Received September 15, 2020, accepted October 23, 2020, date of publication November 2, 2020, date of current version November 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3035256

Two-Stage Optimized Trajectory Planning for ASVs Under Polygonal Obstacle Constraints: Theory and Experiments

GLENN BITAR, ANDREAS B. MARTINSEN, ANASTASIOS M. LEKKAS, (Member, IEEE), AND MORTEN BREIVIK, (Member, IEEE)
Centre for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), 7491 Trondheim, Norway

Corresponding author: Glenn Bitar (glennbitar@outlook.com)

This work was supported in part by the Research Council of Norway under Project 269116, and in part by the Centres of Excellence Funding Scheme under Project 223254.

ABSTRACT We propose a method for energy-optimized trajectory planning for autonomous surface vehicles (ASVs), which can handle arbitrary polygonal maps as obstacle constraints. The method comprises two stages: The first is a hybrid A* search that finds a dynamically feasible trajectory in a polygonal map on a discretized configuration space using optimal motion primitives. The second stage uses the resulting hybrid A* trajectory as an initial guess to an optimal control problem (OCP) solver. In addition to providing the OCP with a warm start, we use the initial guess to create convex regions encoded as halfspace descriptions, which converts the inherent nonconvex obstacle constraints into a convex and smooth representation. The OCP uses this representation in order to optimize the initial guess within a collision-free corridor. The OCP solves the trajectory planning problem in continuous state space. Our approach solves two challenges related to optimization-based trajectory planning: The need for a dynamically feasible initial guess that can guide the solver away from undesirable local optima and the ability to represent arbitrary obstacle shapes as smooth constraints. The method can take into account external disturbances such as wind or ocean currents. We compare our method to two similar trajectory planning methods in simulation and have found significant computation time improvements. Additionally, we have validated the method in full-scale experiments in the Trondheim harbor area.

INDEX TERMS Autonomous vehicles, collision avoidance, marine vehicles, motion planning, polygonal collision-avoidance constraints, trajectory optimization, trajectory planning.

I. INTRODUCTION

In marine applications, we see efforts to increase the level of autonomy in research, defense, and commercial applications. Motivated by benefits to costs, safety, and environmental impact, many actors consider using autonomous vessels in their operations. In 2018, both Wärtsilä and Rolls-Royce Marine (acquired by Kongsberg Maritime) demonstrated autonomous capabilities with the ferries *Folgefonn* and *Falco*, respectively.[1] Both tests included automatic transit and docking. Another example of commercial use of maritime autonomous technology is when the Japanese shipping company NYK completed the world's first maritime autonomous surface ship trial in 2019.[2]

An essential part of an autonomous marine system is path and trajectory planning, where the goal is to plan how the vessel will move from its start location to the goal location. Path planning finds a sequence of collision-free configurations without temporal constraints, while trajectory planning adds temporal constraints, often via a time-parametrized state trajectory. Our interests lie within energy-optimized operations, and since energy consumption is highly sensitive to velocity, we focus on trajectory planning.

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

[1]https://www.maritime-executive.com/article/rolls-royce-and-wartsila-in-close-race-with-autonomous-ferries (accessed September 14, 2020).

[2]https://www.nyk.com/english/news/2019/20190930_01.html (accessed August 31, 2020).

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

A. BACKGROUND AND RELEVANT WORK

Maritime agencies and research institutions actively research autonomous technology for, e.g., underwater operations for ocean mapping and monitoring [1], and autonomous transportation, focusing on the international regulations for preventing collisions at sea (COLREGs) [2]. Seto [3] gives an overview of autonomous technologies for maritime systems, and Pendleton *et al.* [4] give an overview of autonomy in vehicles in general. Pendleton Path and trajectory planning is a crucial technology for enabling autonomy at sea.

In robotics, there are numerous methods developed for path and trajectory planning. A general introduction to path planning is written by LaValle [5], who looks at the topic from the perspective of computer science while introducing widespread notation and nomenclature. Wolek and Woolsey [6] give an overview of model-based approaches to path planning for ground, surface, underwater, and air vehicles. We can coarsely divide planning methods into *roadmap methods* that explore points in the configuration space that, when connected, build a path between start and goal, and *optimization-based methods* that produce connected paths or trajectories using analytical or approximate optimization. Some advantages of roadmap methods include quickly finding the global solution of a path planning problem, and they allow for flexible obstacle representations, e.g., polygonal constraints. On the other hand, roadmap methods are discrete and are not generally able to find an optimal path or trajectory in a continuous domain. Optimization-based methods are often slower and subject to finding local optima. However, they naturally search in the continuous domain. Additionally, gradient-based methods for solving optimization problems require continuously differentiable representations of constraints, restricting how we can represent obstacles.

