**ELECTRONIC PREPRINT: P. Švec, A. Thakur, E. Raboin, B.C. Shah, S.K. Gupta. Target Following with Motion Prediction for Unmanned Surface Vehicle Operating in Cluttered Environments. Autonomous Robots, Accepted for publication.**

**Autonomous Robots, manuscript No.**
(will be inserted by the editor)

# Target Following with Motion Prediction for Unmanned Surface Vehicle Operating in Cluttered Environments

**Petr Švec · Atul Thakur · Eric Raboin · Brual C. Shah · Satyandra K. Gupta**

Received: date / Accepted: date

**Abstract** The capability of following a moving target in an environment with obstacles is required as a basic and necessary function for realizing an autonomous unmanned surface vehicle (USV). Many target following scenarios involve a follower and target vehicles that may have different maneuvering capabilities. Moreover, the follower vehicle may not have prior information about the intended motion of the target boat. This paper presents a trajectory planning and tracking approach for following a differentially constrained target vehicle operating in an obstacle field. The developed approach includes a novel algorithm for computing a desired pose and surge speed in the vicinity of the target boat, jointly defined as a motion goal, and tightly integrates it with trajectory planning and tracking components of the entire system. The trajectory planner generates a dynamically feasible, collision-free trajectory to allow the USV to safely reach the computed motion goal. Trajectory planning needs to be sufficiently fast and yet produce dynamically feasible and short trajectories due to the moving target. This required speeding up the planning by searching for trajectories through a hybrid, pose-position state space using a multi-resolution control action set. The search in the velocity space is decoupled from the search for a trajectory in the pose space. Therefore, the underlying trajectory tracking controller computes desired surge speed for each segment of the trajectory and ensures that the USV maintains it. We have carried out simulation as well as experimental studies to demonstrate the effectiveness of the developed approach.

**Keywords** Unmanned surface vehicle (USV) · Follow behavior · Motion prediction · Trajectory planning · Trajectory tracking

Petr Švec
Simulation Based System Design Laboratory, Maryland Robotics Center, Department of Mechanical Engineering, University of Maryland, College Park, MD 20742, USA
E-mail: petrsvec@umd.edu

Atul Thakur
Department of Mechanical Engineering, Indian Institute of Technology Patna, Patliputra, Bihar, 800013, India
E-mail: athakur@iitp.ac.in

Eric Raboin
Department of Computer Science, University of Maryland, College Park, MD 20742, USA
E-mail: eraboin@umd.edu

Brual C. Shah
Simulation Based System Design Laboratory, Department of Mechanical Engineering, University of Maryland, College Park, MD 20742, USA
E-mail: brual@umd.edu

Satyandra K. Gupta
Simulation Based System Design Laboratory, Maryland Robotics Center, Department of Mechanical Engineering and Institute for Systems Research, University of Maryland, College Park, MD 20742, USA
E-mail: skgupta@umd.edu

## 1 Introduction

Autonomous unmanned surface vehicles (USVs) [1,2,3,4] can increase the capability of other surface or underwater vehicles by continuously following them in marine applications. These include sea-bed mapping and ocean sampling tasks [5,6,7] in environmental monitoring [8,9,10], cooperative surveillance by means of a network of heterogeneous vehicles (e.g., air, ground, and marine unmanned platforms) to provide situational awareness [11,12,13], search and rescue [14,15], or harbor patrolling and protecting vulnerable areas [16,17,18,19,20].

Consider the case of a cooperative team of USVs guarding an asset against hostile boats in naval missions [20]. In

Fig. 1: A typical bay with various types of moving boats (source: Map data ©2013 Google).

these missions, the vehicles are required to approach passing boats, recognize adversaries, and possibly employ active blocking [19] to prevent the adversaries from reaching the asset. In order to maximize the guarding performance of the entire team, it is necessary for each USV to consider its own dynamics when approaching or following the boats. Moreover, each USV should be able to reliably compute its desired position $[x, y]^T$, orientation $\psi$, and surge speed $u$ values, jointly defined as a motion goal, by estimating the future poses of the boats.

Performing autonomous follow task in an unfamiliar, unstructured marine environment (see Figure 1) with obstacles of variable dimensions, shapes, and motion dynamics such as other unmanned surface vehicles, civilian boats, adversaries, shorelines, or docks poses numerous planning challenges. The follow capability of the USV is inherently influenced by its own maneuverability constraints, the specific motion characteristics of the target boat, the amount of knowledge about the intended motion of the target boat, sensing limitations, and the complexity of the marine environment. Due to differences in motion capabilities with respect to the target boat, the USV may not be able to track the same trajectory as the target boat. Instead, it may need to determine a different trajectory to follow, while keeping itself in proximity to the target boat and still avoid collisions. In addition, the USV may need to handle sharp turns while tracking the trajectory. Therefore, the mere utilization of manually developed and tuned control rules would not lead to sufficiently safe and task efficient follow strategies in environments with obstacles. In order to cope with the above outlined challenges, the USV needs to have a) the capability to estimate the future motion of the target boat based on its current state, dynamic characteristics, as well as obstacles in the operating environment, b) the capability to determine an advantageous and safe pose, i.e., in the form of a motion goal, in the close vicinity to the target boat, and c) fast, dynamically feasible trajectory planning and reliable trajectory tracking to guarantee physics-aware obstacle avoidance when approaching the motion goal.

We have developed a planning and tracking approach that incorporates a novel algorithm for motion goal computation, and tightly integrates it with trajectory planning and tracking components of the entire system to perform integrated planning and control. The motion goal is computed based on differential constraints of the USV and the target boat, expected motion of the target that is computed using a probability distribution over its possible control actions, and spatial constraints imposed by the environment. In particular, it forward-projects the control actions of the target boat to estimate a probability distribution of its future pose, computes candidate motion goals, selects the motion goal that minimizes the difference between the arrival time of the USV and the target boat to that goal, and attempts to minimize the length of the USV's trajectory.

The developed trajectory planner incorporates A* based heuristic search [21] to efficiently find a collision-free, dynamically feasible trajectory to a motion goal in a discretized state-action space, forming a lattice [22]. The trajectory is computed by sequencing predefined control actions (i.e., maneuvers or motion primitives) generated using the dynamics model of the USV [23, 24]. The trajectory is executed by a trajectory tracking controller to efficiently follow waypoints that make up the nominal trajectory. The controller computes the desired speed for each segment of the trajectory given the maximum allowable surge speed of the segment. This allows the USV to arrive to the motion goal at the required time. We have carried out experimental tests to determine the required acceleration and deceleration distance for the vehicle to match the desired speed.

In general, one can assume that the USV is capable of rejecting ocean disturbances in low sea states due to its mechanical design and the use of feedback based position control including roll stabilization [24]. However, this assumption may not be valid in higher sea states. Therefore, we determine collision zones around obstacles by computing the region of inevitable collision [25] and replan the trajectory with a high frequency to account for the uncertainty in the vehicle's motion. In addition, we assume that the combined set of sensors (e.g., lidar, stereo cameras, and radar), digital nautical charts, and Kalman filtering [26] will provide us reasonable state estimation of obstacles as well as the USV.

Since the target boat may be moving with a high speed, the trajectory needs to be computed sufficiently fast and

still preserve its dynamical feasibility in the close vicinity to the USV. The quality of the computed trajectory as well as the computational performance of the planner depends on the number and type of control actions and dimensionality of the underlying state space in different planning stages. A high-quality trajectory may be close to optimum in terms of its execution time but may take a long time to be computed on a given machine. On the other hand, a low-quality trajectory will be computed very quickly but may be longer with unnecessary detours. Both the cases would thus lead to a poor overall follow capability. Hence, it was necessary to sacrifice allowable level of optimality by superimposing high and low dimensional state spaces and utilizing a multi-resolution control action set. In particular, the dimensionality of the state space as well as the number of control actions is reduced with the increase of the distance from the USV towards the target. We have conducted a detailed empirical analysis to find a trade-off between fast computation and trajectory length.

This paper builds on our previous work [27], where we introduced a lattice-based planner for computation of a dynamically feasible, collision-free trajectory for the USV to approach a dynamically moving motion goal. We present the following new results:

1. We have developed a novel algorithm for motion goal prediction that has the capability of estimating the future pose of the target boat with increased precision. This is possible through the utilization of known target boat's dynamics as well as its action selection model that defines a probability distribution over the target's control actions.
2. We have carried out a detailed empirical analysis of the effect of varying state space dimension and control action set resolution on the computational efficiency of the planner and trajectory length.
3. We have improved the trajectory tracking technique that computes the desired speed for each segment of the trajectory, given the required arrival time to the motion goal and upper bounds on the speed of its individual segments.
4. We have carried out physical experiments using radio controlled boats to evaluate the developed planner in a real world scenario.

The outline of the paper is as follows. First, we review existing major planning and control approaches in Section 2. Second, we present the definition of the problem in Section 3, followed by an overview of the overall approach in Section 4. This overview includes a description of the developed USV system architecture (see Figure 2) that integrates all planning modules. Next, we describe the state-action space representation that is used in both motion goal prediction in Section 6 as well as in nominal trajectory planning in Section 7.1. This is followed by a description of the developed trajectory tracking technique in Section 7.2. Finally, we present simulation as well as experimental results in Section 8.

## 2 Literature Review

Here, we will provide a review of representative works related to following a moving target boat in an environment with obstacles. Our particular focus is on reviewing techniques for following a moving target that does not necessarily attempt to evade. Readers interested in the pursuit and evasion problems can refer to a comprehensive survey in [28]. This review includes techniques for target following, control and guidance systems in the USV domain, as well as trajectory planning techniques for vehicles with differential constraints.

### 2.1 Target Following

A survey of state-of-the-art approaches for target following can be found in [29]. This includes multi-vehicle motion control. In addition, the paper presents experimental validation of USV following a leader vessel by observing its path and precisely executing it using a path-following algorithm. The experiments were demonstrated on ALANIS and Charlie platforms.

High-speed straight-line tracking capability was developed in [30] to allow underactuated USV follow a moving target. Based on the guidance system previously utilized for interceptor missiles, the motion control system is composed of constant bearing guidance and velocity control schema that allows high and precise maneuverability.

A variety of advanced maneuvers for searching and tracking a target, docking, reactive collision avoidance, U-turn, course tracking, and waypoint following are implemented on the MESSIN system [12]. The system is capable of handling failures of its components through various emergency programs. In addition, the integrated path planning utilizes waypoints and motion primitives represented as circular arcs.

Cooperative control [5] is an extension to the basic follow behavior using which multiple unmanned vehicles can track their corresponding paths while keeping themselves in a formation and maintaining the required speed to track the target. In this work, path and target following primitives were evaluated using physical Aguas Vivas and DELFIMX USV platforms. In addition, simulation results generated using NetMarSyS simulator were presented

where three USVs followed their paths in a marine environment with ocean currents. The aim of the proposed solution was also to cope with communication uncertainties between surface and underwater vehicles during the follow task. Another multi-vehicle follow experiment on real platforms include [31] where a master boat is followed by two USVs.

In order to handle motion uncertainties due to ocean disturbances, a dynamic surface control and adaptive formation controller represented as a neural network was developed in [32]. Similarly, a formation control for following a master vessel while considering uncertain dynamics of the vessel can be found in [33].

### 2.2 Control and Guidance Systems

There is a broader range of literature related to the USV follow capability developed in this paper in terms of control, path and trajectory following, and obstacle avoidance.

Underactuated controller design for USVs using 3 DOF simplified dynamics models (i.e., neglecting roll, pitch, and heave motions) has been extensively explored, e.g., in [34, 35,36,37,38,39].

A PID-based control together with extended Kalman filter was used in [40] to perform basic control tasks such as straight-line following, auto-heading and speed maintaining and adaptation. The control system was evaluated on the autonomous USV prototype CNR-ISSIA Charlie. The MIT SCOUT kayak platforms also implemented a similar PID-based system [41] integrated within a distributed autonomy architecture for sensor adaptive control of USVs in an oceanographic sampling application [6]. The architecture includes a behavior-based multiple objective function control model allowing behavior coordination based on Interval Programming (IP). The developed behaviors include waypoint, stationKeep, constantSpeed, and the Timer.

The problem of following a path or trajectory is closely related to the target following problem in an obstacle-cluttered environment as it is one of its essential components. Path and trajectory following techniques have been mostly adapted from ground and aerial vehicles domains. The goal of path following techniques is to minimize the distance and heading tangential error between the vehicle and the path. In general, the user of the system specifies a particular velocity profile to be executed by the vehicle [42]. In trajectory following, each waypoint also specifies a time of arrival constraint for the vehicle.

A summary of path-following algorithms for an underactuated USV can be found in [43]. The paper also introduces a nonlinear Lyapunov-based control law that minimizes path-following error. Speed adaptation that considers curvature of the path as well as steering prediction is realized through a developed heuristic technique. The developed system was executed on Charlie USV and evaluated using metrics that measured path following precision as well as the total surface area between the executed and nominal paths.

Navigation, guidance, and control system for Springer USV was designed in [10]. The particular applications included pollutant tracking in the broad domain of environmental monitoring. Line-of-sight and waypoint navigation was used for the guidance together with tracing techniques for chemical source detection. Similarly for the ISR/IST DELFIM USV, navigation, guidance, and control system was developed [44] that is also capable of autonomous path and trajectory following, and maneuvering. The platform was used for automated marine data acquisition.

Current USVs employ global and local obstacle avoidance to move between user-specified waypoints. Obstacle avoidance techniques in the USV domain have been mostly used for guidance that is compliant with International Regulations for Preventing Collisions at Sea (COLREGs) [45, 46,47,48,49]. Waypoint navigation (i.e., using the *follow-the-carrot* technique) with reactive obstacle avoidance was developed in SPAWAR Systems Center's guidance system [50]. The system also includes A* based heuristic algorithms for global planning [51,50]. The dynamic obstacle avoidance is implemented using the velocity obstacle method [52] and computation of critical points in forward-projected regions the moving obstacles could take along their future paths. Another A* based global planning positioned within a three layered architecture was developed by Casalino *et al.* in [53].

On a higher level, the current state-of-the-art Control Architecture for Robotic Agent Command and Sensing (CARACaS) system has been developed by the Jet Propulsion Laboratory (JPL) [54,55]. The components of the system include perception, behavior, and dynamic planning (CASPER). The planning is represented by both global and local obstacle avoidance. CARACaS was tested in multiple on-water scenarios and as such is the most comprehensive system to date.

Only a few currently developed systems are able to compute dynamically feasible trajectories, e.g. [56,49,57], which are produced by our trajectory planner in the pose space. The implementation and evaluation of obstacle avoidance techniques on real systems is still very immature, especially as far as dynamic obstacle avoidance is concerned. They are highly dependent on the marine operators in case of collision threats [1].

