, science, and research applications in recent years. The development of new USV systems and increased application domain of these platforms has necessitated the development of new motion planning methods to improve the autonomy level of USVs and provide safe and robust navigation across unpredictable marine environments. This study proposes a feedback motion planning and control methodology for dynamic fully-and under-actuated USV models built on the recently introduced sparse random neighborhood graphs and constrained nonlinear Model Predictive Control (MPC). This approach employs a feedback motion planning strategy based on sparsely connected obstacle-free regions and the sequential composition of MPC policies. The algorithm generates a sparse neighborhood graph consisting of connected rectangular zones in the discrete planning phase. Inside each node (rectangular region), an MPC-based online feedback control policy funnels the USV with nonlinear dynamics from one rectangle to the other in the network, ensuring no constraint violation on state and input variables occurs. We systematically test the proposed algorithms in different simulation scenarios, including an extreme actuator noise scenario, to test the algorithm's validity, effectiveness, and robustness.

INDEX TERMS Nonlinear model predictive control, feedback motion planning, sampling-based motion planning, unmanned surface vehicles.

I. INTRODUCTION

There has been an increasing trend in using autonomous systems in the maritime industry. According to the Annual Overview of Marine Casualties and Incidents (2020), between 2014 and 2019, 44% of the casualties with ships are due to navigational casualties, which include accidents resulting from contact, collision, and grounding/stranding. Of 1801 accident events that occurred between 2014 and 2019, 54% of them were attributed to human erroneous action [2]. In order to decrease the accident rates that are attributed to human errors, studies regarding automation of surface vehicles focusing on motion planning applications is a currently attractive research area.

The main goal of robotic motion planning is to drive the agent from the start configuration to the goal configuration while obeying the constraints coming from the environment and the agent itself. In motion planning applications, the algorithm's success mainly depends on its capability to handle obstacles and dynamics. Even though motion planning is one of the most popular fields in robotics with a vibrant history, safe and robust motion planning is still an active research area open to advancements. Most motion planning algorithms are built on creating a set of piecewise-smooth trajectories or

The associate editor coordinating the review of this manuscript and approving it for publication was Wonhee Kim.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

waypoints. Feedback control methods are later utilized in this dominant class of planning strategies to follow these trajectories (or waypoints). On the other hand, some researchers implemented trajectory-free motion planning concepts rather than a trajectory-based approach. A typical method in this class segments the map with connected sub-regions. After that, the real-time planner and motion controller aims to drive the robot to the goal region while respecting the constraints coming from these regions [3], [4], [5]. Instead of directly following a pre-defined trajectory, this approach relaxes the "constraints" by increasing the possible movement area for the robot.

Model Predictive Control (MPC) is an effective tool for forcing the constraints implemented on the system for the feedback control phase. Since real-world problems are highly nonlinear, a considerable amount of research has been devoted to extending the usage of MPC to constrained nonlinear systems for motion control applications [6], [7], [8], [9]. As opposed to the linear case, the infinite horizon problem for nonlinear systems cannot be solved numerically, and reducing the horizon may cause undesired system behaviors. In order to address this issue, Michalska and Mayne [10] proposed a hybrid MPC framework that replaces the terminal constraint with a terminal region. When the nonlinear system reaches this region, another controller is employed, and as a result, the system is guaranteed to be asymptotically stable. However, to guarantee the stability of the system, a global optimization problem is required to be solved. Chen and Allgöwer [11] on the other hand, use an infinite horizon approach and calculate a penalty term for the final state to bound the infinite horizon cost. They establish the bound by controlling the nonlinear system with fictitious linear state feedback in the predetermined terminal region.

MPC offers a framework that can handle multiple-input multiple-output (MIMO) systems and force constraints for states and inputs, making it an effective instrument in collision-free motion planning applications. With an increasing demand for the utilization of autonomous surface vessels in the military, search and rescue, transportation and exploration, the literature regarding the control of USVs started to flourish in the last few decades [12]. Several works in literature employ MPC for USV control [13], [14], [15], [16], [17], [18], [19]. Zhao et al. [13] propose an improved MPC framework with the inclusion of global course constraints and an event-triggered mechanism. In [14], researchers propose a finite control set model predictive control for collision avoidance problems.

This paper proposes a trajectory-free, sampling-based feedback motion planning algorithm to handle arbitrary obstacle configurations for fully- and under-actuated autonomous surface vehicles. We specifically focus on developing a robust motion planning and control methodology for USV systems and thus we tested our approaches in the presence of unpredictable process noise. The algorithm consists of three phases. The algorithm first generates a graph structure in the obstacle-free region by creating connected rectangular nodes. Then, graph-search algorithms is deployed to find the "optimal" discrete path on the graph in the second phase. Inside each node (region), an MPC controller is responsible from driving the robot to the next node that is determined by the search algorithm in the second phase. In conclusion, via sequential composition of MPC control policies, the whole algorithm can drive the robot to the goal location.

A preliminary version of this work reporting early results was presented by Karagöz et al. [5]. The current article is a significantly improved version with the following qualities: (i) Karagöz et al. [5] tested the algorithm only with a (2DOF) toy model, whereas in this paper we adopted more comprehensive motion models that can capture realistic full- and under-actuated USV system dynamics (ii) the performance of the algorithm in the presence of extreme process noise is evaluated to test the robustness of the methodology, (iii) the algorithm is modified in order to recover from failures caused by noise, and (iv) MPC and graph search costs are modified to increase the performance of the proposed algorithm.

The remainder of this paper is organized as follows. Section II provides fundamentals regarding the dynamic modeling of a USV and nonlinear MPC concept. Section III summarizes the MPC-Graph algorithm. Section IV and Section V give details regarding the implemented robot models and simulation environments and report the obtained results. Section VI proposes future directions for further improvements.

II. PRELIMINARIES AND BACKGROUND

A. UNMANNED SURFACE VEHICLE DYNAMICS

Compared to unmanned ground vehicles (UGV), unmanned surface vehicles (USV) are exposed to different environment dynamics due to their application medium. Since USVs operate in a sea/ocean environment, on top of rigid body dynamics also, added mass and damping terms should be considered in a USV model. Let $\boldsymbol{\eta} = [x\ y\ \psi]^T \in \mathbb{R}^2 \times S$ denote the pose vector, where x and y are the world-fixed reference frame coordinates and ψ is the heading angle, let $\boldsymbol{\nu} = [u\ v\ r]^T \in \mathbb{R}^3$ denote the velocity vector of the dynamic model where u and v are linear velocities, called surge and sway, and r is the angular velocity. As usual, the equation $\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\psi)\boldsymbol{\nu}$ defines the relationship from body reference frame to world fixed reference frame where $\boldsymbol{J}(\psi)$ is in the form of

$$\boldsymbol{J}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1}$$

Fig. 1 details out the abovementioned coordinate systems as well as a schematic for the USV model.