A simple example of roadmap methods is the A* search algorithm [7]. A* is a graph search algorithm commonly used as a path planner by discretizing a continuous map, often into a uniform, rectangular grid. A* quickly provides a piecewise linear path from start to goal. A more involved roadmap method comes from Candeloro *et al.* [8], where the authors discretize a map using a Voronoi diagram, subsequently refining and smoothing the result to give a curvature-continuous path. These methods are fast, but lack dynamic feasibility[3], and can only be optimal in terms of the employed map discretization. Roadmap methods also include sampling-based methods. These methods explore random points to build a roadmap between start and goal. Examples include the probabilistic roadmap [9], as well as rapidly-exploring random trees [10] and variations of those. Sampling-based methods are shown to be useful for planning in high-dimensional configuration spaces, where combinatorial roadmap methods often run into the so-called *curse of dimensionality* [11].

Model-based optimization-based methods are researched in automotive, aerial, and marine applications to create dynamically feasible paths or trajectories. Optimization-based methods are sometimes used to refine the result of a roadmap search or used as the primary tool to plan a trajectory. In [12]–[14], the authors present optimization-based trajectory planning methods that use smooth representations of rectangles and ellipses to approximate the obstacle map. This type of representation makes the optimization problem feasible to solve using gradient-based methods. However, there is an impractical tradeoff between the representation accuracy and number of constraints in the optimization problem. Additionally, these shapes may not be generic enough to represent detailed obstacle maps. By reformulating the obstacle avoidance constraint and introducing auxiliary optimization variables, Zhang *et al.* [15] have developed an alternative method for representing obstacles. This method allows the encoding of arbitrary convex polygons as smooth optimization constraints by introducing auxiliary optimization variables. The method works well for a low number of obstacles, but the optimization problem grows significantly with the number of obstacles and the number of polygon edges, to the point where it is not feasible to use it for marine applications with detailed obstacle maps. Bergman *et al.* [16] propose to bypass the inherent non-convexity of static obstacle avoidance by calculating a series of convex polytopes where their vehicle is allowed to move. The method gives smooth, convex obstacle avoidance constraints for their optimization-based planner, but lacks consideration of environmental disturbances. An optimization-based trajectory planning method for autonomous driving developed by Chen *et al.* [17] can represent polygonal obstacle constraints. Their method is based on linear quadratic control with an iterative optimization solver. A prerequisite for their method is an initial dynamically feasible trajectory in order to perform the optimization. However, their method does not provide a way of determining such a trajectory. This issue is common with optimization-based methods, and without an initial guess, they are prone to locking into solutions that represent undesirable local optima, i.e., solutions that may be far away from the globally optimal solution, as demonstrated in, e.g., [14]. In that example, the optimization-based planner finds a poor solution in the absence of a helpful initial guess. Depending on the objective function, finding a good initial guess to warm-start an optimization-based planner can be straightforward. In the case of finding a minimum-distance path, simple roadmap-based methods may quickly find paths in the discrete domain that lie close to the optimal solution in the continuous domain. Optimization-based methods can use this type of path as an initial guess. For energy-based objective functions, for instance, or when introducing dynamic constraints, creating feasible trajectories to use as initial guesses is more challenging, and suggests alternative approaches.

[3]We use the term "dynamically feasible" to indicate that a trajectory satisfies dynamic constraints in the form of model-based differential equations. A path that consists of a smoothed roadmap is usually feasible in terms of specified a turning radius. This turning radius is dependent on vessel speed, and the path is thus not *dynamically* feasible since it is not based on a model that includes speed.

Zhang *et al.* [15] propose using the hybrid A⋆ algorithm [18] to find such a trajectory for an optimization-based solver. Their application is autonomous parking of a car, described with a dynamical model, and using a cost function that blends minimum time and control effort. A simplified dynamical model and cost function is used in the hybrid A⋆ search stage, and the search solution is used as an initial guess for the optimization-based planner. The method does not take into account external disturbances. Bergman *et al.* [16] have developed a receding-horizon optimization-based planner warm-started by using a graph search method. The graph search method works on a lattice of a marine vessel's discretized state space with optimal state transitions. To facilitate motion in confined harbor areas, the authors use a cost function that blends distance to obstacles, minimum time, control effort, and control smoothness. Zhang *et al.* [19] and Meng *et al.* [20] propose optimization-based trajectory planning methods for autonomous driving that utilize roadmap methods to generate nominal trajectories for geometrical paths and subsequently use optimization to improve them. In both papers, speed profiles are handled separately from the geometrical path planning.