### 2.3 Trajectory Planning Algorithms

Trajectory planning is a broad and important research area in robotics [58]. Here, we summarize only a set of representative research papers with a focus on planning under differential constraints as it is closely related to our planning approach. This particular group of planning algorithms can be divided into the following categories [59]: (1) trajectory planning based on state space sampling, (2) decoupled trajectory planning in space and time, (3) maneuver automatons (MA), (4) mathematical programming, and (5) model predictive control (MPC).

The techniques for state space sampling usually represent the vehicle's state space as as grid of cells [60]. The discretized state space is then searched for a collision-free trajectory optimizing a specified cost. A feedback policy (i.e., a navigation function) is computed over the discretized state space by a value or policy iteration algorithm of dynamic programming (DP) [58]. Another common alternative is rapidly exploring random trees (RRT) technique and its extensions [61,62,63] focused mostly on random sampling of high-dimensional configuration spaces. In order to make the planning in a continuous space feasible, interpolation is usually used as in [64].

Decoupled trajectory planning divides the search into a global search [65,66] and local search between consecutive waypoints of the generated trajectory. The A* based global search considers only the kinematic properties of the vehicle, whereas the local search solves the two-point boundary value problem [58] to produce a trajectory complying with differential constraints of the vehicle. For example, waypoint computation and RTABU search optimization techniques were combined in [65] to generate a collision-free trajectory. Another approach decouples Laplacian-based potential field technique and velocity control as presented in [67].

Trajectory planning using maneuver automatons [68] and related motion primitives [69,22,70] has lately become very popular for unmanned aerial and ground vehicles trajectory planning. This is mainly due to the discretization of a continuous state-action space into a manageable discrete space suitable for fast and dynamically feasible search.

Trajectory planning is represented as a numerical optimization problem in mathematical programming, where the kinodynamics of the vehicle imposes a set of motion constraints. The defined problem is then solved using the mixed integer linear programming, non-linear programming, or other related techniques [71,72,73,74].

Model predictive control also represents the trajectory planning problem as an optimization problem. However, the optimization is carried out only over a finite horizon which saves a substantial amount of computational time [75,76,77].

Trajectory planning for obstacle avoidance in the USV domain has been utilized by Casalino *et al.* [53]. The proposed solution is divided into static, moving, and reactive components encapsulated into a three layered architecture. The static component generates a visibility graph representation of the planning space that is searched using Dijkstra's algorithm [78]. The moving component generates a bounding box for each obstacle and generates a graph searched using A* algorithm to find the shortest, collision-free path.

Additional techniques include Soltan's trajectory planner ([79]) combined with trajectory tracking and nonlinear sliding mode based control of multiple surface vessels. Xu *et al.* introduced a receding horizon control based trajectory replanning approach where the global plan is determined using predetermined level sets from experimental runs [80]. Finally, autonomous guidance based on feedback control is developed by Sandler *et al.* in [81].

## 3 Problem Formulation

Given,

(i.) a continuous, bounded, non-empty state space $X \subset \mathbb{R}^3 \times \mathbb{S}^1$ in which each state $\mathbf{x} = [x, y, u, \psi]^T$ consists of the position coordinates $x$ and $y$, the surge speed $u$, and the orientation $\psi$ (i.e., the heading angle) about the $z$ axis of the Cartesian coordinate system (i.e., the heave, roll, and pitch motion components of the vehicle can be neglected for our application);

(ii.) the current state of the USV $\mathbf{x_U} = [x_U, y_U, u_U, \psi_U]^T$ and the moving target $\mathbf{x_T} = [x_T, y_T, u_T, \psi_T]^T$;

(iii.) an obstacle map $\Omega$ such that $\Omega(\mathbf{x}) = 1$ if $\mathbf{x} \in X_{obs} \subset X$, where $X_{obs}$ is the subspace of $X$ that is occupied by obstacles, otherwise $\Omega(\mathbf{x}) = 0$;

(iv.) a continuous, bounded, state-dependent, control action space $U_U(\mathbf{x}) \subset \mathbb{R}^2 \times \mathbb{S}^1$ of the USV in which each control action $\mathbf{u_U} = [u_d, \delta t, \phi_d]^T$ consists of the desired surge speed $u_d$, rudder angle $\phi_d$, and execution time $\delta t$. Similarly, we are given a control action space $U_T \subset \mathbb{R}^2 \times \mathbb{S}^1$ for the target boat;

(v.) an action selection model $\pi_T : X \times U_T \to [0, 1]$ for the target boat defining a probability distribution over its control actions $U_T$;

(vi.) 3 degrees of freedom parametric model of the USV $\dot{\mathbf{x}}_\mathbf{U} = f_U(\mathbf{x_U}, \mathbf{u_{U,c}})$ [82], where the thrust and moment are generated by the vehicle's actuators. The actuators take $\mathbf{u_{U,c}} = [u_c, \phi_c]^T$ as the control input, where $u_c$ is the speed of the propeller in rpm, and $\phi_c$ is the rudder

angle. Similarly, we are given the target boat dynamics $\dot{\mathbf{x}}_\mathbf{T} = f_T(\mathbf{x_T}, \mathbf{u_{T,c}})$;

Compute,

(i.) a motion goal $\mathbf{x_G} = [x_G, y_G, u_G, \psi_G]^T \in X_{free}$ together with the desired arrival time $t_G$ to reach $\mathbf{x_G}$ from $\mathbf{x_U}$, where $X_{free} = X \backslash X_{obs}$; and

(ii.) a collision-free, dynamically feasible trajectory $\tau : [0, t] \rightarrow X_{free}$ between $\mathbf{x_U}$ and $\mathbf{x_G}$. The desired surge speed $u_d$ of each trajectory segment $\mathbf{u_{U,d,k}}$ needs to be updated such that $\|\mathbf{x_{U,t_G}} - \mathbf{x_{G,t_G}}\| = 0$, where $\mathbf{x_{U,t_G}}$ and $\mathbf{x_{G,t_G}}$ represent the states of the USV at the time $t_G$.

The motion goal $\mathbf{x_G}$ and the trajectory $\tau$ are required to be recomputed in each planning cycle to keep track of the moving target, handle dynamic obstacles, and pose errors introduced due to motion uncertainty caused by the ocean environment. The state estimation is assumed to be provided by the extended Kalman filter [26].

## 4 Overview of Approach

There are three main components of the approach presented in this paper, namely, motion goal prediction (see Section 6), trajectory planning (see Section 7), and trajectory tracking (see Section 7.2). The developed navigation, guidance, and control system architecture depicting the individual components is shown in Figure 2.

We first generate a discrete, lower-dimensional representation $X_d \subset \mathbb{R}^2 \times \mathbb{S}^1$ of the continuous state space $X \subset \mathbb{R}^3 \times \mathbb{S}^1$ in which each discrete state $\mathbf{x_{d,j}} = [x_j, y_j, \psi_j]^T$ omits the surge speed component $u_j$ (see Section 5). In addition, we generate a discrete control action space representation $U_{U,d}$ of the continuous control action space $U_U$. In order to make the trajectory planning feasible, we decouple the search in the velocity space from the search for the trajectory in the pose space. We utilize the trajectory tracking component that determines the required speed for each segment of the trajectory to be able to reach a motion goal $\mathbf{x_G}$ at the required time $t_G$.

The discrete state and action space representations are used for constructing a 3D lattice structure [22] that represents the discretized planning space to be used by the developed motion goal prediction and trajectory planning algorithms. The lattice structure allows us to reduce the planning complexity by adaptive discretization (see Section 7.1.1 and 7.1.2) but at the same time is capable of capturing vehicle's dynamic constraints in a resolution sense.

The motion goal prediction algorithm estimates the state of the target boat within future planning steps by forward-projecting its control actions based on the specified probabilistic action selection model $\pi_{T,d}$. The algorithm computes a suitable location around the moving target in the form of a motion goal $\mathbf{x_G} \in X_{free}$. The motion goal is computed based on the current states of the boats $\mathbf{x_{U,d}}$ and $\mathbf{x_{T,d}}$, the obstacle map $\Omega$, and $\pi_{T,d}$. The details of this layer are described in Section 6.

The trajectory planning component receives the motion goal $\mathbf{x_G}$ and computes a collision-free trajectory $\tau$ represented as a sequence of waypoints from the current state $\mathbf{x_{U,d}}$ of the USV to the motion goal. The planner searches the 3D lattice space [22,83] (see Figure 4) to generate the nominal trajectory. The details of this component are described in Section 7.

Finally, the trajectory tracking component computes the desired surge speed for each segment of the trajectory, and sends speed and yaw control commands (i.e., required throttle and rudder positions) to the low-level controllers of the USV so that it can reliably navigate through the waypoints and arrive to the motion goal at the required time $t_G$. The details of this layer are described in Section 7.2.

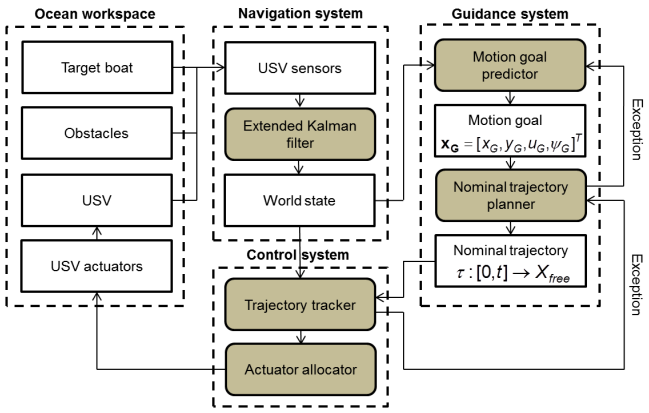



Fig. 2: Developed USV system architecture for following a moving target in a marine environment with obstacles.

## 5 State-Action Space Representation

We discretize the continuous state space $X \subset \mathbb{R}^3 \times \mathbb{S}^1$ into a finite discrete space $X_d$. A state $\mathbf{x_{d,j}} = [x_j, y_j, \psi_j]^T \in X_d$ represents a position $(x_j, y_j)$ and orientation $\psi_j$ of the USV in a global inertial frame of reference. Depending upon the resolution of discretization, $\mathbf{x_{d,j}}$ also represents a polygonal (i.e., a rectangle in case of rectangular discretization) region in $X_d$ in which the pose $[x_j, y_j, \psi_j]^T$ of the USV is assumed to be fixed. This discrete state space representation includes both position and orientation of the vehicle and

hence is useful for incorporating constraints based on the dynamics of the USV. Choosing a suitable discretization of the lattice grid required trading off the speed and resolution completeness [58] of the planner. In our approach, the discretization level was determined empirically by attempting coarse discretization first and then increasing the resolution of the grid until it satisfied our task. The two particular factors that we considered were the dimension of the vehicle and its control action set.

Similarly, we discretize the continuous control action space $U_U$ into a discrete control action set $U_{U,d}$ and map each control action $\mathbf{u_{U,d,k}} \in U_{U,d}$ to a reachable pose $[\Delta x_k, \Delta y_k, \Delta\psi_k]^T$ in the body coordinate system of the vehicle. Each control action is compliant with the vehicle's dynamics $f_U$. We pre-compute the control actions in $U_{U,d}$ using a hand-tuned PID controller. However, other control techniques such as the sliding mode [79], backstepping [84], etc. can be used [39]. An example of a discrete control action set $U_{U,d} = \{\mathbf{u_{U,d,1}}, \ldots, \mathbf{u_{U,d,5}}\}$ is shown in Figure 3a) for the vehicle used in the experiments in Section 8. The choice of the number of control actions required trading off the computational performance, resolution completeness of the planner, and maneuverability of the vehicle. We have determined the number of control actions empirically by starting with three actions and then increasing their count until it satisfied this trade-off. However, there have been few techniques published recently that directly address the design of control action sets [85,86,87,88,89] and as such this problem is still an open research area.

We define a lattice $L$ as a structure that maps $\mathbf{x_{d,j}}$ to $\mathbf{x_{d,j,k}}$ using a control action $\mathbf{u_{U,d,k}}$ for $j = 1, 2, \ldots, |S|$ and $k = 1, 2, \ldots |U_{U,d}|$. The lattice thus represents an instance of a graph with discrete states $\mathbf{x_{d,j}}$ as nodes and actions $\mathbf{u_{U,d,k}}$ as connecting arcs. Each action $\mathbf{u_{U,d,k}} \in U_{U,d}$ is used to establish a connection between dynamically reachable nodes in $L$. For a node $\mathbf{x_{d,j}} = [x_j, y_j, \psi_j]^T \in X_d$, the neighbor corresponding to an action $\mathbf{u_{U,d,k}} = [\Delta x_k, \Delta y_k, \Delta\psi_k]^T \in U_{U,d}$ is given by Equation 1. The lattice structure captures dynamics constraints in the form of dynamically feasible connecting edges between the nodes unlike orthogonal grids [58,22]. The lattice can also be geometrically viewed as multiple X-Y planning layers representing 2D planning spaces with a fixed orientation $\psi$ of the USV as shown in Figure 4.

$$\mathbf{x_{d,j,k}} = \begin{bmatrix} x_j \\ y_j \\ \psi_j \end{bmatrix} + \begin{bmatrix} cos\Delta\psi_j & -sin\Delta\psi_j & 0 \\ sin\Delta\psi_j & cos\Delta\psi_j & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta y_k \\ \Delta\psi_k \end{bmatrix} \tag{1}$$

It is necessary to appropriately design the control action set $U_{U,d}$ (i.e., the number, type, and duration of control actions) such that the required efficiency as well as the quality of the nominal trajectory is achieved. In our approach, we manually designed the control action set using a 3 degrees of freedom simulation model of the USV [83,23, 90]. To generate more complex actions, gradient-based optimization techniques [91] may be used. The control action set can also be generated automatically using special techniques [85,86,87,88,89] that attempt to balance between the computational efficiency and quality of the generated trajectories.

The simulator is based upon a simplified 3-DOF model (see Equation 2 adapted from Fossen [82]).

$$\begin{aligned} m_{11}\dot{u} + d_{11}u &= T \\ m_{22}\dot{v} + d_{22}v &= 0 \\ m_{33}\dot{r} + d_{33}r &= M \\ \dot{x} &= ucos\psi - vsin\psi \\ \dot{y} &= usin\psi + vcos\psi \\ \dot{\psi} &= r \end{aligned} \tag{2}$$

where $d_{ii}$ represents the hydrodynamic damping, $m_{ii}$ represents added mass terms, $T$ represents thrust due to propeller actuation, and $M$ represents the moment due to rudder actuation. The added mass terms $m_{11}, m_{22}$ and $m_{33}$ are computed using Equation 3 [39]. In this equation, $L$, $W$, and $B$ are the length, width, and draft of the boat hull, respectively.