In our simulations, we use the 3 DOF horizontal plane model presented in [20]. The formulation is as follows,

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} \tag{2a}$$

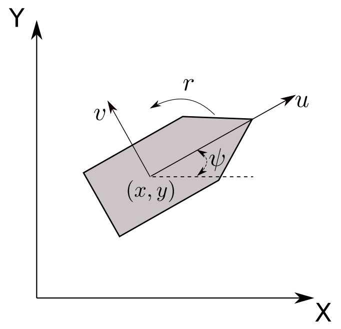


FIGURE 1. Schematic for the USV model. X-Y and *u*-*v* denote the world-fixed and body fixed reference frames, respectively.

where

$$\boldsymbol{M} = \boldsymbol{M}_A + \boldsymbol{M}_{RB} \tag{2b}$$

$$\boldsymbol{C}(\boldsymbol{\nu}) = \boldsymbol{C}_A(\boldsymbol{\nu}) + \boldsymbol{C}_{RB}(\boldsymbol{\nu}) \tag{2c}$$

$$\boldsymbol{D}(\boldsymbol{\nu}) = \boldsymbol{D}_l + \boldsymbol{D}_n(\boldsymbol{\nu}) \tag{2d}$$

In (2a) the general formulation for the USV dynamics is given where $\boldsymbol{M}$, $\boldsymbol{C}(\boldsymbol{\nu})$, $\boldsymbol{D}(\boldsymbol{\nu})$ are inertia, Coriolis/centripetal, damping matrices and $\boldsymbol{\tau}$ is the thruster force vector, respectively. A body moving in a liquid medium, transports some of the surrounding liquid by its motion. As a result, it is observed that the body weighs more compared to its original weight. In order to compensate this effect, added mass terms $\boldsymbol{M}_A$ and $\boldsymbol{C}_A(\boldsymbol{\nu})$ are included in (2b) and (2c). The damping effect is given in (2d) where $\boldsymbol{D}_l$ and $\boldsymbol{D}_n(\boldsymbol{\nu})$ denote the linear and nonlinear damping matrices, respectively. The subscript RB in (2b) and (2c) stands for rigid body terms. The matrices in (2a)-(2d) is represented as

$$\boldsymbol{M}_{RB} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_z \end{bmatrix}, \boldsymbol{M}_A = \begin{bmatrix} X_{\dot{u}} & 0 & 0 \\ 0 & Y_{\dot{v}} & Y_{\dot{r}} \\ 0 & Y_{\dot{r}} & N_{\dot{r}} \end{bmatrix},$$
$$\boldsymbol{C}_A = \begin{bmatrix} 0 & 0 & Y_{\dot{v}}v + Y_{\dot{r}}r \\ 0 & 0 & -X_{\dot{u}}u \\ -Y_{\dot{v}}v - Y_{\dot{r}}r & X_{\dot{u}}u & 0 \end{bmatrix},$$
$$\boldsymbol{C}_{RB} = \begin{bmatrix} 0 & 0 & -mv \\ 0 & 0 & mu \\ mv & -mu & 0 \end{bmatrix} \boldsymbol{D}_l = \begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & Y_r \\ 0 & N_v & N_r \end{bmatrix},$$
$$\boldsymbol{D}_n = \begin{bmatrix} X_{|u|u}|u| & 0 & 0 \\ 0 & Y_{|v|v}|v| & 0 \\ 0 & 0 & N_{|r|r}|r| \end{bmatrix} \tag{3}$$

where m is the mass and I_z is the moment of inertia of the vehicle perpendicular to the horizontal plane, $\{X_{\dot{u}}, Y_{\dot{v}}, Y_{\dot{r}}, N_{\dot{r}}\}$, $\{X_u, Y_v, Y_r, N_v, N_r\}$ and $\{X_{|u|u}, Y_{|v|v}, N_{|r|r}\}$ are added mass, linear damping and nonlinear damping parameters, respectively. These parameters are scalar constants that do not change over time.

B. MODEL PREDICTIVE CONTROL

For the control of the nonlinear USV systems, we use the approach presented in [11]. The work of Chen and Allgöwer is applicable to both stable and unstable systems, and with the inclusion of terminal region and terminal cost matrix, the algorithm guarantees asymptotic closed-loop stability. The objective cost function comprises two parts: an integral square error calculated over a finite horizon and a quadratic terminal cost term as in the linear case.

The optimization problem of MPC can be formalized as follows:

$$J(\mathbf{x}(t), \mathbf{u}(\cdot)) = \int_t^{t+T_p} \left(||\mathbf{x}(\tau)||_Q^2 + ||\bar{\mathbf{u}}(\tau)||_R^2 \right) d\tau + ||\mathbf{x}(t+T_p)||_P^2 \tag{4a}$$

$$\text{subject to } \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{4b}$$

$$\mathbf{u}(\tau) \in U, \tau \in [t, t+T_p] \tag{4c}$$

$$\mathbf{x}(t+T_p) \in \Omega. \tag{4d}$$

where $\mathbf{x}(\cdot)$ is the state vector, $\mathbf{u}(\cdot)$ is the input vector to the system defined by a set of nonlinear differential equations, $f(\mathbf{x}(\cdot), \mathbf{u}(\cdot))$ and T_p is prediction horizon. Q and R are positive-definite and symmetric cost matrices for state and input variables, respectively. We obtained the terminal state penalty matrix $\mathbf{P}$ and terminal region Ω based on the procedure proposed by Chen & Allgöwer [11]. First, the Jacobian linearization of the system at the origin is calculated,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{5}$$

where $\mathbf{A} = (\partial \mathbf{f}/\partial \mathbf{x})(0, 0)$ and $\mathbf{B} = (\partial \mathbf{f}/\partial \mathbf{u})(0, 0)$. Provided that (5) is stabilizable, a linear state feedback $\mathbf{u} = \mathbf{K}\mathbf{x}$ can be obtained and by substituting $\mathbf{u}$ in (5),

$$\dot{\mathbf{x}} = (\mathbf{A} + \mathbf{B}\mathbf{K})\mathbf{x} \tag{6}$$

the obtained matrix $\mathbf{A_K} = \mathbf{A} + \mathbf{B}\mathbf{K}$ is asymptotically stable. The related Lyapunov equation takes the following form,

$$(\mathbf{A}_K + \kappa\mathbf{I})^T\mathbf{P} + \mathbf{P}(\mathbf{A_K} + \kappa\mathbf{I}) = -\mathbf{Q}^{\star} \tag{7}$$

where,

$$\mathbf{Q}^{\star} = \mathbf{Q} + \mathbf{K}^{\mathbf{T}}\mathbf{R}\mathbf{K} \in \mathbb{R}^{n\times n} \tag{8}$$

admits a unique positive-definite and symmetric solution $\mathbf{P}$. In (7), κ is chosen such that it satisfies the following condition,

$$\kappa < -\lambda_{max}(\mathbf{A_K}) \qquad \kappa \in [0, \infty). \tag{9}$$

After determining the terminal state penalty matrix $\mathbf{P}$, the procedure continues with determination of the terminal region. Provided that, there exists a constant $\alpha \in (0, \infty)$, a region Ω_α in the neighborhood of α is defined as follows,

$$\Omega_\alpha = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x}^T\mathbf{P}\mathbf{x} \le \alpha\}. \tag{10}$$

such that $\forall x_1 \in \Omega_\alpha$, infinite horizon cost J^∞,

$$J^\infty(\mathbf{x}_1, \mathbf{u}) = \int_{t_1}^{\infty} (\|\mathbf{x}(t)\|_Q^2 + \|\mathbf{u}(t)\|_R^2) dt \tag{11}$$

starting from $\mathbf{x}(t_1) = \mathbf{x}_1$ and controlled by $\mathbf{u} = \mathbf{K}\mathbf{x}$ is bounded from above as follows,

$$J^\infty(\mathbf{x}_1, \mathbf{u}) \le \mathbf{x}_1^T\mathbf{P}\mathbf{x}_1. \tag{12}$$

With the inclusion of an upper bound to the infinite horizon cost, the asymptotic stability of the system is guaranteed.