B. CONTRIBUTIONS

We have developed a method that plans energy-optimized trajectories in an environment defined by polygonal obstacles for an autonomous surface vehicle (ASV) under the influence of external disturbances. Our method is based on continuous optimal control, and the optimal control problem (OCP) solver is warm-started by the solution of a hybrid A⋆ search algorithm. The method's proposed use case is to plan an ASV voyage's transit stage before the voyage starts. The method handles only static obstacles, and is thus suitable for use as the top layer in a hybrid collision avoidance scheme, as proposed in [2], [21], [22]. Figure 1 shows a high-level block diagram of the trajectory planning method. The main differences between our method and the planner described in [16] are that we use a hybrid A⋆ search to calculate the initial guess, which allows us to account for estimated external disturbances, such as wind. Additionally, we use an alternative method to calculate the convex envelopes in preparation for the trajectory optimization stage. Like the method in [15], we also use hybrid A⋆ to generate an initial guess before optimizing. However, we propose an alternative obstacle representation, which scales more efficiently with the number of polygons and polygon edges in the obstacle map, in terms of the number of optimization variables. Our method shares similarities with [23] as well, where the workspace is decomposed into triangular cells to account for static, polygonal obstacles, and an optimization-based search finds sub-trajectories in each of the triangular cells. However, that method does not include an initial guess to warm-start the OCP solver.

Our contributions are as follows:

- We have extended the hybrid A⋆ search developed by Dolgov *et al.* [18] to the ASV application by using an energy-based cost function that depends on the velocity relative to external disturbances such as wind.
- We use a trajectory of pose, velocity, and force from the hybrid A⋆ solution as an initial guess to a general OCP solver.
- In the OCP, we utilize a sequence of convex polygons to generate a state corridor in a nonconvex obstacle map. This representation of obstacles causes the OCP's obstacle avoidance constraint to be convex, rather than nonconvex. Additionally, it allows us to easily use polygonal obstacle maps in the gradient-based OCP solver, which is generally hard due to their piecewise linear and nonconvex nature.
- We have compared our method to similar trajectory planning methods and found significant improvements in terms of run time, with equivalent energy use.
- We have performed full-scale experiments that have validated our method based on the experimental vessel's capability to track the resulting trajectory.

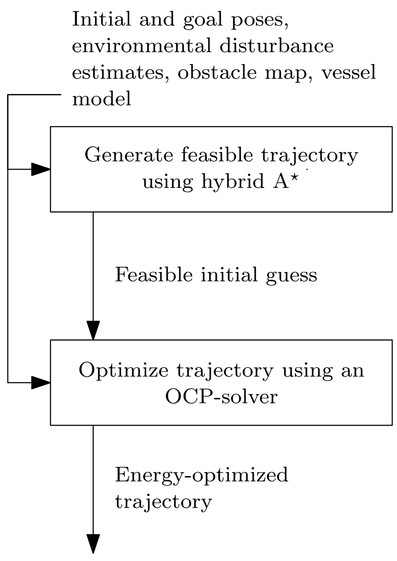

Initial and goal poses, environmental disturbance estimates, obstacle map, vessel model
Generate feasible trajectory using hybrid A⋆
Feasible initial guess
Optimize trajectory using an OCP-solver
Energy-optimized trajectory

FIGURE 1. A block diagram of the high-level functionality of our proposed trajectory planning method.

C. OUTLINE

In Section II we cover preliminary information about notation and vessel modeling. Sections III and IV present the development of our trajectory planning method. In Section III, we describe the hybrid A⋆ method that generates the initial guess. The section covers the generation of motion primitives, two different search heuristics, and the search algorithm itself. In Section IV, we present the OCP, how we convert the obstacle map to a sequence of convex polygons, the transcription of the OCP to a nonlinear program (NLP), and how we solve the NLP Section V contains simulations and comparisons to other trajectory planning methods. The results are compared in quantitative measures of planning time and energy-usage when tracking the trajectories. In Section VI, we present results from full-scale experiments, which serve as validation of the method and show how well the experimental vessel can track the produced trajectories. Section VII gives concluding remarks.

II. PRELIMINARIES

A. NOTATION

From LaValle [5], we have widely used notation related to *path* planning. As opposed to trajectories, a path places no temporal constraints on the following vehicle. Except for this, the two topics of planning paths and trajectories are similar. We let $\mathcal{W} := \mathbb{R}^2$ denote the world that contains our vessel and obstacles. The union of obstacles is $\mathcal{O} \subset \mathcal{W}$. The free workspace is defined to be $\mathcal{W}_{\text{free}} := \mathcal{W} \setminus \mathcal{O}$.