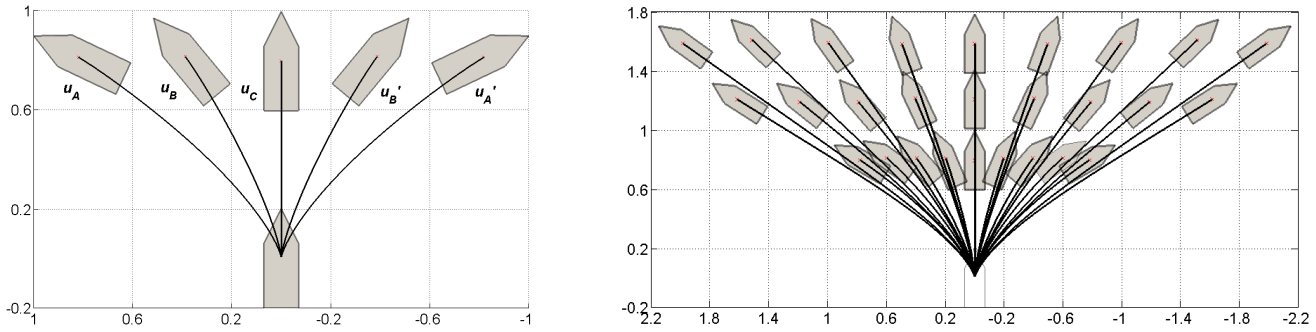
$$\begin{aligned} m_{11} &= 1.05m \\ m_{22} &= m + 0.5\rho\pi D^2 L \\ m_{33} &= \frac{m(L^2 + W^2) + 0.05mB^2 + 0.5\rho\pi D^2 L^3}{12} \end{aligned} \tag{3}$$

There are many actuator models available for the USVs for computing actuation thrust $T$ and moment $M$ and can be substituted into Equation 2 [82,92]. We used an actuation model [92] given in Equation 4.

$$\begin{aligned} T &= K_1 \parallel u_c \parallel u_c \\ M &= K_2 K_1 \parallel u_c \parallel u_c \phi_c \end{aligned} \tag{4}$$

where $K_1$ and $K_2$ are actuator constants, $u_c$ is the propeller's rpm, and $\phi_c$ is the rudder angle.

The target boat may have different motion characteristics than the USV. Hence, similarly as we discretize the state-action space of the USV, we discretize the state-action space for the target boat. This results in a discrete state space $X_{T,d}$ and action space $U_{T,d}$. This discretization is mostly needed for motion prediction of the target boat and motion goal computation as described in Section 6.




(a) A sparse control action set containing 5 actions.

(b) A dense control action set containing 27 actions.

Fig. 3: Action sets for building the lattice data structure. The dense action set discretizes the action space into 27 control actions. This set is particularly useful when a higher number of obstacles are present or demand on optimality is stricter. However, the dense action set makes the planning speed slower due to more number of connected nodes in the planning space, i.e., in the lattice.

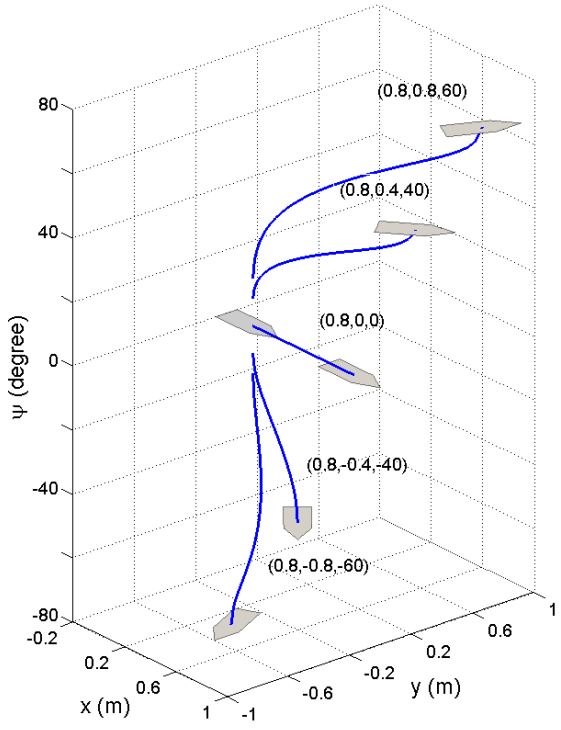



Fig. 4: Planning space representation.

**Algorithm 1** ComputeMotionGoal()

**Input:** The current poses $\mathbf{x_{T,d}}$ and $\mathbf{x_{U,d}}$ of the target boat and USV, a probabilistic model of the target boat $\pi_{T,d}$, and a map of obstacles $\Omega$.

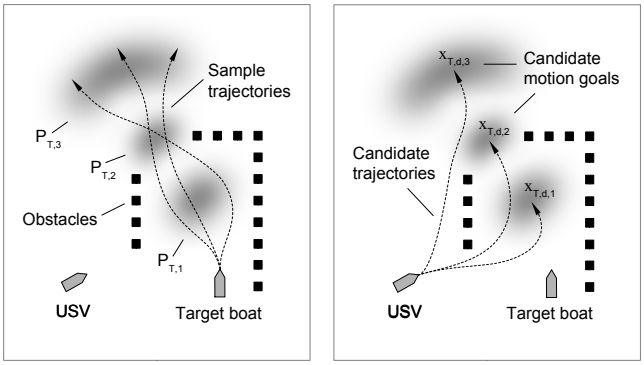
**Output:** A desired motion goal $\mathbf{x_G}$ and arrival time $t_G$.

1: Let $R_T$ be a set of $N$ randomly generated trajectories for the target boat, where each trajectory $\tau_{T,i} \in R_T$ begins at state $\mathbf{x_{T,d}}$ and time $t_0$ and continues until time $t_k$, such that each sampled action $\mathbf{u_{T,d,j}}$ is selected with probability $\pi_{T,d}(\mathbf{x_{T,d,i}}, \mathbf{u_{T,d,j}})$.
2: **for** each time point $t_i \in \{t_1, t_2, \dots t_k\}$ **do**
3:  **for** each trajectory $\tau_{T,j} \in R_T$ **do**
4:  Increment $P_{T,i}(x_{i,j}, y_{i,j})$ by $1/N$, where $(x_{i,j}, y_{i,j})$ is the location of $\tau_{T,j}$ at time $t_i$.
5:  **end for**
6:  Smooth $P_{T,i}$ by applying an $m \times m$ Gaussian kernel.
7:  Let $(x_i^*, y_i^*)$ be the location that maximizes $P_{T,i}(x_i^*, y_i^*)$ and let $\psi_i^*$ be the mean orientation at $(x_i^*, y_i^*)$ across all trajectories in $R_T$ at time $t_i$.
8:  Let $\mathbf{x_{G,i}} \in X_{U,d} = [x_j, y_j, u_U, \psi_j]$ be a candidate motion goal for the USV which minimizes the distance to the projected state, $|(x_i^*, y_i^*) - (x_j, y_j)| + \alpha\Delta(\psi_i^*, \psi_j)$
9: **end for**
10: Let $\mathbf{x_G}$ equal the candidate motion goal $\mathbf{x_{G,i}} \in X_G$ that minimizes the cost function $c(\mathbf{x_G})$ and ensures a collision-free path up to time $T_{ric}$
11: Compute a desired arrival time $t_G(\mathbf{x_G})$ such that the USV will arrive at $\mathbf{x_G}$ shortly after the target boat
12: **return** $(\mathbf{x_G}, t_G(\mathbf{x_G}))$.

## 6 Motion Goal Prediction

To maintain a suitable distance between the USV and the target boat, the motion goal predictor selects a desired motion goal $\mathbf{x_G}$ and arrival time $t_G$. This is done by estimating the future poses of the target boat and then evaluating several candidate motion goals for the USV. Since the motion goal must be computed relatively quickly, we search for a sub-optimal solution by combining Monte-carlo sampling and heuristic evaluation techniques. The overall process is described in Algorithm 1.

The action selection model $\pi_{T,d} : X_{T,d} \times U_{T,d} \to [0, 1]$ defines a probability distribution over the target's discretized control actions at each pose $\mathbf{x_{T,d}} \in X_{T,d}$. To explore the future poses of the target, we sample $N$ random trajectories $\tau_{T,j} \in R_T$ starting from the target's current pose and forward-projecting the target's actions up to some finite time horizon $t_k$. During the sampling process, each control




(a) (b)

Fig. 5: Motion goal prediction steps (a) Probability distributions $P_{T,1}$, $P_{T,2}$ and $P_{T,3}$, generated by sampling future target trajectories at several time points. (b) Candidate motion goals generated by selecting the most probable target pose.

action $\mathbf{u_{T,d,k}}$ is selected with probability $\pi_{T,d}(\mathbf{x_{T,d,i}}, \mathbf{u_{T,d,k}})$. Sampled trajectories that lead to a collision with an obstacle are discarded from $R_T$.

By recording the poses reached by the target boat along each sampled trajectory, it is possible to estimate the probability $P_{T,i}(x_j, y_j)$ that the target will be at position $(x_j, y_j)$ at time $t_i$. A two-dimensional Gaussian kernel is used to smooth out the probability distribution represented by $P_{T,i}$, as illustrated in Figure 5. We also compute $\psi_{T,i}(x_j, y_j)$, the mean orientation at $(x_j, y_j)$ across all samples at time $t_i$.

For each time point $t_i \in \{t_0, t_1, \dots t_k\}$, the future position of the target boat can be approximated by

$$(x_i^*, y_i^*) = \arg \max_{(x_j, y_j)} P_{T,i}(x_j, y_j). \quad (5)$$

Similarly, let $\psi_i^* = \psi_{T,i}(x_i^*, y_i^*)$ approximate of the future orientation of the target boat. We select the candidate motion goal $\mathbf{x_{G,i}} \in X_{U,d} = [x_i, y_i, u_T, \psi_i]^T$ that is closest to the projected target pose at time $t_i$, such that

$$\mathbf{x_{G,i}} = \arg \min_{\mathbf{x_{G,j}} \in \mathbf{X_{U,d}}} |(x_i^*, y_i^*) - (x_j, y_j)| + \alpha \Delta(\psi_i^*, \psi_j) \quad (6)$$

where $\alpha\Delta(\psi_i^*, \psi_j)$ is the interior angle between $\psi_i^*$ and $\psi_j$ weighted by some parameter $\alpha$.

Let $g(\mathbf{x_{G,i}})$ be the amount of time it takes the USV to travel from its current location $\mathbf{x_{U,d}}$ to the candidate goal $\mathbf{x_{G,i}}$. This can be computed by the A* heuristic search process described in Section 7.

We define the cost function for motion goal $\mathbf{x_{G,i}}$ as,

$$c(\mathbf{x_{G,i}}) = \begin{cases} \delta^{-i}(g(\mathbf{x_{G,i}}) - t_i + 1), & \text{if } g(\mathbf{x_{G,i}}) > t_i \\ \delta^{-i}, & \text{otherwise,} \end{cases} \quad (7)$$

where $g(\mathbf{x_{G,i}}) - t_i$ is the estimated difference in arrival time between the USV and target boat, and $\delta \in [0, 1]$ is a discount factor. The smaller that $\delta$ is, the stronger the bias towards earlier goals. From the set of candidate motion goals $X_G$, a final motion goal $\mathbf{x_G}$ is selected such that the cost function is minimized,

$$\mathbf{x_G} = \arg \min_{x_{G,i} \in X_G} c(\mathbf{x_{G,i}}). \quad (8)$$

Given the final motion goal $\mathbf{x_G}$, the desired arrival time $t_G$ is determined by,

$$t_G(\mathbf{x_G}) = \begin{cases} g(\mathbf{x_G}), & \text{if } g(\mathbf{x_G}) > t_i + t_{lag} \\ t_i + t_{lag}, & \text{otherwise,} \end{cases} \quad (9)$$

where $t_{lag}$ is the desired amount of time that the USV should follow behind the target boat. This ensures that the USV will reduce its speed when it can afford to do so, such as when it is already close the target boat.

Due to the cost function $c(\mathbf{x_G})$, the algorithm only considers motion goals which guarantee a collision-free trajectory for the USV up to some time $T_{ric}$. States which lead to an inevitable collision are given an infinite cost, as described in detail in Section 7.1. For simplicity, we may use sampled poses from $R_T$ as motion goals if the dynamics of the USV and target boat are equivalent, since the trajectories in $R_T$ are guaranteed to be collision free. If no collision-free motion goals are found, the algorithm will default to the currently assigned motion goal.

## 7 Trajectory Planning and Tracking

### 7.1 Nominal Trajectory Planning

The nominal trajectory planner generates a collision free trajectory $\tau : [0, t] \to X_{free}$ between the current state $\mathbf{x_U}$ of the USV and the motion goal $\mathbf{x_G}$ in the obstacle field $\Omega$. The trajectory $\tau$ is computed by concatenating control actions in $U_{U,d}$ which are designed a-priori. The planner is based on A* heuristic search [21] over the lattice. The planner associates $\mathbf{x_G}$ and $\mathbf{x_U}$ with their closest corresponding states $\mathbf{x_{G,d}}$ and $\mathbf{x_{U,d}}$ in the lattice $L$. Similarly, the planner maps the state of every obstacle $\mathbf{x_O} \in X$ to its closest corresponding state $\mathbf{x_{O,d}} \in X_d$.

We also extract the region of inevitable collision [25] in $X_d$ for the given obstacle field $\Omega$. In general, the region of inevitable collision is defined as a set of states that lead to a collision regardless of a control action the vehicle executes. We avoid searching for the trajectory in this region (i.e., during the search, we do not expand candidate states that fall into this region) by assigning an infinite cost to all its states.

In this paper, we approximate the region of inevitable collision by determining a set of lattice nodes from which the vehicle will collide with an obstacle after a specified

time horizon. In the computation, we assume that the vehicle will continue in its course by moving straight and with maximum speed. This is generally a valid assumption as the length scale of the control action is much smaller and possibility of sudden steering is less in such a small distance. In addition, our 3D pose based representation takes orientation of the USV into account. A USV that is very close to an obstacle can be said to be inside the region of inevitable collision if it is approaching the obstacle. On the other hand, if the USV is moving away from the obstacle from the same pose then it can be said to be out of the region of inevitable collision.