III. MPC-GRAPH

The proposed MPC-Graph algorithm executes three successive phases to sample the arena and navigate the robot: graph generation, graph search, and motion control. The graph generation phase takes the map as input and then samples the obstacle-free areas with overlapping rectangular regions until the predefined termination condition is satisfied. After completing the graph generation phase, Dijkstra's algorithm executes to search the obtained neighborhood graph for the shortest available path from any node to the goal node. The motion control phase takes the determined nodes as input and navigates the robot to the goal configuration while respecting the constraints coming from the states and system inputs.

A. GRAPH GENERATION

The MPC-Graph framework starts its execution by sampling the free space. After guaranteeing the sampled point q_{rand} is in the region unobstructed and not covered by previously sampled rectangular nodes, then a new node $Node_k$ is generated and expanded around q_{rand}, see Fig. 2. Note that these nodes can be in different geometric shapes such as square [4], circle [3], [21], rectangle [5]. In this work, we use rectangular nodes due to their compliance with MPC and to achieve a more sparse graph structure [5].

Then, $Node_k$ is added to the graph, and edges are created between this node and the neighboring nodes overlapping with it. Node generation and expansion process continue until the defined termination condition is satisfied.

In the algorithm, we implemented the condition presented in [22] which mainly estimates the quality of the coverage of the sampled space. Let $\mathcal{C}_{free}$ denote the set of configurations in the obstacle-free region and μ be de Lebesgue measure in $\mathcal{C}_{free}$ and $\mathcal{B}$ be the union of the nodes

$$\mathcal{B} = \bigcup_k Node_k,$$

then the implemented termination condition in (13), implicitly determines estimates for the expression $\mu(\mathcal{B})/\mu(\mathcal{C}_{free})$. For that purpose, statistics collected from randomly taking samples to find a new configuration in $\mathcal{C}_{free} \setminus \mathcal{B}$ is analyzed. The implemented termination condition is formalized as follows,

$$m \geq \frac{\ln(1 - P_c)}{\ln\alpha} - 1 \tag{13}$$

where m is the number of successive failures followed by the first success, P_c, and α are user-determined parameters that affect the density of the coverage of the map. A more detailed information about the derivation of (13) is presented in [22]. After satisfying the termination condition, the algorithm continues with the graph search phase to find the optimal route. The computational time regarding this stage is presented in detail in our previous work [5] which gives results for several different maps.

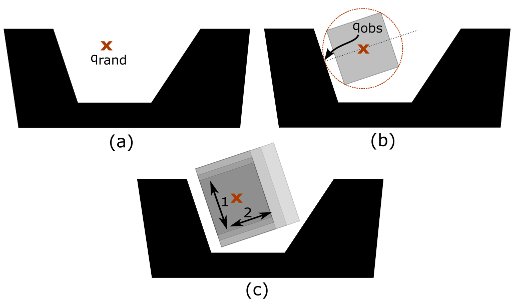


FIGURE 2. Generation of a node: (a) initial map of the arena, (b) a square node is generated, (c) square node is expanded in discrete steps along directions indicated as 1 and 2. A more detailed information about the node generation is presented in [5].

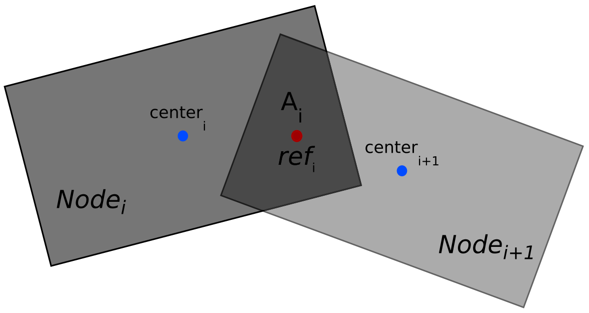


FIGURE 3. Visualization of edge cost parameters. The subscripts *i* and *i* + 1 stand for parent and child node, respectively. Dark gray region A_i shows the overlapping area.

B. GRAPH SEARCH

After the termination condition given in (13) is satisfied, the algorithm continues by searching the obtained graph, G, with Dijkstra's search algorithm to find the optimal discrete planner. The search algorithm returns a policy, Po, that gives the order of the nodes robot should pass to reach the goal.

The constructed graph has an edge between two nodes if they have an overlapping area. We calculate the edge cost as follows,

$$\begin{aligned} cost_{edge} &= \|center_{n_i} - ref_i\|_2 \\ &\quad + \|center_{n_{i+1}} - ref_i\|_2 + \frac{\gamma}{A_i} \end{aligned} \tag{14}$$

where *center* is the intersection point of the diagonals of the corresponding node. The area of the intersection region which is indicated with a dark gray color in Fig. 3 is denoted as A and the center of this region is given as *ref*. $\|\cdot\|_2$ refers to the l_2 norm, i.e. Euclidean distance. We include the reciprocal of the intersection area as a parameter for the edge cost since larger areas provide smoother behavior for the robot. In our simulations we used weight γ for the reciprocal of the intersection area which is set as $\gamma = 1$. Thus, Dijkstra's search algorithm chooses a route in favor of larger intersection areas. Fig. 3 visualizes the parameters given in (14).

C. MOTION CONTROL

The last phase of the proposed MPC-Graph algorithm is motion control. At this stage of the algorithm, with the previously determined policy Po, MPC navigates the robot

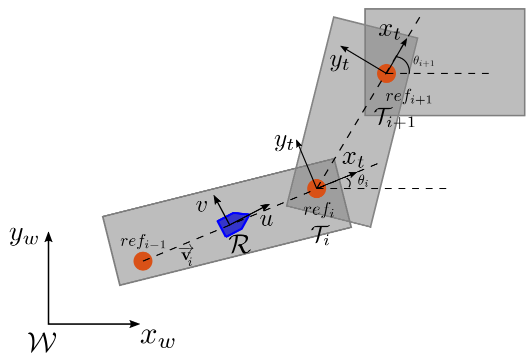


FIGURE 4. Representation of world $\mathcal{W}$, target $\mathcal{T}$ and robot frames $\mathcal{R}$ with the past, current and next nodes.

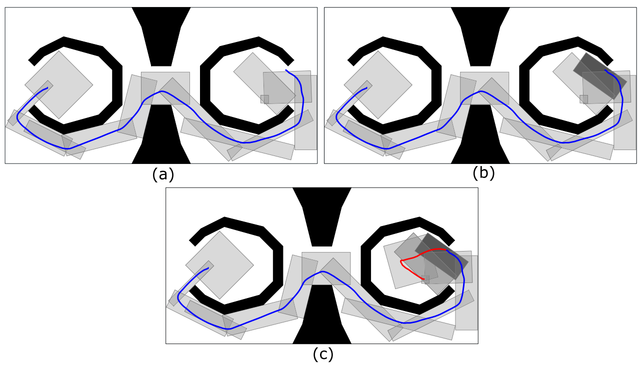


FIGURE 5. (a) Robot arrives to an unsampled region. (b) A new node is generated in the unsampled region which is indicated with dark grey color. (c) Robot follows the red route by using the newly created region.

from an arbitrary node to the *GoalNode* in discrete time steps. The motion control phase aims to safely and smoothly navigate the robot from its *CurrentNode* to the *NextNode* with the established policy *Po* while forcing the state and input constraints. It is important to note that, as long as the robot stays inside the sampled rectangular regions, it is guaranteed that collisions with obstacles are avoided. In order to simulate the physical limits for the robot, we implemented velocity and acceleration constraints. We adopted the quasi infinite MPC [11] approach to guarantee stability of the system whose Jacobian linearization is stabilizable.