More formally, for each lattice node $\mathbf{x_{d,j}} \in X_d$ that represents a pose $[x_j, y_j, \psi_j]^T$ of the USV, we use the dynamic model of the vehicle to determine a lattice node $\mathbf{x'_{d,j}}$ that represents the vehicle's pose after the time $T_{ric}$. If the forward projected lattice node $\mathbf{x'_{d,j}}$ lies on an obstacle, i.e., $\Omega(\mathbf{x'_{d,j}}) = 1$, then we label $\mathbf{x_{d,j}}$ to be lying inside the region of inevitable collision (see Equation 10).

$$RIC(\mathbf{x_{d,j}}, u_{max}, T_{ric}) = \begin{cases} 1, \text{ if } \mathbf{x'_{d,j}} \in \Omega \\ 0, \text{ otherwise,} \end{cases} \tag{10}$$

In Equation 10, $\mathbf{x'_{d,j}}$ is the lattice node closest to the actual state $\mathbf{x'_j} = \mathbf{x_j} + T_{ric} u_{max} [cos\psi_j, sin\psi_j, 0, 0]^T$, and $u_{max}$ is the maximum surge speed of the vehicle.

The time horizon $T_{ric}$ for determining the region of inevitable collision can be set by the user of the planning algorithm depending upon the preferred allowable risk. In particular, in this paper, we set $T_{ric}$ experimentally based on a state transition model of the USV (i.e., a probabilistic state transition function) which was developed in our previous work [83]. The state transition model gives us a probability distribution over future states of the vehicle given its initial state, a control action set, and sea state. We developed this model using Monte Carlo runs of a high-fidelity 6 DOF dynamics simulation of interaction between the USV and ocean waves. Our simulator can handle any USV geometry, dynamics parameters, and sea state. This approach for computing the region of inevitable collision allows us to more precisely define the size of the collision zones and thus make the planner less conservative when planning in narrow regions.

During the search for the trajectory, lattice nodes are incrementally expanded towards $\mathbf{x_{G,d}}$ in the least-cost fashion according to the trajectory cost $f(\mathbf{x_d}) = g(\mathbf{x_d}) + h(\mathbf{x_d})$, where $g(\mathbf{x_d})$ is the optimal cost-to-come from $\mathbf{x_{U,d}}$ to $\mathbf{x_d}$, and $h(\mathbf{x_d})$ is the heuristic cost-to-go between $\mathbf{x_d}$ and $\mathbf{x_{G,d}}$. The admissible heuristic cost function $h(\mathbf{x_d})$ (i.e., not allowed to overestimate the actual cost of the trajectory) reduces the total number of expanded nodes in the lattice as per the A* graph search algorithm that guarantees optimality of the computed plan.

The cost function was designed in terms of trajectory length. The cost-to-come $g(\mathbf{x_d})$ represents the total length of a trajectory between the current state $\mathbf{x_{U,d}}$ of the vehicle and $\mathbf{x_d}$ and is computed as $g(\mathbf{x_d}) = \sum_{k=1}^{K} l(\mathbf{u_{U,d,k}})$ over $K$ planning stages, where $l(\mathbf{u_{U,d,k}})$ is the length of the projected control action $\mathbf{u_{U,d,k}} \in U_{U,d}$. If the control action $\mathbf{u_{U,d}}$ takes the vehicle to a collision state $\mathbf{x_{col,d}}$ (i.e., the state for which $\Omega(\mathbf{x_{col,d}}) = 1$ or $RIC(\mathbf{x_{col,d}}, u_{max}, T_{ric}) = 1$), then $l(\mathbf{u_{U,d}})$ is set to $\infty$. Each control action is sampled using intermediate waypoints. These waypoints are then used for collision checking. The heuristic cost $h(\mathbf{x_d})$ is represented as the Euclidean distance between $\mathbf{x_d}$ and $\mathbf{x_{G,d}}$. Alternatively, according to [22], one can utilize a precomputed look-up table containing heuristic costs of trajectories between different pairs of states in an environment without obstacles.

The dynamically feasible trajectory $\tau$ is a sequence $\{\mathbf{u_{U,d,1}}, \mathbf{u_{U,d,2}}, \ldots, \mathbf{u_{U,d,K}}\}$ of precomputed atomic control actions, where $\mathbf{u_{U,d,k}} \in U_{U,d}$ for $k = 1, \ldots, K$. This sequence of control actions is then translated into a sequence of $K+1$ waypoints $\{\mathbf{w_{d,1}}, \ldots, \mathbf{w_{d,K+1}}\}$ leading to the motion goal $\mathbf{x_{G,d}}$. Thus, the generated nominal trajectory is guaranteed to be dynamically feasible and ensures that the USV will be able to progressively reach the waypoints in a sequence without substantial deviations, which otherwise may lead to collisions.

In case of the described *follow* task, the motion goal keeps changing dynamically and hence the trajectory planner needs to compute the plans repeatedly. For this reason, the planner must be very fast and efficient. For the very requirement of following the target boat in a cluttered space, the planning space must be 3D pose based. A higher dimensional planning space may significantly slow down the trajectory planner. For this reason, we enhanced the computational performance of the planner by (a) superimposing higher and reduced dimension state spaces, and (b) utilizing a control action set with multiple levels of resolution. These two extensions are explained in detail in the following two subsections.

#### *7.1.1 Superimposition of Higher and Reduced Dimension State Spaces*

Superimposing pose based 3D lattice and position based 2D grid state space representations can improve planner performance by sacrificing allowable level of optimality. At closer distances (i.e., $g(\mathbf{x_d}) < \beta$) from the initial state $\mathbf{x_{U,d}}$, the planning space is chosen to be a 3D lattice, whereas at a farther distance (i.e., $g(\mathbf{x_d}) \geq \beta$) the planning space is chosen to be a 2D grid. This scheme is useful

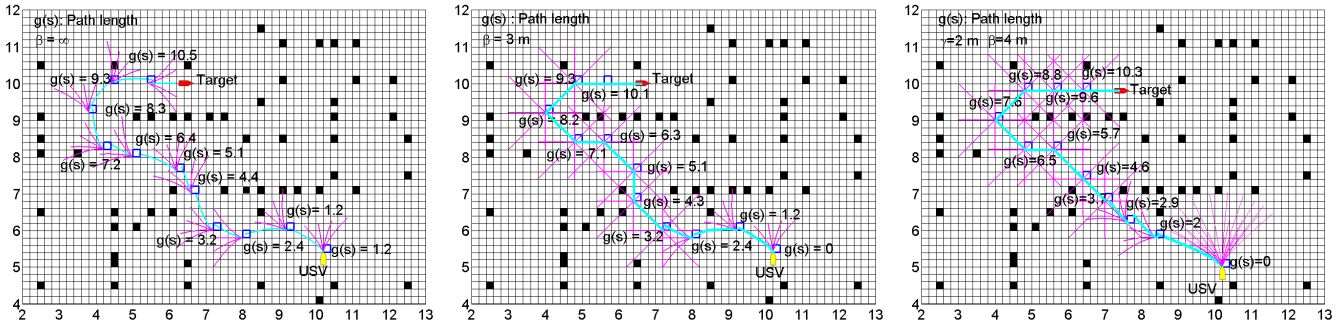



Fig. 6: (a) A trajectory computed by searching entirely in 3D pose state space ($\beta = \infty$). (b) A trajectory computed by searching partially in 3D pose state space ($\beta = 3$ m) and 2D position state space. (c) A trajectory computed using a dense action set with 27 control actions up to the trajectory length of $\gamma = 2$ m with $\beta = 4$ m. The boat length is 0.6 $m$.

as trajectory waypoints in a closer vicinity to the USV must be dynamically feasible and hence should be on a 3D lattice. However, farther waypoints can be tolerated to be on a rectangular grid. This simplification may, however, lead to the computation of a sub-optimal trajectory plan.

The threshold distance $\beta$ can be set by the user of the planning system depending upon the need of performance improvement and tolerance to trajectory sub-optimality. In order to superimpose the 2D and 3D representations, we have designed a neighbor function (see Equation 11). The 2D control action set $U^{2D}_{U,d}$ unlike the 3D control action set $U_{U,d}$ has only 2D actions with their terminal orientations along the $x$ axis.

$$neighbor(\mathbf{x_d}) = \begin{cases} \{\mathbf{x_d} + \mathbf{u_{U,d,k}} | \mathbf{u_{U,d,k}} \in U_{U,d}\}, & \text{if } g(\mathbf{x_d}) < \beta \\ \{\mathbf{x_d} + \mathbf{u^{2D}_{U,d,k}} | \mathbf{u_{U,d,k}} \in U^{2D}_{U,d}\}, & \text{otherwise.} \end{cases} \tag{11}$$

The above neighbor function restricts the search to 2D space when the trajectory length $g(\mathbf{x_d})$ increases beyond $\beta$. Figure 6a) shows a trajectory being searched completely in 3D pose space, where the parameter $\beta$ is set to infinity. The trajectory being searched for the case when $\beta = 7$ m is depicted in Figure 6b). As soon as the length of the trajectory gets greater than this threshold (i.e., $\beta > 7$), the search continues in 2D position space.

### *7.1.2 Multi-resolution Control Action Set*

In the present problem of following a moving target boat, the generated trajectory is useful only up to a limited time horizon. This is because as the USV follows the trajectory, the target boat also moves and thus a large portion of the original trajectory can quickly become obsolete. This necessitates recomputation of the trajectory by the USV, which implies that the trajectory can be of smaller accuracy at a farther distance from the USV. By using a dense action set (see Figure 3b) in close proximity (i.e., $g(\mathbf{x_d}) \leq \gamma$) to the USV and a sparse action set (see Figure 3a) at a farther distance (i.e., $g(\mathbf{x_d}) > \gamma$), the computation time can be significantly reduced. This, however, may result in a highly suboptimal trajectory. So the user of the system needs to carefully tune $\gamma$ in order to get the best balance between the speed of computation and quality of the generated trajectory.

The combined influence of the parameters $\beta$ (introduced in Section 7.1.1) and $\gamma$ is shown in Figure 6c. In this case, the planner searches through a graph connected using a dense, dynamically feasible control action set up to the trajectory length $\gamma$, then it switches to using a sparse, dynamically feasible action set for $max(0, \beta - \gamma)$ distance, and finishes the search through 2D position space for the rest of the trajectory length.

## 7.2 Nominal Trajectory Tracking

We have developed a controller to track the computed trajectory. The controller is based on the classic Line-Of-Sight (LOS) algorithm [82,93] for driving the USV between consecutive waypoints. According to the LOS algorithm, the vehicle always heads towards the next waypoint when it gets within a *circle-of-acceptance* of the current waypoint. During following the waypoints, the controller is not responsible for considering obstacle avoidance, as the set of waypoints generated by the planner is guaranteed to be in $X_{free}$ (see Section 7.1).

**Algorithm 2** ComputeDesiredSpeed(): Compute desired surge speed for all segments of the trajectory $\tau$ so that the USV can reach its motion goal $x_G$ at the required time $t_G$.

**Input:** A trajectory $\tau = \{\mathbf{u_{U,d,k}}\}_{k=1}^{K}$ that defines the maximum allowable speed $u_{d,k,max}$ for each of its segments $\mathbf{u_{U,d,k}}$, and the required arrival time $t_G$ to the motion goal $x_G$.

**Output:** Desired surge speeds $V_d = \{u_{d,k}\}_{k=1}^{K}$ for all trajectory segments $\{\mathbf{u_{U,d,k}}\}_{k=1}^{K}$.

1: Compute initial desired surge speeds $V_d = \{u_{d,k}\}_{k=1}^{K}$ for all segments of the trajectory $\tau$. The initial speed is computed as $u_{d,k} = L/t_G$, where $L = \sum_{k=1}^{K} l(\mathbf{u_{U,d,k}})$ is the total length of the trajectory and $l(\mathbf{u_{U,d,k}})$ is the length of the trajectory segment $\mathbf{u_{U,d,k}}$.
2: Let $Q$ be the priority queue containing the trajectory segments $\{\mathbf{u_{U,d,k}}\}_{k=1}^{K}$ in ascending order according to their maximum allowable surge speed $u_{d,k,max}$.
3: **while** $Q$ not empty **do**
4: Remove a segment $\mathbf{u_{U,d,k}} \leftarrow Q$.First() from $Q$ with the minimum allowable surge speed $u_{d,k,max}$.
5: Compute the time lost $t_{lost} = t_{d,k,max} - t_{d,k}$ when traversing $\mathbf{u_{U,d,k}}$ with $u_{d,k,max}$. The time needed to traverse the segment $\mathbf{u_{U,d,k}}$ under the constraint $u_{d,k,max}$ is $t_{d,k,max} = l(\mathbf{u_{U,d,k}})/u_{d,k,max}$, whereas $t_{d,k} = l(\mathbf{u_{U,d,k}})/u_{d,k}$ is the time it takes to traverse the segment with the desired speed.
6: **if** $t_{lost} > 0$ **then**
7: Set $u_{d,k} \leftarrow u_{d,k,max}$ in $V_d$.
8: Set $u_{d,l} = l(\mathbf{u_{U,d,l}})/(t_{d,l} - t_{lost}/|Q|)$ for all remaining segments $l \neq k$, where $t_{d,l} = l(\mathbf{u_{U,d,l}})/u_{d,l}$.
9: **end if**
10: **end while**
11: **return** $V_d$.

The boat dynamics considered in this paper is underactuated and hence its speed in the sway direction cannot be controlled directly. The control required to follow waypoints must be divided between surge speed and heading controllers. We accomplish this by using a PID based cascaded surge speed $u$ and yaw $\psi$ rate controllers. The heading controller produces corresponding rudder commands. We assume that the roll and pitch is maintained to zero.

The desired heading angle at $i^{th}$ time instance is calculated using $\psi_{U,d} = atan2(y_i - y, x_i - x)$, where $[x, y]^T$ is the current position of the USV and $\mathbf{w_{d,i}} = [x_i, y_i, u_i]^T$ is the current active waypoint, where $u_i$ is the desired speed for reaching this waypoint. After this, the yaw error is computed using $e_{\psi_U} = \psi_{U,d} - \psi_U$, where $\psi_U$ is the current orientation of the USV. The yaw error is then used by the PID controller to obtain the control action, i.e., the rudder angle $\phi_i$.

$$\phi_i = P_{yaw} e_{\psi_U} + I_{yaw} \int e_{\psi_U} dt + D_{yaw} \frac{de_{\psi_U}}{dt} \tag{12}$$

In Equation 12, $P_{yaw}$, $I_{yaw}$, and $D_{yaw}$ are proportional, integral, and derivative parameters of the PID controller, respectively.

Due to the discretization of the state and control action spaces, there may occur small gaps between two consecutive control actions in the computed trajectory (see Fig. 6a). In order to ensure reliable tracking of such trajectory, we define a region of acceptance around each waypoint and increase the size of the collision zones around the obstacles to minimize the probability of a collision. This also resolves other possible tracking problems that may arise because of the approximation of USV's dynamics when computing the control action set for the trajectory planner. Moreover, this also prevents creation of loops when the vehicle is not able to precisely reach a given waypoint. It is possible to design control actions such that they connect seamlessly. However, this requires substantial design effort. Lastly, since the trajectory is recomputed with a high frequency and the controller always follows the first few, dynamically-reachable waypoints along the trajectory, the USV will not experience the discontinuous transition between 3D and 2D lattices (see Section 7.1).