The node that the robot is currently in is denoted as the *CurrentNode*, and the target node that the robot is traveling to is called the *NextNode*. The reference point, ref_i, robot aiming to reach is chosen to be the centroid of intersection for that region, $[x_{ref_i}\ y_{ref_i}]^T = Centroid(CurrentNode \cap NextNode)$. After determining the centroid, by taking that point as the origin, a target reference frame $\mathcal{T}_i$ is placed. In order to determine the orientation of that frame, a hypothetical vector $\vec{\mathbf{v}}_i$ starting from the previous reference point ref_{i-1} and ending at the target reference point ref_i is constructed. The angle θ_i between the world frame $\mathcal{W}$ and $\vec{\mathbf{v}}_i$ is taken as the orientation for the target frame. Fig. 4 illustrates world $\mathcal{W}$, target $\mathcal{T}_i$ and robot frames $\mathcal{R}$.

After defining the necessary frames for the algorithm, the following transformation matrices are constructed,

$$\mathbf{T}_t^w = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & x_{ref} \\ \sin\theta & \cos\theta & 0 & y_{ref} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$
$$\mathbf{T}_r^w = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & x \\ \sin\psi & \cos\psi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

where, $\mathbf{T}_t^w \in SE(3)$ is the pose of the target frame with respect to world frame and $\mathbf{T}_r^w \in SE(3)$ is the pose of the robot frame with respect to world frame. By using the matrices given in (15), we calculate the pose of the robot frame with respect to target frame $\mathbf{T}_r^t \in SE(3)$ as

$$\mathbf{T}_r^t = \left(\mathbf{T}_t^w\right)^{-1}\mathbf{T}_r^w. \tag{16}$$

By using the transformation matrix $\mathbf{T}_r^t$, we calculate the position vector of the robot with respect to the target frame $\mathbf{q}_{tr} = [x_{tr}\ y_{tr}\ \psi_{tr}]^T$. In cost function (4a), we use $\mathbf{q}_{tr}$ for robot states. This approach aims to overlap the target frame with a body-fixed robot frame. Using MPC, we calculate the optimal finite-horizon input sequence that satisfies the constraints and navigates the robot towards the origin of the target frame.

When robot enters the intersection area, the *NextNode* becomes the new *CurrentNode* and the new *NextNode* is determined by checking the next element in policy *Po*. It is important to note that for every new region the robot enters, the state constraints coming from the boundaries of the rectangular nodes are re-calculated. This process executes recursively until the robot arrives at the node, which includes the goal point q_{goal}. In the goal region, q_{goal} becomes the reference point, and *NextNode* is no longer applicable.

Due to unpredictable effects such as process noise, the robot may end up in an unsampled region or another node different from its *CurrentNode*. For the former case, the last position of the robot is treated as the sampled random point, q_{rand}, and fed to the node generation function. This newly generated node is inserted into the previously obtained graph, G, and taken as the new *CurrentNode* for the robot. Then, Dijkstra's search algorithm is executed to generate a new policy. These steps are given in lines 11-19 of Algorithm 1. An illustration of this resampling procedure is presented in Fig. 5.

For the latter case, the obtained graph is searched in order to determine the possible *CurrentNode*, which ensures the minimum cost route according to (14). After determining the *CurrentNode*, a new route is obtained using the previously generated policy, *Po*. These steps are given in lines 21-23 of Algorithm 1. A complete procedure of the motion control phase is summarized in Algorithm 1.

IV. ROBOT MODELS

A. FULLY ACTUATED USV MODEL

For the fully actuated USV model, we implement the outlined model illustrated in Fig. 6(a) that consists of four thrusters which generates the indicated force vector, F_1, F_2, F_3 and F_4. With this outlined model, the force vector $\boldsymbol{\tau}_{fa}$ takes the

Algorithm 1 Motion Control

```
1:  $CurrentNode \leftarrow StartNode()$
2:  $NextNode \leftarrow Po$.Next($CurrentNode$)
3:  $ref \leftarrow$ Centroid($CurrentNode \cap NextNode$)
4:  while $q_{goal}$ not reached do
5:      if $q_t \in GoalNode$ then
6:          $ref \leftarrow q_{goal}$
7:      else if $q_t \in NextNode$ then
8:          $CurrentNode \leftarrow NextNode$
9:          $NextNode \leftarrow Po$.Next($CurrentNode$)
10:         $ref \leftarrow$ Centroid($CurrentNode \cap NextNode$)
11:     else if $q_t \in UnsampledRegion$ then
12:         $q_t \leftarrow$ UniformRandConf()
13:         $Node_i \leftarrow$ GenerateRectRegion($q_t$)
14:         $Node_i \leftarrow$ Expand($Node_i$)
15:         $G$.InsertNode($Node_i$)
16:         Po = DijkstraAlgorithm($G$)
17:         $CurrentNode \leftarrow Node_i$
18:         $NextNode \leftarrow Po$.Next($CurrentNode$)
19:         $ref \leftarrow$ Centroid($CurrentNode \cap NextNode$)
20:     else
21:         $CurrentNode \leftarrow$ SearchNodes($G$, $q_t$)
22:         $NextNode \leftarrow Po$.Next($CurrentNode$)
23:         $ref \leftarrow$ Centroid($CurrentNode \cap NextNode$)
24:     end if
25:     $u_t \leftarrow$ MPC($q_t$, $ref$, $CurrentNode$)
26: end while
```

form of

$$\boldsymbol{\tau}_{fa} = \begin{bmatrix} (F_1 + F_2 - F_3 - F_4) sin\alpha \\ (F_1 + F_3 - F_2 - F_4) cos\alpha \\ (F_1 + F_4 - F_2 - F_3)(sin\alpha + cos\alpha) b/2 \end{bmatrix}. \quad (17)$$

With the oriented placement of the thrusters with respect to the body, the vehicle can generate a force vector in direction v as opposed to the differential model. In the implementation, we adopted the parameters given in [23] for inertia, damping, and added mass terms. The state vector and the input vector for the model takes the form $\mathbf{q} = [x\ y\ \theta\ u\ v\ r]^T$ and $\mathbf{u} = [F_1\ F_2\ F_3\ F_4]^T$, respectively.

In the simulations we discretize the continuous nonlinear dynamics of the system and uniform synchronous sampling of measurements with a sampling frequency of $f_s = 10\,Hz$(or $T_s = 0.1s$). We navigate the vehicle throughout the map with quasi-infinite horizon MPC. The finite horizon length is $T_p = 1.5s$, which gives us enough degrees of freedom in enforcing the state and input constraints. We choose the state and input matrices $\mathbf{Q} = diag(5, 5, 5, 2.5, 2.5, 0.2)$ and $\mathbf{R} = diag(0.1, 0.1, 0.1, 0.1)$, respectively. After determining the cost matrices, by following the procedure presented in [11], we obtain the state feedback gain $\mathbf{K}$ and the terminal penalty matrix $\mathbf{P}$.