In the follow task, the controller must be able to handle variations in the surge speed during steep turns. Each segment $\mathbf{u_{U,d,k}}$ of the trajectory $\tau$ thus defines the maximum allowable surge speed $u_{d,k,max}$ of the vehicle. In addition, the USV cannot immediately decrease its surge speed before turning, rather it should start decelerating in advance to attain the desired speed near the turn.

In our experiments, we selected a higher speed of 0.8 $ms^{-1}$ for straight segments of the trajectory while a smaller speed of 0.3 $ms^{-1}$ for turns. We determine the distance $d_{decel}$ required for the USV to decelerate to a given surge speed using physical experiments. Figure 7 shows the result of an experimental test to determine $d_{decel}$. In this test, we controlled the boat to go in a straight line along the $x$ axis with a surge speed of 0.8 $ms^{-1}$. At a distance of 8.5 $m$, the boat was commanded to reduce its surge speed to 0.3 $ms^{-1}$. We used PID controller for controlling the surge speed. We determined that the boat needs to run for about $d_{decel} = 1.5$ $m$ in order to decelerate to the desired speed of 0.3 $ms^{-1}$.

We adjust the desired surge speed $u_{d,k}$ of each trajectory segment $\mathbf{u_{U,d,k}}$ (i.e., the desired speed for reaching each waypoint $\mathbf{w_{d,k}}$) so that the USV arrives to $\mathbf{x_G}$ at the required time $t_G$. We compute the desired speed iteratively (see Algorithm 2) given the maximum allowable surge speed $u_{d,k,max}$ constraint defined for each segment of the trajectory. We start with the segment with the minimum allowable speed, compute the time lost while traversing this segment, and uniformly increase the speed of other segments to compensate for the time loss. The algorithm

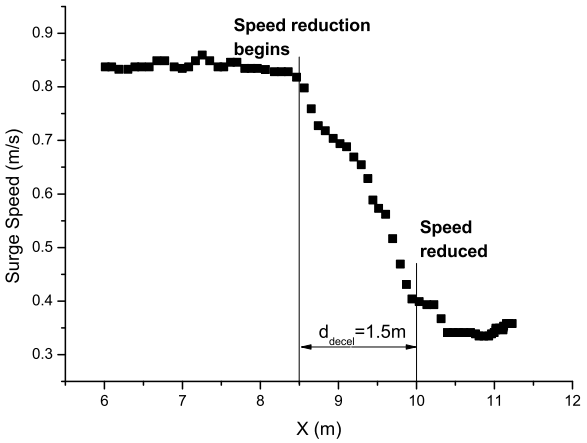



Fig. 7: The boat is commanded to go along $x$ axis up to the distance of 8.5 $m$ with the surge speed of 0.8 $ms^{-1}$, after which the surge speed is lowered to 0.3 $ms^{-1}$ resulting in the deceleration distance $d_{decel}$ of 1.5 $m$.

continues until no segment is left which requires adjusting its speed.

The algorithm is not optimal since it does not account for the speed transitions between the segments. However, since the trajectory is supposed to be recomputed with a high frequency, the desired speed of the segments is gradually adjusted with each newly computed trajectory and the state of the vehicle.

Once the desired surge speed $u_{d,k}$ is computed for all segments of the trajectory $\tau$, then the error for the speed controller is computed using $e_u = u_{d,k} - u$, where $u$ is the current surge speed of the vehicle. The speed error $e_u$ is then used in the surge PID controller to obtain the control action, i.e., the throttle control voltage $V_{throttle}$.

$$V_{throttle} = P_u e_u + I_u \int e_u dt + D_u \frac{de_u}{dt} \tag{13}$$

Similarly as in Equation 12, the parameters $P_u$, $I_u$, and $D_u$ in Equation 13 represent proportional, integral, and derivative terms of the PID controller, respectively.

## 8 Simulation and Experimental Results

### 8.1 Numerical Experimental Setup for Evaluation of Trajectory Planner Parameters

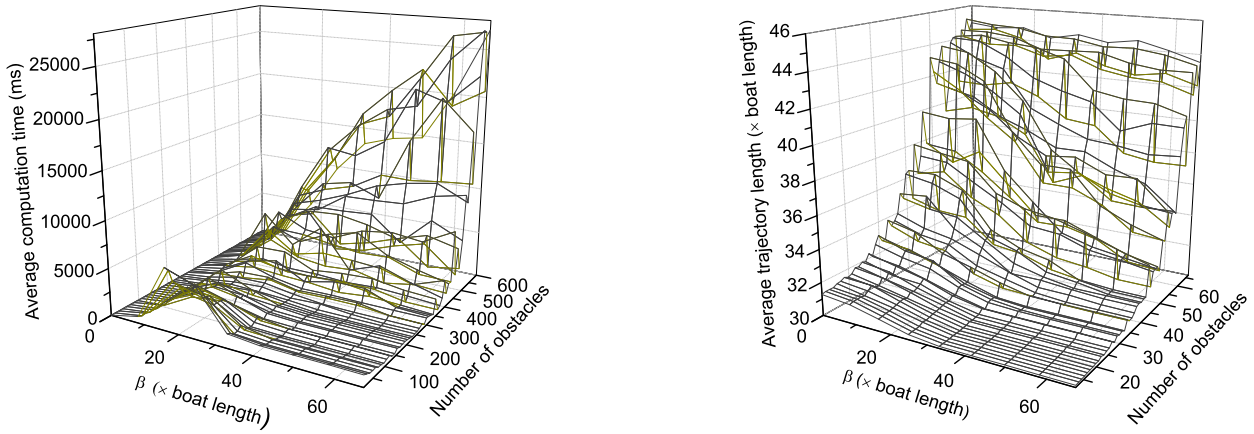
We conducted simulation experiments in order to test the influence of $\beta$ and $\gamma$ parameters on the trajectory length and planner performance. For these experiments, we generated 50 obstacle densities with the increasing number of obstacles from 12 to 600 in the increments of 12. For each obstacle density, we created 25 randomly generated cases. So we acquired $50 \times 25 = 1250$ obstacle scenes with 50 different obstacle densities. In addition, we generated 50 instances of initial and goal locations of the USV for each obstacle scene, resulting in the total of $50 \times 25 \times 50 = 62500$ test cases. For each test case, we computed a trajectory and recorded its length and computation time using the developed trajectory planner (see Section 7.1). In the presented results, the parameters $\beta$ and $\gamma$, trajectory length, and the dimension of the environment are expressed as multiples of boat length that was set to 0.6 $m$. The dimension of the environment was set to $33.3 \times 33.3$ boat lengths (i.e., $20 \times 20$ $m$).

### 8.2 Evaluation of $\beta$ Parameter

In order to test the influence of the parameter $\beta$ on the computational time and trajectory length, we varied its value from 0 to 67 boat lengths (i.e., 0 to 40 $m$ given the boat length of 0.6 $m$) in the increment of 8 (i.e., 5 $m$). Corresponding to each value of $\beta$, we generated a trajectory for each of the 62500 cases as described above. This led to $9 \times 62500 = 562500$ evaluations in total. This included computation of $25 \times 50 = 1250$ trajectories for each value of $\beta$ and a specific obstacle density. The plot in Figure 8a) shows the variation of the computation time averaged over 1250 samples for each value of the $\beta$ parameter and obstacle density. Similarly as in the previous subsection, the plot in Figure 8b) shows the variation of average trajectory length as multiples of boat length.

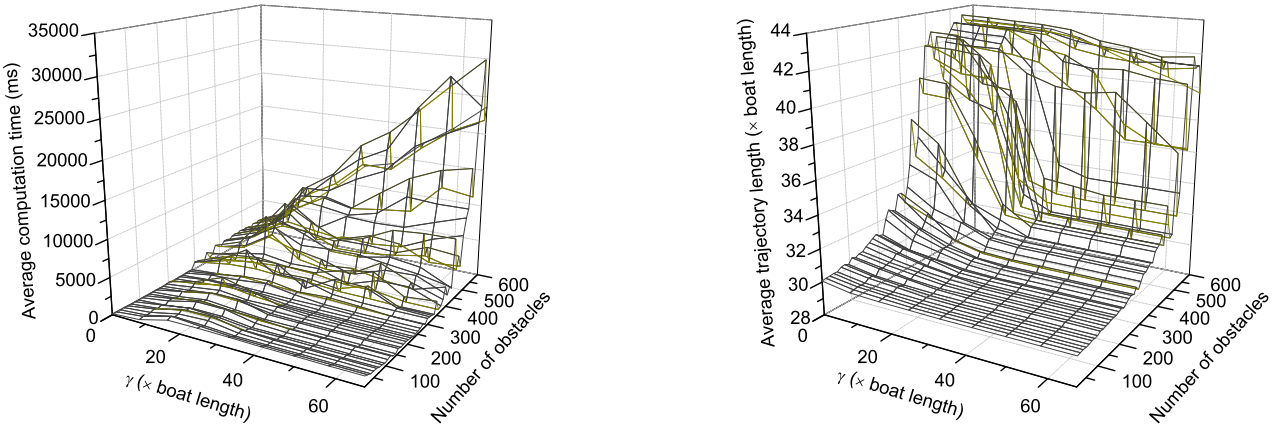
A few observations can be made as follows:

1. As the value of the parameter $\beta$ increases, the computation time also increases up to a certain value of $\beta$. The reason is that as $\beta$ increases, the length of the trajectory for which the search is performed in 3D space is increased in proportion to the trajectory length for which the search is carried out in 2D space. However, the computation time starts reducing beyond a particular value of $\beta$. This occurs because if the trajectory is searched in 3D up to a larger distance, much shorter trajectory may be found. A shorter trajectory entails a smaller number of nodes being expanded, which decreases the required computational time. For further higher values of $\beta$, the computation time saturates. This is because, beyond the optimum trajectory length determined in 3D space, increasing $\beta$ has no effect.
2. The saturation value of computation time occurring for large values of $\beta$ is greater than the computation time of the pure 2D search.
3. The trajectory length is large when $\beta$ is small as 2D search is not able to exploit smooth turns that are




(a) (b)

Fig. 8: Variation of the average trajectory computation time (see plot a)) and trajectory length (see plot b)) with respect to the trajectory length threshold $\beta$ for switching between 3D and 2D representations of the state space.




(a) (b)

Fig. 9: Variation of the average trajectory computation time (see plot a)) and trajectory length (see plot b)) with respect to the trajectory length threshold $\gamma$ for switching between a control action set consisting of 27 and 5 control actions.

possible using 3D lattice based search. This however is not true in general for all cases. For example, a lattice based structure will prevent transitions to adjoining states due to inbuilt vehicle constraints. However, we would like to report here that in general, the average case trajectory length can be smaller for a lattice search as compared to pure 2D search.

4. The value of $\beta$ should be chosen such that it does not lie in the peaked region. From the experimental test cases we can determine the prohibited regions of $\beta$ in which computation performance becomes worse for a given obstacle density. In other words, there is a certain range of $\beta$ for a given obstacle density for which the planner performance gets deteriorated, however, any value of $\beta$ less than that will result in performance improvement. This improvement in the performance, however, will have an adverse effect on the optimality of the generated trajectories. This can be observed from the plot of trajectory length variation versus the parameter $\beta$ (see Figure 8b)), which shows that for smaller $\beta$, the lengths of trajectories are greater. A balance can be chosen by the user of the algorithm to get performance gain by

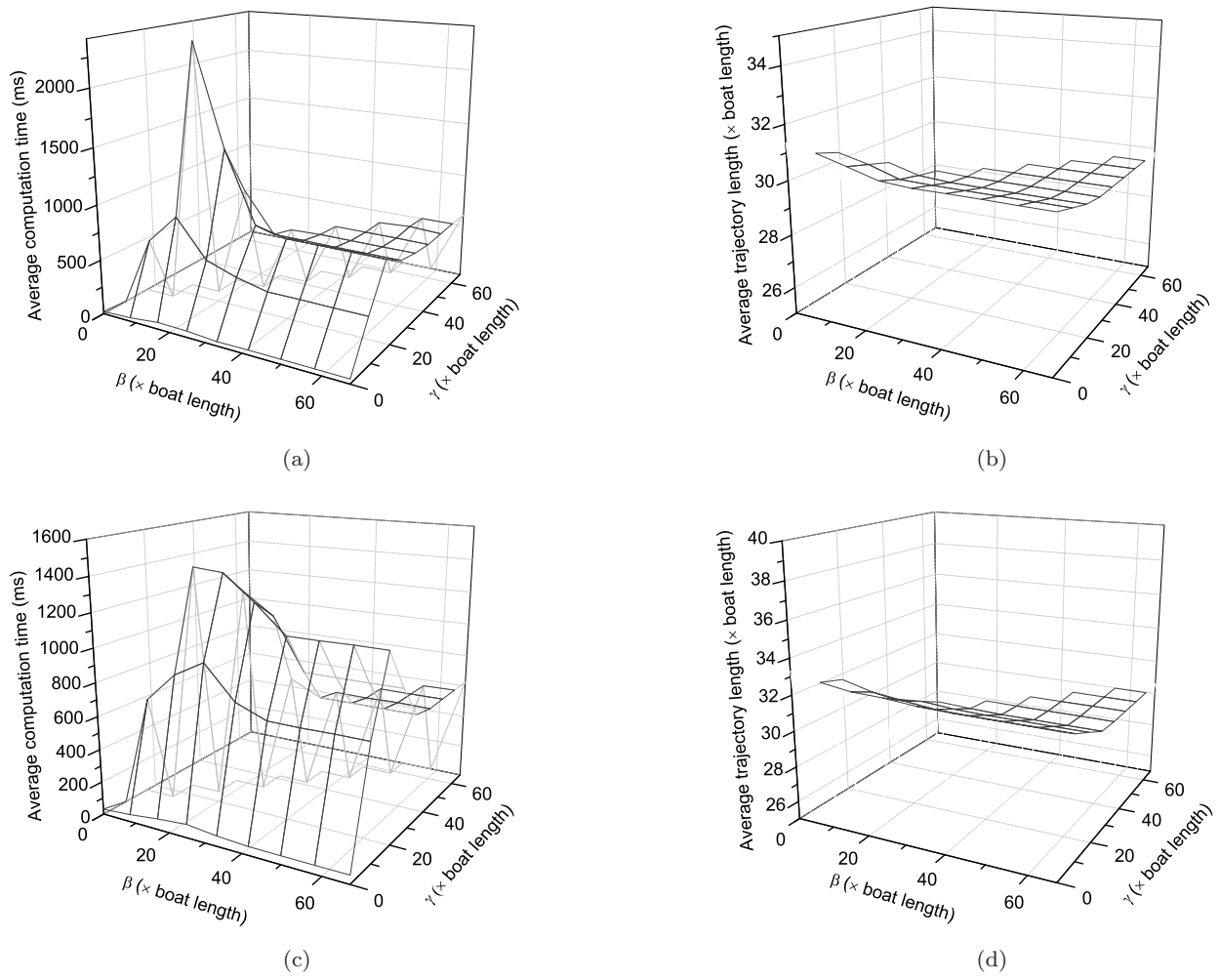



Fig. 10: Variation of the average computation time with threshold parameters $\gamma$ and $\beta$ for 120 (see plot a)) and 300 obstacles (see plot c)) in the scene. Variation of the average trajectory length with threshold parameters $\gamma$ and *beta* for 120 (see plot b)) and 300 obstacles (see plot d)) in the scene.

incurring allowable loss of optimality of the computed trajectory.

### 8.3 Evaluation of $\gamma$ Parameter

In order to test the influence of the parameter $\gamma$ on the computational time and trajectory length, we varied its value from 0 to 67 boat lengths in the increment of 8 as described above for the parameter $\beta$. This lead to $9 \times 62500 = 562500$ evaluations in total. For each $\gamma$ and obstacle scene we generated 1250 trajectories. The plot in Figure 9a) shows the variation of the computation time averaged over 1250 samples for each value of the $\gamma$ parameter and obstacle density. Similarly, the plot in Figure 9b) shows the variation of average trajectory length.

A few observations can be made as follows:

1. As the parameter $\gamma$ increases, the average computation time also increases. This is because the trajectory is searched in a densely connected lattice up to $\gamma$ distance of trajectory length, after which the sparsely connected lattice is used. However, beyond a certain limit of $\gamma$, the trajectory length gets smaller, causing lesser number of nodes being expanded and hence, the computation time reduces again. This effect is similar to the effect observed with the parameter $\beta$. The computation time saturates at values greater than optimal trajectory length searched in the densely connected lattice.

2. The saturation value of the computational time occurring for large values of $\gamma$ (searched purely in densely connected lattice) is higher than that of very small values of $\gamma$ (searched purely in sparsely connected lattice).
3. The average trajectory length reduces as $\gamma$ increases because trajectory searched in a dense lattice (i.e., due to a large value of $\gamma$) is more optimal compared to the one searched in a sparsely connected lattice.
4. The value of $\gamma$ should be chosen such that it does not lie in the peaked region. From the experimental test cases, we can determine the prohibited regions of $\gamma$ in which computation performance becomes worse for a given obstacle density. This is again similar to the behavior of $\beta$ parameter.

### 8.4 Evaluation of Combined Effect of $\beta$ and $\gamma$ Parameters

In order to study the combined influence of $\beta$ and $\gamma$ parameters, we have designed a test case, with $\beta$ and $\gamma$ varying from 0 to 67 of boat lengths. It should be noted that $\beta \geq \gamma$, as the influence of $\gamma$ (i.e., computing in densely connected lattice) takes priority over the influence of $\beta$. For each combination of the parameters $\gamma$ and $\beta$ we ran tests in two obstacle densities (namely with 120 and 300 obstacles). For each obstacle density we ran 25 randomly generated cases as described before. This resulted in running 62500 cases and computing a trajectory for each case. The variation of average trajectory lengths and computation time is illustrated in Figures 10a-d). The plots show the prohibited regions for each combination of $\beta$ and $\gamma$. In these regions, average computation time increases.

### 8.5 Simulation Results of Overall Planning Approach

To evaluate the motion goal prediction algorithm described in Section 6 we conducted a series of simulation experiments using randomly generated test cases. Each test case consisted of a fixed trajectory for the target boat and a set of 48, 72, 96, 120 or 144 randomly placed obstacles in an environment with the dimension of $33 \times 33$ boat lengths (i.e., $20 \times 20$ $m$). We generated 200 different trajectories for a total of 1000 different test cases. An example test case with the trajectories for both the USV and target boat is shown in Figure 11c.

The USV and target boat were given a maximum velocity of 0.7 boat lengths/$s$ (i.e., 0.4 $m/s$). Each second, the simulator invoked calls to compute a new motion goal for the USV, either using the heuristic motion goal predictor described in Section 6, or by directly using the current pose of the target boat, specified by $\mathbf{x_{T,d}}$. The purpose of the experiments was to demonstrate the effectiveness of the heuristic predictor of the motion goal when compared to the simple target boat tracking. The comparison was made in terms of the distance traveled by the USV and the amount of time spent inside a circular proximity region $X_P$ around the target, specified by minimum radius $r_{min}$ and maximum radius $r_{max}$. Figure 11b illustrates the reduction in trajectory length when using the heuristic motion goal predictor with a trajectory approximately 3.3 boat lengths shorter than the trajectory in Figure 11a generated by the USV using the simple tracking.

Throughout the experiments, the value of $r_{min}$ was fixed to 1.6 boat lengths (i.e., 1 $m$), while $r_{max}$ was varied between 3.3 (i.e., 2 $m$) and 10 boat lengths (i.e., 6 $m$). The lag time required by the heuristic motion goal predictor was set to $t_{lag} = (r_{min} + r_{max})/(2u_U)$, encouraging the USV to position itself half way between $r_{min}$ and $r_{max}$. The number of time points used by the heuristic, $k$, was set to 5 and the time horizon, $t_k$, was set to 10 $s$, and the discount factor, $\delta$, was fixed at 0.9. The trajectory planner used by the motion goal predictor was configured to terminate after 5000 iterations to reduce its running time. If no viable trajectory was found to any of the candidate motion goals, then the simple motion goal was used instead.

At the start of each test case, the USV was positioned within 8.3 boat lengths (i.e., 5 $m$) of the initial location of the target boat, and the USV was oriented in a direction facing the target. The target boat then followed its pre-defined trajectory for 2 minutes while the simulator recorded the amount of time that the USV was within the proximity region and the length of the trajectory followed by the USV.

Results for each of the test cases are presented in Figure 12, with a more detailed breakdown in Figure 13 comparing the heuristic motion goal prediction with simple tracking. As expected, increasing $r_{max}$ reduces the total traveled distance of the USV using the heuristic motion goal prediction, while it has no effect on the USV using the simple tracking. When $r_{max}$ is equal to 10, there is a nearly 10% reduction in trajectory length. This is achieved with no apparent decrease in the average amount of time spent in the proximity region, and in some cases (e.g., when $r_{max} = 3.3$) the amount of time spent in the proximity region noticeably increases when compared to the simple tracking. As the obstacle density increases, the average time spent in the proximity region decreases for both the motion goal computed using the developed heuristic and the motion goal determined as the current pose of the target boat, but the heuristic motion goal appears to retain its advantage. On the other hand, as the obstacle density increases, the average travelled distance by the USV increases or remains approximately to be the same for different values of $r_{max}$ (see Figure 12b)).

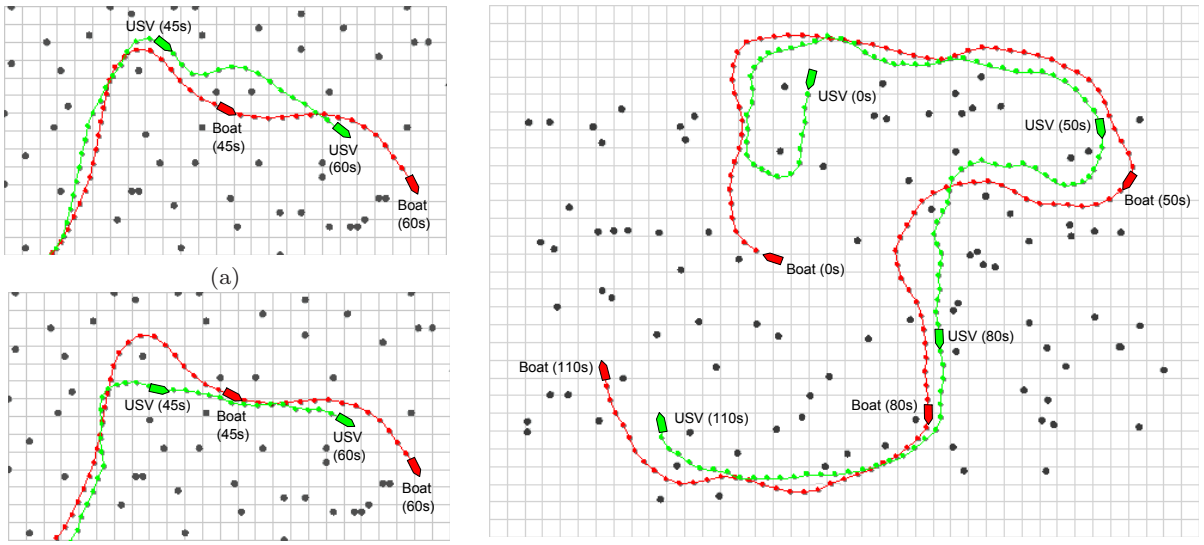



Fig. 11: (a) Trajectory of the USV using simple target boat tracking. (b) Trajectory of the USV using heuristic motion goal prediction. (c) Trajectory of the USV using heuristic motion goal prediction in a randomly generated scenario with 96 obstacles.

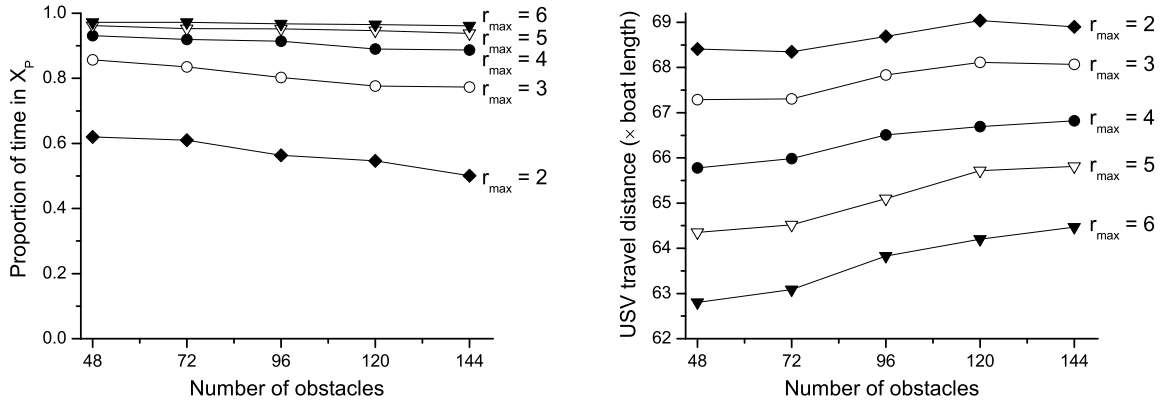
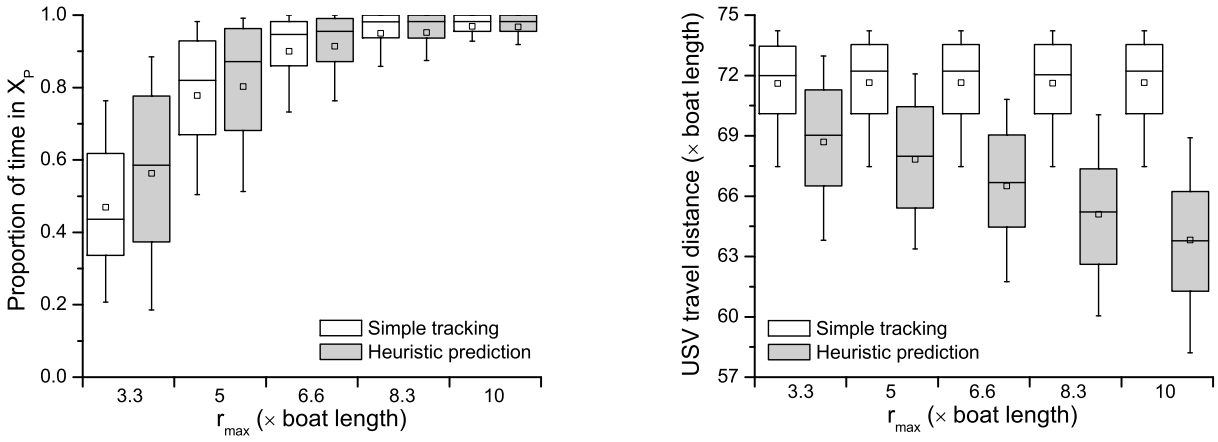



Fig. 12: USV performance while using heuristic motion goal prediction for various obstacle densities and $r_{max}$ values. (a) Proportion of time the USV spends in the proximity region. (b) Average distance traveled by the USV.

Although the simulation was not performed in real-time, the heuristic motion goal predictor had an average running time of 1.78 $s$, with a standard deviation of 1.07 $s$, just slightly slower than the allotted time of 1 $s$ to perform the computation. The computation in real-time would be achieved by utilizing the dimensionality reduction for the trajectory planner as described in Sections 7.1.1 and 7.1.2 and optimization of the planner software.

### 8.6 Experimental Setup

We have developed an experimental setup (see Figure 14) for physical evaluation of the planning approach for target boat following. The evaluation was conducted using two radio controlled boats in a 15 $m$ wide tank within the Neutral Buoyancy Research Facility (NBRF) at the University of Maryland. The boats are powered with a single




(a) (b)

Fig. 13: Comparison of simple and heuristic motion goal prediction for a set of 200 randomly generated test cases with 96 obstacles.(a) Portion of time the USV spends in the proximity region. (b) Average distance traveled by the USV. For each figure, the box plot shows the median, upper and lower quartile. The whiskers show the upper and lower decile. The mean value is represented by a small square.