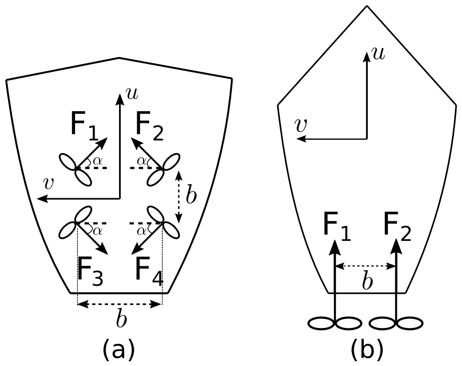


FIGURE 6. Placement of thrusters for (a) fully actuated USV model and (b) differential USV model.

We implemented the following state and input constraints to the system,

$$A_i [x\ y]^T \leq B_i \quad (18a)$$
$$-1\ rad/s \leq r \leq 1\ rad/s \quad (18b)$$
$$-1.5\ m/s \leq u \leq 1.5\ m/s \quad (18c)$$
$$-20\ N \leq F_1, F_2, F_3, F_4 \leq 20\ N \quad (18d)$$

Equation (18a) is calculated for each node on the path.

B. DIFFERENTIAL USV MODEL

For the differential USV model with two thrusters, we implement the model illustrated in Fig. 6(b). With this model, the force vector $\boldsymbol{\tau}_{ua}$ takes the form of

$$\boldsymbol{\tau}_{ua} = \begin{bmatrix} F_1 + F_2 \\ 0 \\ b(F_1 - F_2) \end{bmatrix}. \quad (19)$$

For the differential USV model, the placement of thrusters prevents the generation of a force vector in the v direction. In the implementation, we adopted the parameters given in [24] for inertia, damping, and added mass terms. The state vector and the input vector for the model takes the form $\mathbf{q} = [x\ y\ \theta\ u\ v\ r]^T$ and $\mathbf{u} = [F_1\ F_2]^T$, respectively.

In the simulations, we discretize the continuous nonlinear dynamics of the system and uniform synchronous sampling of measurements with a sampling frequency of $f_s = 10Hz$ (or $T_s = 0.1s$). The linearized system at the origin is not stabilizable for the under-actuated model, so the terminal region and terminal cost matrix cannot be calculated. In order to control the system and predict its future behavior, we selected the finite horizon length as $T_p = 9s$. It is six times larger than the horizon length used for the fully actuated USV model. We choose the state and input matrices $\mathbf{Q} = diag(5, 5, 10, 2.5, 2.5, 0.2)$ and $\mathbf{R} = diag(0.1, 0.1)$, respectively.

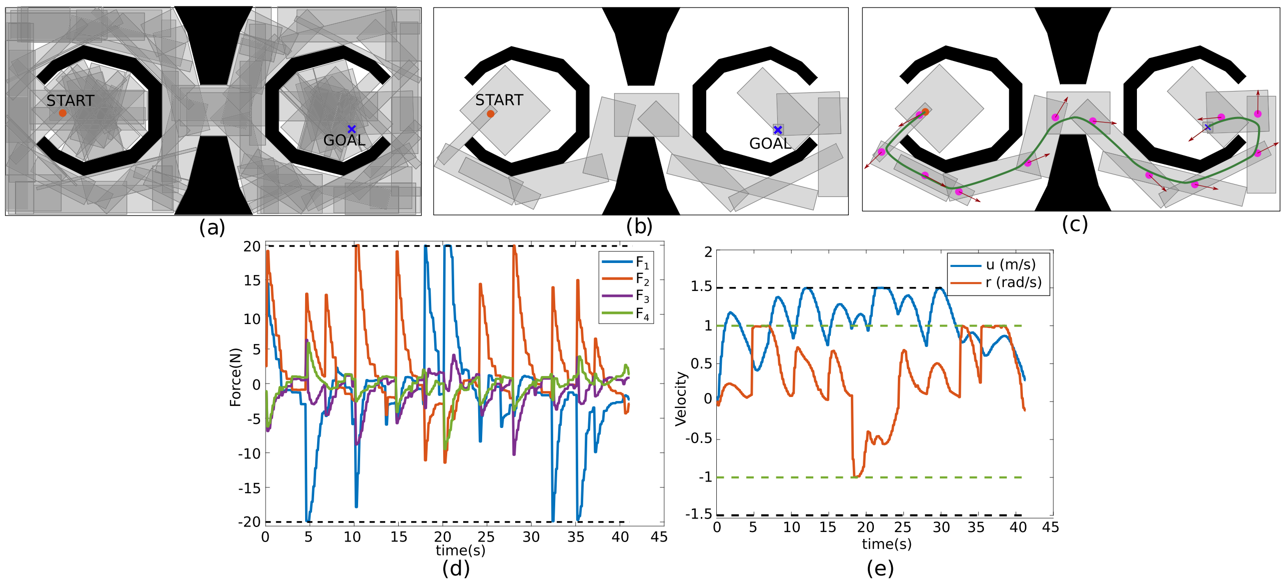


FIGURE 7. (a) A total of 114 nodes are generated in node generation phase. (b) The calculated optimal route consists of 14 nodes. (c) Green curve indicates the trajectory followed by the robot. Pink points represent the calculated reference points. Red arrows show the calculated target frame $\mathcal{T}$ orientations in positive x_t direction. (d) Applied thruster input forces F_1, F_2, F_3, F_4 to the system. Dark dashed lines indicate upper and lower constraints for the input. (e) Surge speed u and angular rate r of the robot. Black and green dashed lines correspond to the constraints for surge speed and angular rate, respectively.

We implemented the following state and input constraints to the system,

$$A_i[x\ y]^T \leq B_i \tag{20a}$$

$$-2\ rad/s \leq r \leq 2\ rad/s \tag{20b}$$

$$-0.5\ m/s \leq u \leq 3\ m/s \tag{20c}$$

$$-20\ N \leq F_1, F_2 \leq 20\ N \tag{20d}$$

Equation (20a) is calculated for each node on the path.

V. SIMULATION RESULTS

This section reports the simulation results obtained from implementing the MPC-Graph algorithm. We implemented our algorithm on MATLAB and performed simulations on a laptop with Intel i7 2.4 GHz processor running Windows OS.

A. FULLY ACTUATED USV MODEL

1) PERFORMANCE WITHOUT PROCESS NOISE

In this simulation scenario, we used a map that consists of 4 obstacles. In the graph generation phase, a total of 114 nodes are generated. In order to show the ability of the controller to force the constraints coming from the boundaries of the rectangular nodes, we replaced the goal node with a smaller square region. After executing Dijkstra's search algorithm, the optimal route consists of 14 nodes. Fig. 7(a) and (b) visualizes the obtained rectangular regions and the node-set that constitutes the followed route by the robot, respectively.

Fig. 7(c) shows the route followed by the robot and the calculated reference points with target orientations. Red arrows are the positive $\mathbf{x_t}$ directions for the target frames $\mathcal{T}$. The figure points out that the robot obeys the constraints coming from the boundaries of the sampled rectangular regions. Even in the smaller goal region ($\sim$ 24 times smaller concerning the average size of the nodes in Po) robot approaches to goal point smoothly by considering the dynamics and the imposed constraints on the system.

Plot presented in Fig. 7(d) shows the forces F_1, F_2, F_3, F_4 calculated in the motion control phase. In order to simulate a realistic system, we set upper and lower bounds 20 N and -20 N for the input forces, respectively. Dashed black lines in the figure indicate these constraints. Furthermore, we also added velocity constraints to the system. For the surge speed u, we set upper and lower constraints as 1.5 m/s and -1.5 m/s, respectively. For the angular rate r, we set upper and lower constraints as 1 rad/s and -1 rad/s, respectively. In Fig. 7(e), the imposed constraints on surge speed and angular rate are indicated with black and green dashed lines, respectively. It can be inferred from the figures that MPC can force both state and input constraints successfully. CPU time of MPC for each is at average $t_{CPU} = 0.021s$.

2) PERFORMANCE IN THE PRESENCE OF PROCESS NOISE

We also analyzed the performance of the MPC-Graph algorithm in the presence of process noise. To this end, we added noise (SNR=1) to the calculated thruster forces, $u = [F_1\ F_2\ F_3\ F_4]^T$ and performed Monte-Carlo experiments (# simulations = 1000) on the same map with the same nodes. We determine the thrusters' input saturation limit as $F = 25\ N$. From 1000 Monte-Carlo experiments, in 8 of them robot failed to reach the goal location due to process noise. Failures include the cases in which the robot ends up

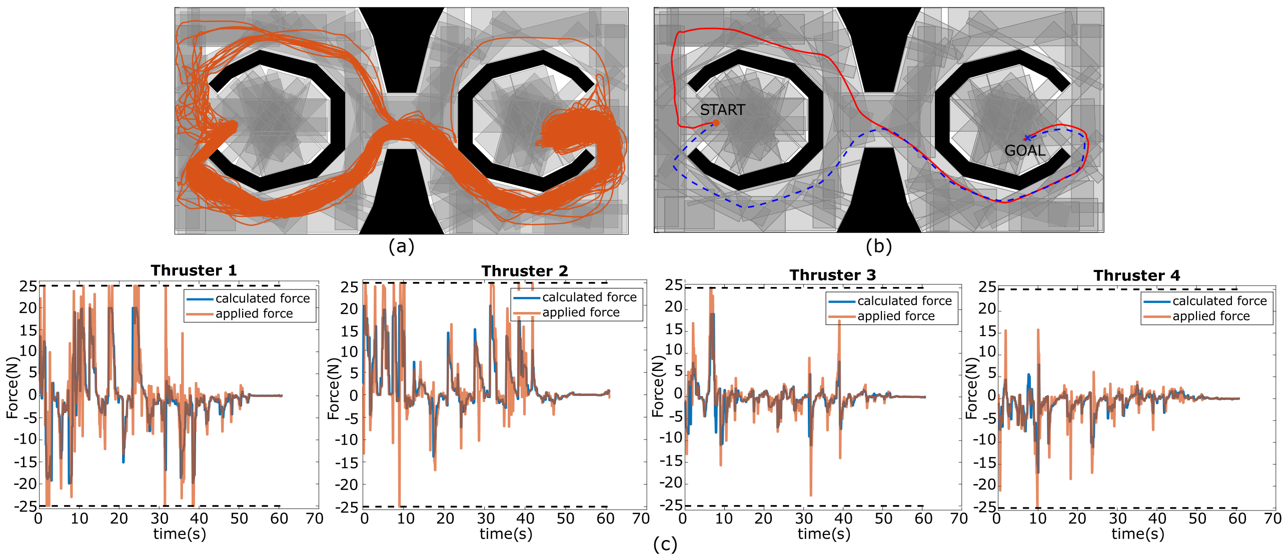


FIGURE 8. Obtained trajectories from Monte-Carlo experiments in the presence of process noise. (a) Successful trajectories ended up in goal point. (b) Red trajectory indicates the followed route in the presence of process noise. Dashed blue trajectory indicates the followed trajectory without the process noise. (c) Plots show the calculated forces and applied forces on each thruster. Black dashed lines indicate the saturation limits for the thrusters.

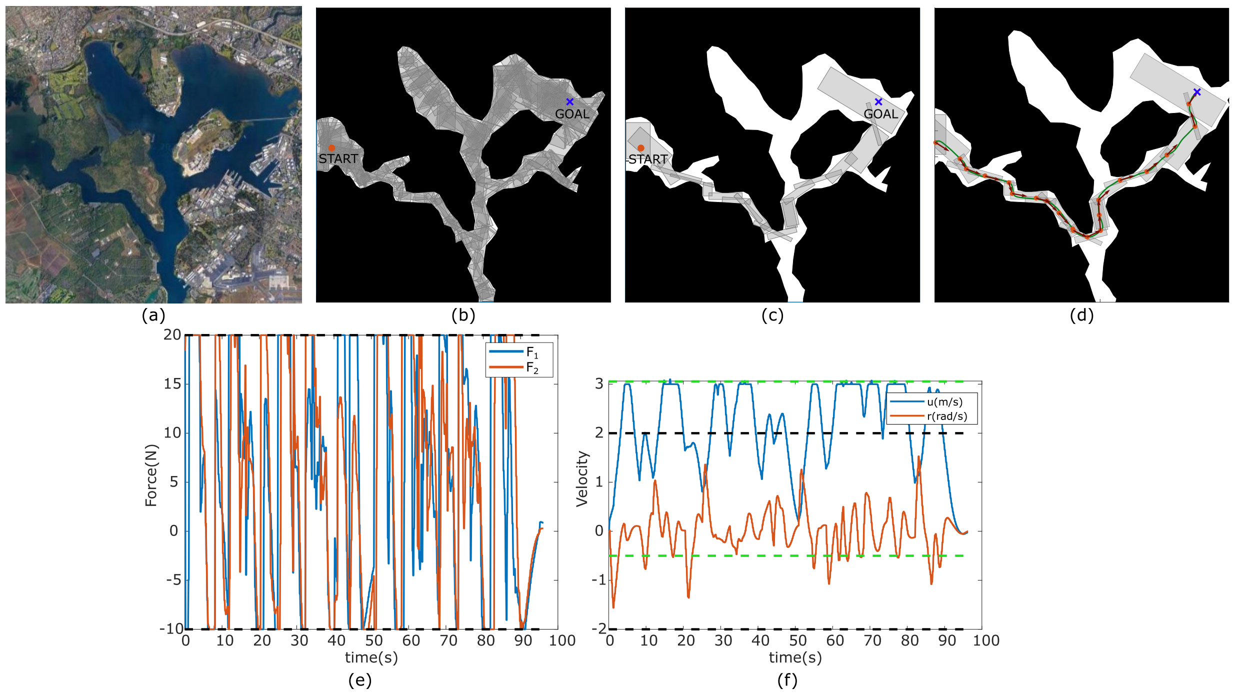


FIGURE 9. (a) Satellite view of Pearl Harbor is presented. (b) Node generation phase. As a result 255 nodes are generated. (c) Graph search phase determines the shortest route consisting of 21 nodes. (d) Green curve indicates the trajectory followed by the robot. Orange points represent the calculated reference points. Red arrows show the calculated target frame $\mathcal{T}$ orientations in positive x_t direction. (e) Applied thruster input forces F_1, F_2 to the system. Dark dashed lines indicate upper and lower constraints for the input. (f) Surge speed u and angular rate r of the robot. Black and green dashed lines correspond to the constraints for surge speed and angular rate, respectively.

outside the limits of the arena or inside the obstacles. Fig. 8(a) visualizes the successful attempts for reaching the goal point.