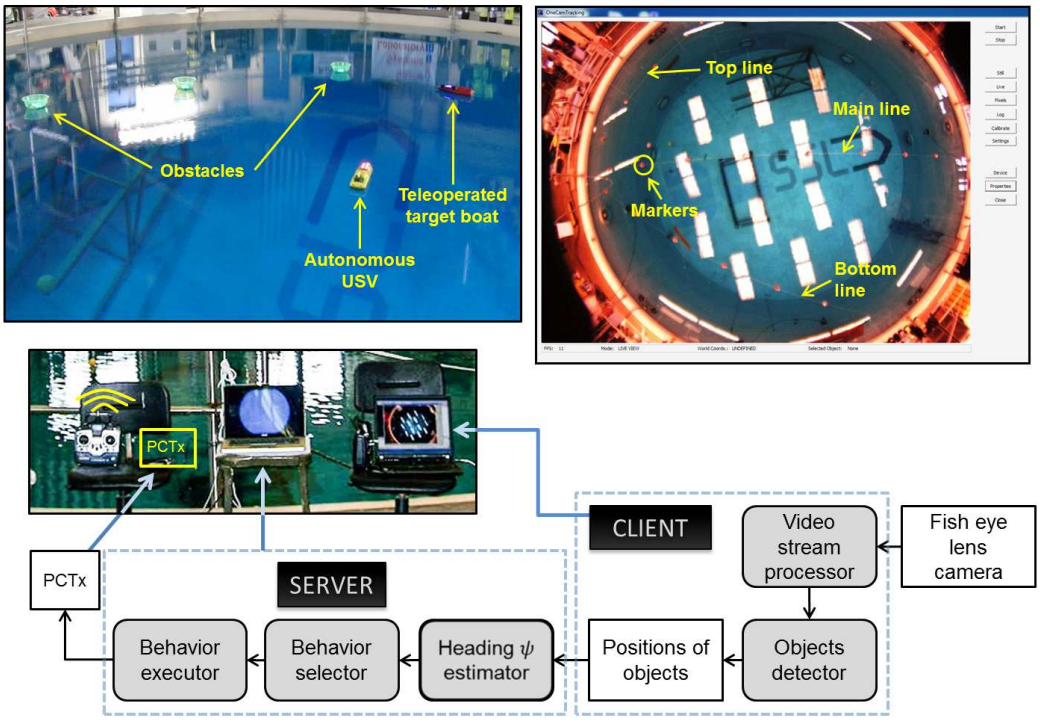



Fig. 14: Developed physical setup for testing autonomous behaviors in the Neutral Buoyancy Research Facility (NBRF) at the University of Maryland. The developed planner for following the target boat is represented as a follow behavior executed by the behavior selection and execution components of the entire system.

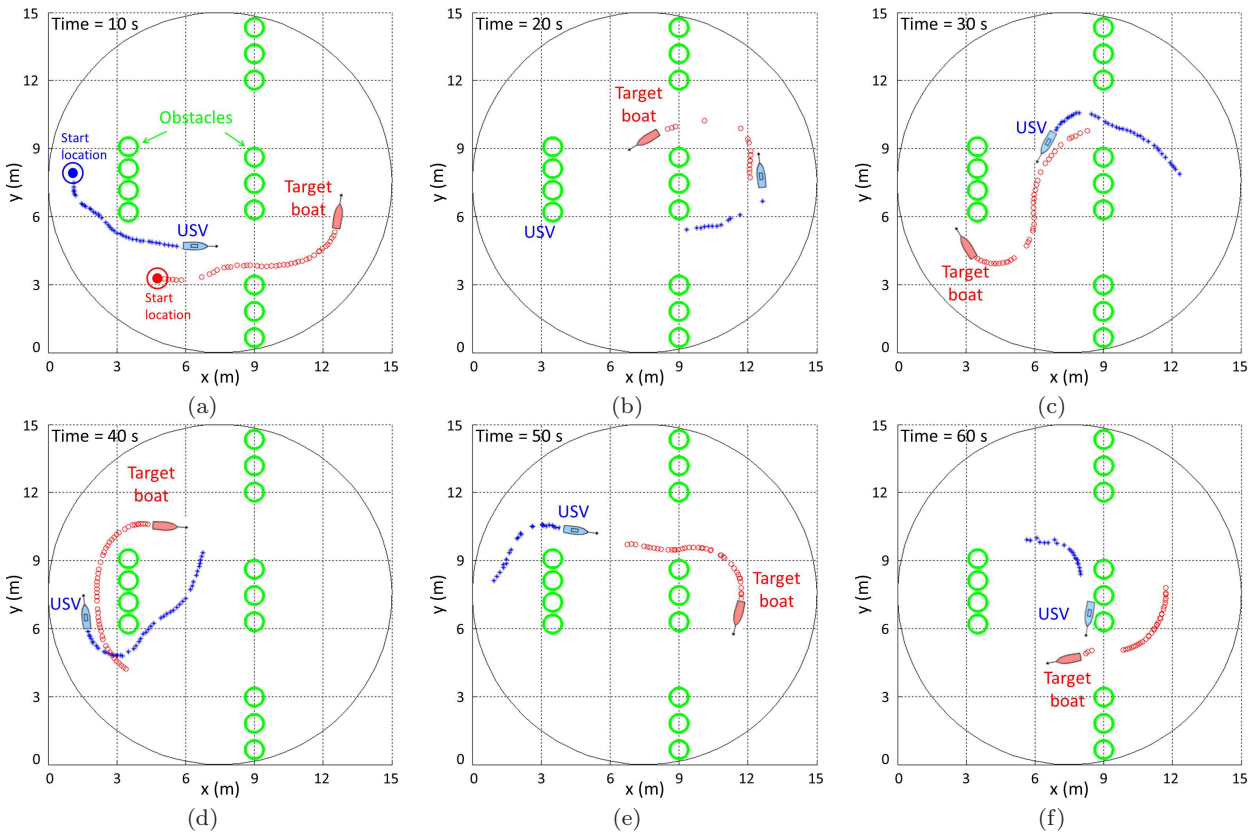



Fig. 15: Experimental result of the developed planning approach for following a target boat by an autonomous surface vehicle (USV) in an environment with static obstacles. The case f) illustrates a situation in which the USV takes a shortcut while following the target boat.

9.6 volt battery and consist of one DC motor to control the forward and reverse speed of the propeller and a servo motor to control the angular motion of rudder. We use Hitec transmitter and receiver pair which operates on 2.4 GHz bandwidth. We utilize Endurance PCTx interface between the laptop and the RC transmitter to allow radio transmission of actuator commands to control the throttle and rudder positions of the autonomous USV. The USV is remotely controlled by the developed nominal trajectory planner and trajectory tracking controller running on a laptop. The setup also allows us to control the target boat using a remote controller interface.

The positions of the boats are perceived using a vision system that includes a single fish eye lens camera mounted 8 m above the level of the pool. We have developed a color blob detection image processing algorithm to detect the boats. The USV as well as the target boat use SMD LEDs with high intensity to be easily recognizable by the blob detection algorithm.

We have also developed a camera calibration module to relate the measured entire pool area to a local coordinate frame of the USV planning system running on a laptop. The calibration of the camera provides us a model of its geometry as well as distortion model of the lens. The calibration system uses three reference lines with the same color markers separated by a fixed distance of 1.5 $m$ (see Figure 14). The input to the algorithm are the markers on each reference line starting from the origin of the coordinate frame and relative angles of the two of the lines (labeled as the top and bottom lines in Figure 14) with the main line. The algorithm then maps the entire physical pool to the planning workspace with the main line as the positive direction of the $x$ axis. The algorithm is based on the calibration procedure developed by Sturm and Maybank [94] and the Matlab camera calibration toolbox [95]. The algorithm represents a simplified version of the referred algorithms since the boats move approximately in the same plane.

The sensor uncertainty is handled through the use of the extended Kalman filter [26]. Due to the difficulty of reliable perception of all obstacles in the pool, we let the user to mark their positions using the interface of the planning

system. In this way, the user has the option to define a set of virtual obstacles that may reflect the physical obstacles in the scene.

For physical experiments, the trajectory planner utilizes the control action set for the USV as shown in Figure 3a). It consists of 5 control actions $u_A, u_B, u_C$ (and their symmetric counterparts $u'_A and u'_B$) with different final orientations $u_{A,\theta} = 0.785\ rad$, $u_{B,\theta} = 0.4636\ rad$,$u_{C,\theta} = 0\ rad$, and positions $u_{A,f} = [0.8, 0.8]^T\ m$,$u_{B,f} = [0.8, 0.4]^T\ m$, $u_{C,f} = [0.8, 0]^T\ m$. The control actions were determined based on the dynamic characteristics of the boat as well as its dimension ($0.6 \times 0.15 \times 0.13\ m$) and the obstacle density of the experimental environment. We specified the final orientation as well as position for each control action and employed a PID controller to validate the dynamical feasibility of the control action in the simulator. In addition, we set the maximum surge speed for the USV to be constant of 0.8 $ms^{-1}$ for straight segments of the trajectory and 0.3 $ms^{-1}$ for turns.

### 8.7 Experimental Results of Overall Planning Approach

In the designed physical experiment, an autonomous USV was supposed to follow a human-teleoperated target boat within the tank. The USV had a complete knowledge of the operating scene. The developed planner was utilized to find the shortest possible, dynamically feasible trajectory among obstacles to approach the target with a given orientation. Due to the limited dimension of the scene, we did not evaluate the motion goal prediction algorithm in these experiments. The developed trajectory tracking controller allowed the USV to follow the trajectory without overshooting and taking unnecessary corrective maneuvers.

Figure 15 shows a sequence of planning situations that arose during the execution of the follow task. In all these situations, the planner computes the shortest possible trajectory by taking the current positions of both the USV and the teleoperated boat, and estimating their heading. In most of the situations, the USV takes the same trajectory as the teleoperated boat. However, as can be seen in the situation illustrated in Fig. 15f), the USV chooses a shorter, different trajectory to approach the teleoperated boat. The experiment also shows that the developed system is able to compute and reliably execute the nominal trajectory.

## 9 Conclusions

This paper presents an approach for target boat following using an autonomous unmanned surface vehicle. The developed approach consists of the motion goal prediction, trajectory planning, and trajectory tracking algorithms. The motion goal prediction algorithm is used for estimation of the future pose of the target boat and computation of the desired pose for the USV to efficiently follow the target. A fast trajectory planner was developed for computation of a trajectory that complies with differential constraints of the USV. A trajectory tracking controller was developed for computation of desired velocities along the trajectory.

We have demonstrated the capabilities of the planner using both simulation and physical experiments. We have shown the efficiency of the motion goal prediction component in terms of the time spent in a proximity region around the target boat and travelled distance savings while following the target boat in environments with varying obstacle densities. The follow task imposes strict planning time constraints. Hence, it was necessary to satisfy these constraints by searching through a hybrid, pose-position state space using a multi-resolution control action set. We have carried out a detailed study of the developed techniques to speed up the search by finding a balance between computational time requirements and trajectory length. For the evaluation of the trajectory planner and tracker on a physical boat, we have developed a physical setup that encompasses vision, image processing, control, and software components.

One possible extension of the developed approach is to directly incorporate the motion uncertainty (in the form of a state transition model) into the computation of a physics-aware feedback plan [58] as we have already done in our previous work [83, 90]. Given the state transition model of the USV, we can model the planning problem as a Markov Decision Process (MDP) and solve it using value iteration to compute the feedback plan. The feedback plan indirectly defines the inevitable collision region of the planning space and can be used by the USV to determine a resolution-optimal action in every state. We also aim to explicitly handle the sensing uncertainty by enhancing the current version of the planner. We will perform thorough studies of the influence of both types of uncertainties on the performance of the planner using catamaran platforms in a realistic ocean environment. As another part of the future work, we will carry out theoretical studies of the effect of the dimension of the state-space and the resolution of the control action set on the computational efficiency and quality reduction of the computed trajectory. Finally, we will enhance the trajectory planner by locally optimizing control actions to increase the flexibility of the planner.

**Acknowledgements** This work was supported by the U.S. Office of Naval Research under Grants N00014-10-1-0585 and N00014-12-1-0494, managed by R. Brizzolara. In addition, we would like to thank Dr. David Akin for allowing us to perform experiments

in the Neutral Buoyancy Research Facility (NBRF) at the University of Maryland.

## References


1. S.J. Corfield and J.M. Young. Unmanned surface vehicles–game changing technology for naval operations. *Advances in unmanned marine vehicles*, pages 311–328, 2006.
2. V. Bertram. Unmanned surface vehicles–a survey. *Skibsteknisk Selskab, Copenhagen, Denmark*, 2008.
3. J.E. Manley. Unmanned surface vehicles, 15 years of development. In *OCEANS 2008*, pages 1–4, 2008.
4. M.M. Graham. Unmanned surface vehicles: An operational commander's tool for maritime security. Technical report, DTIC Document, 2008.
5. A. Aguiar, J. Almeida, M. Bayat, B. Cardeira, R. Cunha, A. Häusler, P. Maurya, A. Oliveira, A. Pascoal, A. Pereira, et al. Cooperative control of multiple marine vehicles. In *Proc. of 8th IFAC International Conference on Manoeuvering and Control of Marine Craft*, pages 16–18.
6. D.P. Eickstedt, M.R. Benjamin, and J. Curcio. Behavior based adaptive control for autonomous oceanographic sampling. In *IEEE International Conference on Robotics and Automation*, pages 4245–4250, 2007.
7. JP Ryan, SB Johnson, A. Sherman, K. Rajan, F. Py, H. Thomas, JBJ Harvey, L. Bird, JD Paduan, and RC Vrijenhoek. Mobile autonomous process sampling within coastal ocean observing systems. *Limnol. Oceanogr.: Methods*, 8:394–402, 2010.
8. W. Naeem, R. Sutton, and J. Chudley. Modelling and control of an unmanned surface vehicle for environmental monitoring. In *UKACC International Control Conference, August, Glasgow, Scotland* , 2006.
9. E.T. Steimle and M.L. Hall. Unmanned surface vehicles as environmental monitoring and assessment tools. In *OCEANS 2006*, pages 1–5. IEEE, 2006.
10. W. Naeem, T. Xu, R. Sutton, and A. Tiano. The design of a navigation, guidance, and control system for an unmanned surface vehicle for environmental monitoring. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 222(2):67, 2008.
11. R. Yan, S. Pang, H. Sun, and Y. Pang. Development and missions of unmanned surface vehicle. *Journal of Marine Science and Application*, 9(4):451–457, 2010.
12. J. Majohr and T. Buch. Modelling, simulation and control of an autonomous surface marine vehicle for surveying applications measuring dolphin messin. *IEE Control Engineering Series*, 69:329, 2006.
13. T.C. Furfaro, J.E. Dusek, and K.D. von Ellenrieder. Design, construction, and initial testing of an autonomous surface vehicle for riverine and coastal reconnaissance. In *OCEANS 2009, MTS/IEEE Biloxi-Marine Technology for Our Future: Global and Local Challenges*, pages 1–6, 2009.
14. R.R. Murphy, E. Steimle, C. Griffin, C. Cullins, M. Hall, and K. Pratt. Cooperative use of unmanned sea surface and micro aerial vehicles at hurricane wilma. *Journal of Field Robotics*, 25(3):164–180, 2008.
15. A.J. Shafer, M.R. Benjamin, J.J. Leonard, and J. Curcio. Autonomous cooperation of heterogeneous platforms for sea-based search tasks. In *Proceedings of MTS/IEEE OCEANS*, pages 1–10. IEEE, 2008.
16. R.M. Kilgore, K.A. Harper, C. Nehme, and ML Cummings. Mission planning and monitoring for heterogeneous unmanned vehicle teams: A human-centered perspective. In *AIAA Infotech@ Aerospace Conference in Sonoma, CA*, 2007.
17. E. Simetti, A. Turetta, G. Casalino, E. Storti, and M. Cresta. Towards the use of a team of USVs for civilian harbour protection: Real time path planning with avoidance of multiple moving obstacles. In *IEEE 3rd Workshop on Planning, Perception and Navigation for Intelligent Vehicles (IROS'09), St. Louis, MO, US*, 2009.
18. Enrico Simetti, Alessio Turetta, Giuseppe Casalino, Enrico Storti, and Matteo Cresta. Protecting assets within a civilian harbour through the use of a team of usvs: Interception of possible menaces. In *IARP Workshop on Robots for Risky Interventions and Environmental Surveillance-Maintenance (RISE'10), Sheffield, UK*, 2010.
19. P. Švec and S.K. Gupta. Automated synthesis of action selection policies for unmanned vehicles operating in adverse environments. *Autonomous Robots*, 32(2):149–164, 2012.
20. E. Raboin, P. Švec, D. S. Nau, and S. K. Gupta. Model-predictive target defense by team of unmanned surface vehicles operating in uncertain environments. In *IEEE International Conference on Robotics and Automation (ICRA'13)*, Karlsruhe, Germany, May 6-10 2013.
21. Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.
22. M. Pivtoraiko, R. A. Knepper, and A. Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.
23. A. Thakur and S.K. Gupta. Real-time dynamics simulation of unmanned sea surface vehicle for virtual environments. *Journal of Computing and Information Science in Engineering*, 11(3):031005, 2011.
24. T.I. Fossen. *Handbook of marine craft hydrodynamics and motion control*. Wiley, 2011.
25. Thierry Fraichard and Hajime Asama. Inevitable collision states a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.
26. S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT Press, 2005.
27. P. Švec, A. Thakur, B. C. Shah, and S. K. Gupta. USV trajectory planning for time varying motion goal in an environment with obstacles. In *ASME Mechanisms and Robotics Conference*, Chicago, USA, August 12-15 2012.
28. T.H. Chung, G.A. Hollinger, and V. Isler. Search and pursuit-evasion in mobile robotics. *Autonomous Robots*, pages 1–18, 2011.
29. M. Bibuli, M. Caccia, L. Lapierre, and G. Bruzzone. Guidance of unmanned surface vehicles: Experiments in vehicle following. *Robotics & Automation Magazine, IEEE*, (99):1–1.
30. M. Breivik, V.E. Hovstein, and T.I. Fossen. Straight-line target tracking for unmanned surface vehicles. *Modeling, Identification and Control*, 29(4):131–149, 2008.
31. M. Breivik. *Topics in guided motion control of marine vehicles*. PhD thesis, Norwegian University of Science and Technology, 2010.
32. Z. Peng, D. Wang, Z. Chen, X. Hu, and W. Lan. Adaptive dynamic surface control for formations of autonomous surface vehicles with uncertain dynamics. *IEEE Transactions on Control Systems Technology*, (99):1–8.
33. M. Breivik, V.E. Hovstein, and T.I. Fossen. Ship formation control: A guided leader-follower approach. In *World Congress*, volume 17, pages 16008–16014, 2008.