If the robot ends up in a different node than the *CurrentNode* due to process noise, the algorithm generates another route considering the previously determined policy *Po*. In Fig. 8(b), red and dashed blue trajectories indicate the routes followed by the robot in the presence of process noise and without noise, respectively. Fig. 8(c) shows the thruster forces and process noise applied on each thruster. Even in the presence of high-level noise, in 99.2% of the experiments

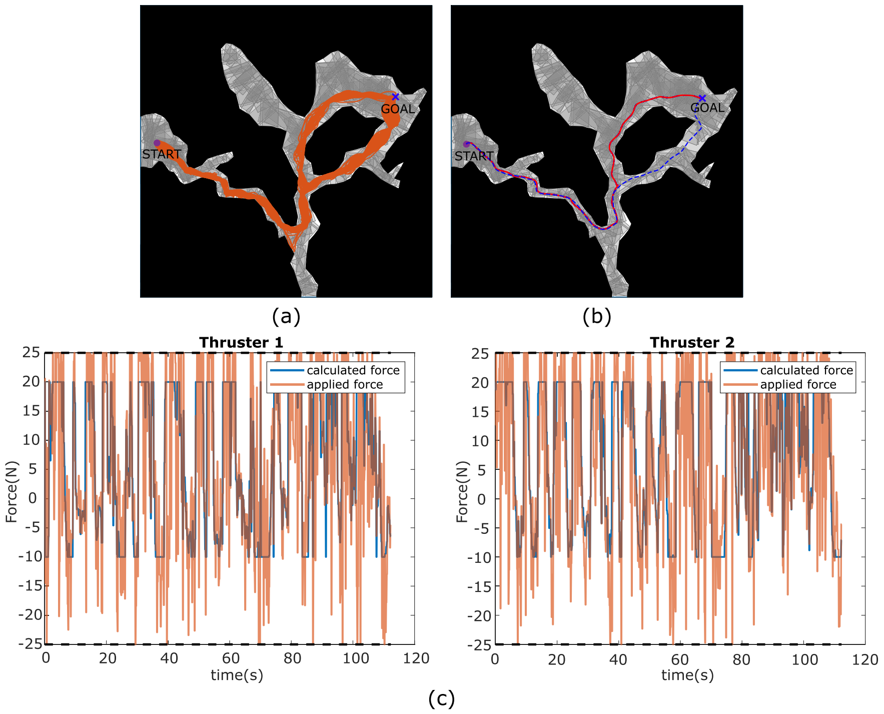


FIGURE 10. **Obtained trajectories from Monte-Carlo experiments in the presence of process noise. (a) Successful trajectories ended up in goal point. (b) Red trajectory indicates the followed route in the presence of process noise. Dashed blue trajectory indicates the followed trajectory without the process noise. (c) Plots show the calculated forces and applied forces on each thruster. Black dashed lines indicate the saturation limits.**

fully actuated USV model reaches the goal location. Results indicate that MPC is an effective controller for handling high noise scenarios.

B. DIFFERENTIAL USV MODEL

1) PERFORMANCE WITHOUT PROCESS NOISE

In this simulation scenario, we used the map of Pearl Harbor. Graph generation phase constructed a graph structure consisting of 255 nodes and Dijkstra's search algorithm find the optimal route that consists of 21 nodes. Fig. 9(a) and (b) visualize the satellite view of Pearl Harbor and obtained rectangular regions, respectively. Fig. 9(c) and (d) illustrate the node-set robot navigates in and path followed by the robot, respectively.

We set the upper and lower bounds 20 N and -10 N for the input forces, respectively. In order to restrict the backward motion of the vehicle, we set the lower limit for the thruster forces to be -10 N. Velocity constraints for the system are 3 m/s and -0.5 m/s for the surge speed u and 2 rad/s and -2 rad/s for the angular rate r, respectively. Figures 9(e) and 9(f) show the calculated forces F_1, F_2 and surge speed-angular rate of the vehicle throughout the simulation. It is important to note that the linearized differential USV model is not stabilizable. Thus, it is not possible to determine a terminal region and a terminal cost matrix for this system. Although the stability is not guaranteed for the differential USV model, with the adjusted MPC parameters, the controller can drive the system to the goal location while obeying the imposed constraints. CPU time of MPC for each iteration for this model is at average $t_{CPU} = 0.28s$.

2) PERFORMANCE IN THE PRESENCE OF PROCESS NOISE

For the simulation scenario with noise, we performed Monte-Carlo experiments (# simulations = 1000) on the same map with the same nodes and plotted the trajectories followed by the robot. The process noise (SNR=1) is applied to the calculated thruster forces $u = [F_1\ F_2]^T$ which have an input saturation limit of $F = 25\ N$.

From 1000 Monte-Carlo experiments, in 29 of them robot failed to reach the goal location. Fig. 10 (a) visualizes the successful attempts to reach the goal point. In Fig. 10(b), red and dashed blue trajectories indicate the routes followed by the robot in the presence of process noise and without noise, respectively. Fig. 10(c) shows the thruster forces and process noise applied on each thruster. Even in the presence of high-level noise, in 97.1% of the experiments differential USV model reaches the goal location.

VI. CONCLUSION

This study presents a feedback motion planning and control methodology for dynamic fully- and under-actuated USV models, utilizing sparse random neighborhood graphs and constrained nonlinear Model Predictive Control (MPC). The proposed approach enables safe and robust navigation in marine environments by providing an autonomous motion planning strategy that ensures no constraint violation on state and input variables. The algorithm generates a sparse neighborhood graph consisting of connected rectangular zones in the planning phase, and inside each node, an MPC-based online feedback control policy funnels the USV with nonlinear dynamics from one rectangle to the other in the network. The effectiveness and robustness of the proposed algorithm is systematically tested in different simulation scenarios, including an extreme actuator noise scenario, and the results demonstrate the validity and effectiveness of the proposed approach. The execution time of the MPC for each iteration provides a bottleneck for the algorithm. For the fully actuated model, the average CPU time is 0.021s which is strictly lower than the sampling time. We believe that with the indicated CPU time, the proposed algorithm can also be applicable to real-time. However, there are still several directions for future research that can be explored to further enhance the proposed approach.

Firstly, although the proposed algorithm showed promising results, the under-actuated model's performance can be further improved. One potential approach to achieve this is to investigate the use of different control techniques that can guarantee the under-actuated model's stability while still ensuring obstacle avoidance and robustness.

Secondly, the simulation results showed that the main drawback of using an MPC-based controller is the high computational cost. Hence, it is important to further decrease the CPU time of the algorithm to make it more practical and applicable in real-world scenarios. This can be achieved by exploring the use of different and more efficient solvers, such as embedded optimization solvers [25], computational optimization techniques [26], or efficient MPC formulations [27].

Furthermore, as USVs become more widely used in different applications, it is essential to consider more complex scenarios with multiple USVs and dynamic obstacles. Thus, future research can focus on developing multi-agent motion planning and control methods that can handle multiple USVs navigating in a shared environment and avoiding collisions with each other and dynamic obstacles.

Another potential avenue for future work is to develop a corridor-based method to increase the robustness of the algorithm, especially for the cases where narrow tunnels exist. Several works in literature implement corridor-based methods for docking applications of USVs [28] and safe trajectory planning for autonomous dispatch on flight deck [29] which show promising results and can be further implemented for safe trajectory planning for USVs.

The proposed algorithm provides a promising approach for improving the autonomy level of USVs and ensuring safe and robust navigation across unpredictable marine environments. However, further research is needed to address the limitations and enhance the proposed approach's effectiveness and practicality in real-world scenarios.

ACKNOWLEDGMENT

An earlier version of this paper has been presented in Simay Atasoy's M.Sc. thesis [1].

REFERENCES

[1] S. Atasoy, "MPC-Graph: Nonlinear feedback motion planning using sparse sampling based neighborhood graph," M.S. thesis, Middle East Tech. Univ., 2022.

[2] *Annual Overview of Marine Casualties and Incidents 2020*, Eur. Maritime Safety Agency, 2020.

[3] E. Ege and M. M. Ankarali, "Feedback motion planning of unmanned surface vehicles via random sequential composition," *Trans. Inst. Meas. Control*, vol. 41, no. 12, pp. 3321–3330, Aug. 2019.

[4] F. Golbol, M. M. Ankarali, and A. Saranli, "RG-trees: Trajectory-free feedback motion planning using sparse random reference governor trees," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 6506–6511.

[5] O. K. Karagoz, S. Atasoy, and M. M. Ankarali, "MPC-graph: Feedback motion planning using sparse sampling based neighborhood graph," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 6797–6802.

[6] C. Sun, X. Zhang, Q. Zhou, and Y. Tian, "A model predictive controller with switched tracking error for autonomous vehicle path tracking," *IEEE Access*, vol. 7, pp. 53103–53114, 2019.

[7] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," in *Proc. IEEE-RAS 17th Int. Conf. Humanoid Robot. (Humanoids)*, Nov. 2017, pp. 577–584.

[8] T. A. V. Teatro, J. M. Eklund, and R. Milman, "Nonlinear model predictive control for omnidirectional robot motion planning and tracking with avoidance of moving obstacles," *Can. J. Electr. Comput. Eng.*, vol. 37, no. 3, pp. 151–156, 2014.

[9] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 952–964, Feb. 2017.

[10] H. Michalska and D. Q. Mayne, "Robust receding horizon control of constrained nonlinear systems," *IEEE Trans. Autom. Control*, vol. 38, no. 11, pp. 1623–1633, Nov. 1993.

[11] H. Chen and F. Allgower, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205–1217, 1998.

[12] A. Vagale, R. Oucheikh, R. T. Bye, O. L. Osen, and T. I. Fossen, "Path planning and collision avoidance for autonomous surface vehicles I: A review," *J. Mar. Sci. Technol.*, vol. 26, pp. 1292–1306, Jan. 2021.

[13] B. Zhao, X. Zhang, C. Liang, and X. Han, "An improved model predictive control for path-following of USV based on global course constraint and event-triggered mechanism," *IEEE Access*, vol. 9, pp. 79725–79734, 2021.

[14] X. Sun, G. Wang, Y. Fan, D. Mu, and B. Qiu, "Collision avoidance using finite control set model predictive control for unmanned surface vehicle," *Appl. Sci.*, vol. 8, no. 6, p. 926, Jun. 2018.

[15] X. Zhou, Y. Wu, and J. Huang, "MPC-based path tracking control method for USV," in *Proc. Chin. Autom. Congr. (CAC)*, Nov. 2020, pp. 1669–1673.

[16] Z. Liu, C. Geng, and J. Zhang, "Model predictive controller design with disturbance observer for path following of unmanned surface vessel," in *Proc. IEEE Int. Conf. Mechatronics Autom. (ICMA)*, Aug. 2017, pp. 1827–1832.

[17] W. Wang, T. Shan, P. Leoni, D. Fernandez-Gutierrez, D. Meyers, C. Ratti, and D. Rus, "Roboat II: A novel autonomous surface vessel for urban environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 1740–1747.

[18] B. Hu, Y. Wan, and Y. Lei, "Collision avoidance of USV by model predictive control-aided deep reinforcement learning," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Aug. 2022, pp. 1–6.

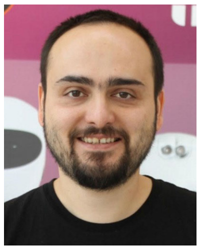
[19] X. Wang, J. Liu, H. Peng, X. Qie, X. Zhao, and C. Lu, "A simultaneous planning and control method integrating APF and MPC to solve autonomous navigation for USVs in unknown environments," *J. Intell. Robotic Syst.*, vol. 105, no. 2, p. 36, Jun. 2022.

[20] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Hoboken, NJ, USA: Wiley, 2011.

[21] L. Yang and S. M. LaValle, ''A framework for planning feedback motion strategies based on a random neighborhood graph,'' in *Proc. IEEE Int. Conf. Robot. Automat.*, Apr. 2000, pp. 544–549.
[22] L. Yang and S. M. LaValle, ''The sampling-based neighborhood graph: An approach to computing and executing feedback motion strategies,'' *IEEE Trans. Robot. Autom.*, vol. 20, no. 3, pp. 419–432, Jun. 2004.
[23] M. Kumru, ''Navigation and control of an unmanned sea surface vehicle,'' M.S. thesis, Middle East Tech. Univ., 2015.
[24] R. Skjetne, T. I. Fossen, and P. V. Kokotovic, ''Adaptive maneuvering, with experiments, for a model ship in a marine control laboratory,'' *Automatica*, vol. 41, no. 2, pp. 289–298, Feb. 2005.
[25] B. Stellato, V. V. Naik, A. Bemporad, P. Goulart, and S. Boyd, ''Embedded mixed-integer quadratic optimization using the OSQP solver,'' in *Proc. Eur. Control Conf. (ECC)*, Jun. 2018, pp. 1536–1541.
[26] S. Richter, *Computational Complexity Certification of Gradient Methods for Real-Time Model Predictive Control*. Zurich, Switzerland: ETH Zurich, 2012.
[27] O. Arpacik and M. M. Ankarali, ''An efficient implementation of online model predictive control with field weakening operation in surface mounted PMSM,'' *IEEE Access*, vol. 9, pp. 167605–167614, 2021.
[28] X. Wang, Z. Deng, H. Peng, L. Wang, Y. Wang, L. Tao, C. Lu, and Z. Peng, ''Autonomous docking trajectory optimization for unmanned surface vehicle: A hierarchical method,'' *Ocean Eng.*, vol. 279, Jul. 2023, Art. no. 114156.
[29] X. Wang, B. Li, X. Su, H. Peng, L. Wang, C. Lu, and C. Wang, ''Autonomous dispatch trajectory planning on flight deck: A search-resampling-optimization framework,'' *Eng. Appl. Artif. Intell.*, vol. 119, Mar. 2023, Art. no. 105792.

SİMAY ATASOY (Member, IEEE) received the B.S. degree in mechanical engineering and the M.S. degree in electrical and electronics engineering from Middle East Technical University (METU), in 2017 and 2022, respectively. She is currently a Research Engineer with Aselsan Inc. Her research interests include model predictive control, motion planning, and collision avoidance.

OSMAN KAAN KARAGÖZ (Member, IEEE) received the B.S. and M.S. degrees in electrical and electronics engineering from Middle East Technical University (METU), Ankara, in 2017 and 2020, respectively, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Electronics Engineering. His research interests include nonlinear dynamical systems, control theory, and motion control in robotics and biological systems.



MUSTAFA MERT ANKARALI (Member, IEEE) received the B.S. degree in mechanical engineering and the M.S. degree in electrical and electronics engineering from Middle East Technical University (METU), in 2007 and 2010, respectively, and the Ph.D. degree in mechanical engineering from Johns Hopkins University (JHU), in 2015. Following his Ph.D. degree, he was a Postdoctoral Fellow with the Laboratory for Computational Sensing and Robotics (LCSR), JHU, until 2016. In 2017, he joined the Electrical and Electronics Engineering Department, METU, where he is currently an Associate Professor. His research interests include analyzing, identifying, and controlling dynamic behaviors in robotic and biological systems.