34. M. R. Katebi, M. J. Grimble, and Y. Zhang. H ∞ robust control design for dynamic ship positioning. *Control Theory and Applications, IEEE Proceedings*, 144(2):110–120, Mar 1997.
35. A. Loria, T. I. Fossen, and E. Panteley. A separation principle for dynamic positioning of ships: Theoretical and experimental results. *IEEE Transactions on Control Systems Technology*, 8:332–343, 2000.
36. F. Mazenc, K. Pettersen, and H. Nijmeijer. Global uniform asymptotic stabilization of an underactuated surface vessel. *IEEE Transactions on Automatic Control*, 47(10):1759–1762, Oct 2002.
37. K. D. Do, Z. P. Jiang, and J. Pan. Underactuated ship global tracking under relaxed conditions. *IEEE Transactions on Automatic Control*, 47(9):1529–1536, Sep 2002.
38. E. Lefeber, K.Y. Pettersen, and H. Nijmeijer. Tracking control of an underactuated ship. *IEEE Transactions on Control Systems Technology*, 11(1):52–61, Jan 2003.
39. H. Ashrafiuon, K. R. Muske, and L. C. McNinch. Review of nonlinear tracking and setpoint control approaches for autonomous underactuated marine vehicles. In *American Control Conference (ACC'10)*, pages 5203 –5211, July 2010.
40. M. Caccia, M. Bibuli, R. Bono, and G. Bruzzone. Basic navigation, guidance and control of an unmanned surface vehicle. *Autonomous Robots*, 25(4):349–365, 2008.
41. J. Curcio, J. Leonard, and A. Patrikalakis. Scout-a low cost autonomous surface platform for research in cooperative autonomy. In *Proceedings of MTS/IEEE OCEANS*, pages 725–729, 2005.
42. A. Pascoal, C. Silvestre, and P. Oliveira. Vehicle and mission control of single and multiple autonomous marine robots. *IEE Control Engineering Series*, 69:353, 2006.
43. M. Bibuli, G. Bruzzone, M. Caccia, and L. Lapierre. Path-following algorithms and experiments for an unmanned surface vehicle. *Journal of Field Robotics*, 26(8):669–688, 2009.
44. J. Alves, P. Oliveira, R. Oliveira, A. Pascoal, M. Rufino, L. Sebastiao, and C. Silvestre. Vehicle and mission control of the delfim autonomous surface craft. In *14th Mediterranean Conference on Control and Automation (MED'06)*, pages 1–6, 2006.
45. M. R. Benjamin, J. A. Curcio, and P. M. Newman. Navigation of unmanned marine vehicles in accordance with the rulesof the road. In *IEEE International Conference on Robotics and Automation (ICRA'06)*, pages 3581 –3587, May 2006.
46. J. Larson, M. Bruch, and J. Ebken. Autonomous navigation and obstacle avoidance for unmanned surface vehicles. Technical report, DTIC Document, 2006.
47. W. Naeem, G.W. Irwin, and A. Yang. Colregs-based collision avoidance strategies for unmanned surface vehicles. *Mechatronics*, 2011.
48. Y. Kuwata, M.T. Wolf, D. Zarzhitsky, and T.L. Huntsberger. Safe maritime navigation with colregs using velocity obstacles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11)*, pages 4728–4734. IEEE, 2011.
49. Petr Švec, Brual C Shah, Ivan R. Bertaska, Jose Alvarez, Armando J. Sinisterra, Karl von Ellenrieder, Manhar Dhanak, and Satyandra K. Gupta. Dynamics-aware target following for an autonomous surface vehicle operating under COLREGs in civilian traffic. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13)*, Tokyo, Japan, 2013.
50. J. Ebken. Applying unmanned ground vehicle technologies to unmanned surface vehicles. Technical report, DTIC Document, 2005.
51. A. Larson, J., M. Bruch, and J. Ebken. Autonomous navigation and obstacle avoidance for unmanned surface vehicles. In *SPIE Proc. 6230: Unmanned Systems Technology VIII*, pages 17–20, 2006.
52. P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.
53. Giuseppe Casalino, Alessio Turetta, and Enrico Simetti. A three-layered architecture for real time path planning and obstacle avoidance for surveillance usvs operating in harbour fields. In *OCEANS'09*, pages 1–8. IEEE, 2009.
54. T. Huntsberger, H. Aghazarian, D. Gaines, M. Garrett, and G. Sibley. Autonomous operation of unmanned surface vehicles (usv's). In *Proc. IEEE ICRA Workshop on Robots in Challenging and Hazardous Environments*, 2007.
55. T. Huntsberger, H. Aghazarian, A. Castano, G. Woodward, C. Padgett, D. Gaines, and C. Buzzell. Intelligent autonomy for unmanned sea surface and underwater vehicles. *In AUVSI Unmanned Systems North America*, 2008.
56. M. Greytak and F. Hover. Motion planning with an analytic risk cost for holonomic vehicles. In *Proceedings of the 48th IEEE Conference on Decision and Control held jointly with the 2009 28th Chinese Control Conference (CDC/CCC'09)*, pages 5655–5660. IEEE, 2009.
57. I.R. Bertaska, J. Alvarez, S. Armando, K. D. von Ellenrieder, M. Dhanak, B. Shah, P. Švec, and S. K. Gupta. Experimental evaluation of approach behavior for autonomous surface vehicles. In *ASME Dynamic Systems and Control Conference (DSCC'13)*, Stanford University, Palo Alto, CA, October 21-23 2013.
58. S. M. LaValle. *Planning algorithms.* Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu.
59. C. Goerzen, Z. Kong, and B. Mettler. A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent Robotic Systems*, 57:65–100, 2010. 10.1007/s10846-009-9383-1.
60. B. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *J. ACM*, 40:1048–1066, November 1993.
61. E. Frazzoli, M.A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. In *Proceedings of the 2001 American Control Conference*, volume 1, pages 43 –49, 2001.
62. S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.
63. D. Hsu, R. Kindel, J. C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233–255, 2002.
64. S. M. Lavalle and P. Konkimalla. Algorithms for computing numerical optimal feedback motion strategies. *The International Journal of Robotics Research*, 20(9):729–752, 2001.
65. S. Suzuki. Online four-dimensional flight trajectory search and its flight testing. *AIAA GNC Conference and Exhibit*, 2005.
66. M. Bennewitz, W. Burgard, and S. Thrun. Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. *Robotics and autonomous systems*, 41(2-3):89–99, NOV 30 2002.
67. S. Scherer, S. Singh, L. Chamberlain, and S. Saripalli. Flying fast and low among obstacles. In *IEEE International Conference on Robotics and Automation (ICRA'07)*, pages 2023 –2029, april 2007.

68. E. Frazzoli, M.A. Dahleh, and E. Feron. A hybrid control architecture for aggressive maneuvering of autonomous helicopters. In *Proceedings of the 38th IEEE Conference on Decision and Control*, volume 3, pages 2471 –2476 vol.3, 1999.
69. A. Kelly, A. Stentz, O. Amidi, M. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, P. Rander, S. Thayer, N. Vallidis, and Ra. Warner. Toward reliable off road autonomous vehicles operating in challenging environments. *The International Journal of Robotics Research*, 25(5-6):449–483, 2006.
70. Y. Li and J. Xiao. On-line planning of nonholonomic trajectories in crowded and geometrically unknown environments. In *IEEE International Conference on Robotics and Automation (ICRA'09)*, pages 3230–3236. IEEE, 2009.
71. A. Elnagar and A. Hussein. On optimal constrained trajectory planning in 3d environments. *Robotics and Autonomous Systems*, 33(4):195 – 206, 2000.
72. A. Richards and J.P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the 2002 American Control Conference (ACC'02)*, volume 3, pages 1936 – 1941 vol.3, 2002.
73. F. Borrelli, D. Subramanian, A.U. Raghunathan, and L.T. Biegler. MILP and NLP techniques for centralized trajectory planning of multiple unmanned air vehicles. In *American Control Conference*, page 6 pp., june 2006.
74. M. G. Earl and R. D'Andrea. Iterative MILP methods for vehicle-control problems. *IEEE Transactions on Robotics*, 21(6):1158 – 1167, dec. 2005.
75. C. Colombo, M. Vasile, and G. Radice. Optimal low-thrust trajectories to asteroids through an algorithm based on differential dynamic programming. *Celestial Mechanics and Dynamical Astronomy*, 105:75–112, 2009. 10.1007/s10569-009-9224-3.
76. T. Howard and A. Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. *International Journal of Robotics Research*, 26(1):141–166, February 2007.
77. O. Purwin and R. Andrea. Trajectory generation and control for four wheeled omnidirectional vehicles. *Robotics and Autonomous Systems*, 54(1):13 – 22, 2006.
78. Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.
79. R. A. Soltan, H. Ashrafiuon, and K. R. Muske. State-dependent trajectory planning and tracking control of unmanned surface vessels. In *American Control Conference (ACC'09)*, pages 3597 –3602, 2009.
80. B. Xu, A. Kurdila, and D. J. Stilwell. A hybrid receding horizon control method for path planning in uncertain environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'09)*, pages 4887 –4892, 2009.
81. M. Sandler, A. Wahl, R. Zimmermann, M. Faul, U. Kabatek, and E. D. Gilles. Autonomous guidance of ships on waterways. *Robotics and Autonomous Systems*, 18(3):327 – 335, 1996.
82. T. I. Fossen. *Guidance and control of ocean vehicles.* Wiley, Chicester, England, 1994.
83. Atul Thakur, Petr Švec, and Satyandra K. Gupta. {GPU} based generation of state transition models using simulations for unmanned surface vehicle trajectory planning. *Robotics and Autonomous Systems*, 60(12):1457 – 1471, 2012.
84. I. Zohar, A. Ailon, and R. Rabinovici. Mobile robot characterized by dynamic and kinematic equations and actuator dynamics: Trajectory tracking and related application. *Robotics and autonomous systems*, 59(6):343–353, June 2011.
85. M. Pivtoraiko and A. Kelly. Generating near minimal spanning control sets for constrained motion planning in discrete state spaces. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*, pages 3231–3237. IEEE, 2005.
86. Colin Green and Alonzo Kelly. Toward optimal sampling in the space of paths. In *13th International Symposium of Robotics Research*, November 2007.
87. Mihail Pivtoraiko, Issa AD Nesnas, and Alonzo Kelly. Autonomous robot navigation using advanced motion primitives. In *IEEE Aerospace conference*, pages 1–7. IEEE, 2009.
88. Lawrence H Erickson and Steven M LaValle. Survivability: Measuring and ensuring path diversity. In *IEEE International Conference on Robotics and Automation (ICRA'09)*, pages 2068–2073. IEEE, 2009.
89. Mikhail Pivtoraiko. *Differentially Constrained Motion Planning with State Lattice Motion Primitives.* PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, February 2012.
90. P. Švec, M. Schwartz, A. Thakur, and S. K. Gupta. Trajectory planning with look-ahead for unmanned sea surface vehicles to handle environmental disturbances. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11)*, San Francisco, CA, USA, September 25-30 2011.
91. S. Russell and P. Norvig. *Artificial intelligence: A modern approach.* Prentice Hall, 2009.
92. P. Krishnamurthy, F. Khorrami, and S. Fujikawa. A modeling framework for six degree-of-freedom control of unmanned sea surface vehicles. In *Proc. and 2005 European Control Conference Decision and Control CDC-ECC '05. 44th IEEE Conference on*, pages 2676–2681, December 12–15, 2005.
93. A.J. Healey and D. Lienard. Multivariable sliding mode control for autonomous diving and steering of unmanned underwater vehicles. *IEEE Journal of Oceanic Engineering*, 18(3):327 –339, July 1993.
94. Peter F Sturm and Stephen J Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1. IEEE, 1999.
95. Jean-Yves Bouguet. *Camera Calibration Toolbox for Matlab.* 2010. Available at http://www.vision.caltech.edu/bouguetj/calib_doc.