Our vessel lives in $\mathcal{W}$, but its configuration is better described in the configuration space

$$\boldsymbol{\eta} = \begin{bmatrix} x & y & \psi \end{bmatrix}^\top \in \mathcal{C} := \mathbb{R}^2 \times S. \tag{1}$$

Here, x and y are the vessel's position coordinates North and East of some origin, respectively, and ψ is its heading angle relative to North. The position coordinates refer to the vessel's center of gravity, which is at its centroid. The vector $\boldsymbol{\eta}$ is referred to as the vessel's *pose*. We denote its footprint in the workspace as a set of points $\mathcal{A}(\boldsymbol{\eta}) \subset \mathcal{W}$, which defines the vessel's shape. The set of noncolliding configurations is thus

$$\mathcal{C}_{\text{free}} := \{\boldsymbol{\eta} \in \mathcal{C} \mid \mathcal{A}(\boldsymbol{\eta}) \cap \mathcal{O} = \emptyset\}. \tag{2}$$

Most path planning algorithms operate on a discretized version of the configuration space, denoted by $\mathcal{C}_d \subset \mathcal{C}$. In our work we uniformly discretize the configuration space on a grid with resolution

$$\boldsymbol{r}_{\mathcal{C}} := \begin{bmatrix} r_p & r_p & r_h \end{bmatrix}^\top, \tag{3}$$

where $r_p > 0$ is the positional resolution and $r_h > 0$ is the angular heading resolution. Similarly, the discrete free configuration space is denoted $\mathcal{C}_{d,\text{free}}$. While points in the continuous configuration space are denoted by $\boldsymbol{\eta}$, we use a tilde for points in the discrete configuration space: $\tilde{\boldsymbol{\eta}}$. The mapping from $\mathcal{C}$ to $\mathcal{C}_d$ is denoted $\textsc{Key} : \mathcal{C} \mapsto \mathcal{C}_d$ and is done by rounding $\boldsymbol{\eta}$ to its closest multiple of $\boldsymbol{r}_{\mathcal{C}}$.

The formal goal of path planning is to find a continuous path, entirely in $\mathcal{C}_{\text{free}}$, from a start pose $\boldsymbol{\eta}_0 \in \mathcal{C}_{\text{free}}$ to a goal pose $\boldsymbol{\eta}_f \in \mathcal{C}_{\text{free}}$. In discrete algorithms, the paths are often piecewise linear, with connections on $\mathcal{C}_{d,\text{free}}$. Generally, this problem has many solutions, however, we usually also associate the problem with a definition of an *optimal* path, e.g., the shortest. In trajectory planning, the goal is similar, but we have additional kinodynamic constraints to satisfy, e.g., a set of time-parametrized differential equations. Section II-B introduces such constraints in the form of a mathematical vessel model.

B. ASV MODELING

Our ASV is modeled as a surge-decoupled three-degree-of-freedom displacement vessel, with the state vector

$$\boldsymbol{x} := \begin{bmatrix} \boldsymbol{\eta}^\top & \boldsymbol{\nu}^\top \end{bmatrix}^\top \in \mathcal{X} := \mathcal{C} \times \mathbb{R}^3 \tag{4}$$

with $\boldsymbol{\eta}$ being the pose described in (1), and $\boldsymbol{\nu} := [u, v, r]^\top$ the body-fixed velocity vector, where u is the surge velocity, v sway velocity and r yaw rate. The state space is denoted $\mathcal{X}$. The kinematic relationship between the pose and velocity is described by

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\boldsymbol{\nu}, \tag{5}$$

where

$$\mathbf{R}(\psi) := \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{6}$$

The kinetics of the ASV is described by

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{env}}. \tag{7}$$

This notation is widely used for vessel models in the maritime control literature [24]. Here, $\mathbf{M} \in \mathbb{R}^{3\times3}$ is the positive definite system inertia matrix, $\mathbf{C}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$ is the skew-symmetrix Coriolis and centripetal matrix, and $\mathbf{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$ is the positive definite damping matrix. The force vector $\boldsymbol{\tau} = [X, Y, N]^\top \in \mathcal{T} \subset \mathbb{R}^3$ are the control forces produced by the ASV's actuators in surge, sway and yaw, respectively, where $\mathcal{T}$ denotes the space of valid inputs. These are in turn governed by dynamical models of the actuators. For simulation purposes we include those models, but for planning and control we have simplified the model to let $\boldsymbol{\tau}$ be directly controllable. The environmental forces $\boldsymbol{\tau}_{\text{env}}$ can come from wind, ocean current and waves. We have only modeled wind effects for our experimental vessel, and the environmental forces are a function of relative wind velocity:

$$\boldsymbol{\tau}_{\text{env}} = \boldsymbol{\tau}_{\text{env}}(\psi, \boldsymbol{\nu}, \boldsymbol{V}_w), \tag{8}$$

where $\boldsymbol{V}_w \in \mathbb{R}^2$ is the wind velocity in North and East components. Matrices $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{D}$, along with the actuator models, as well as a wind model are defined in [25].

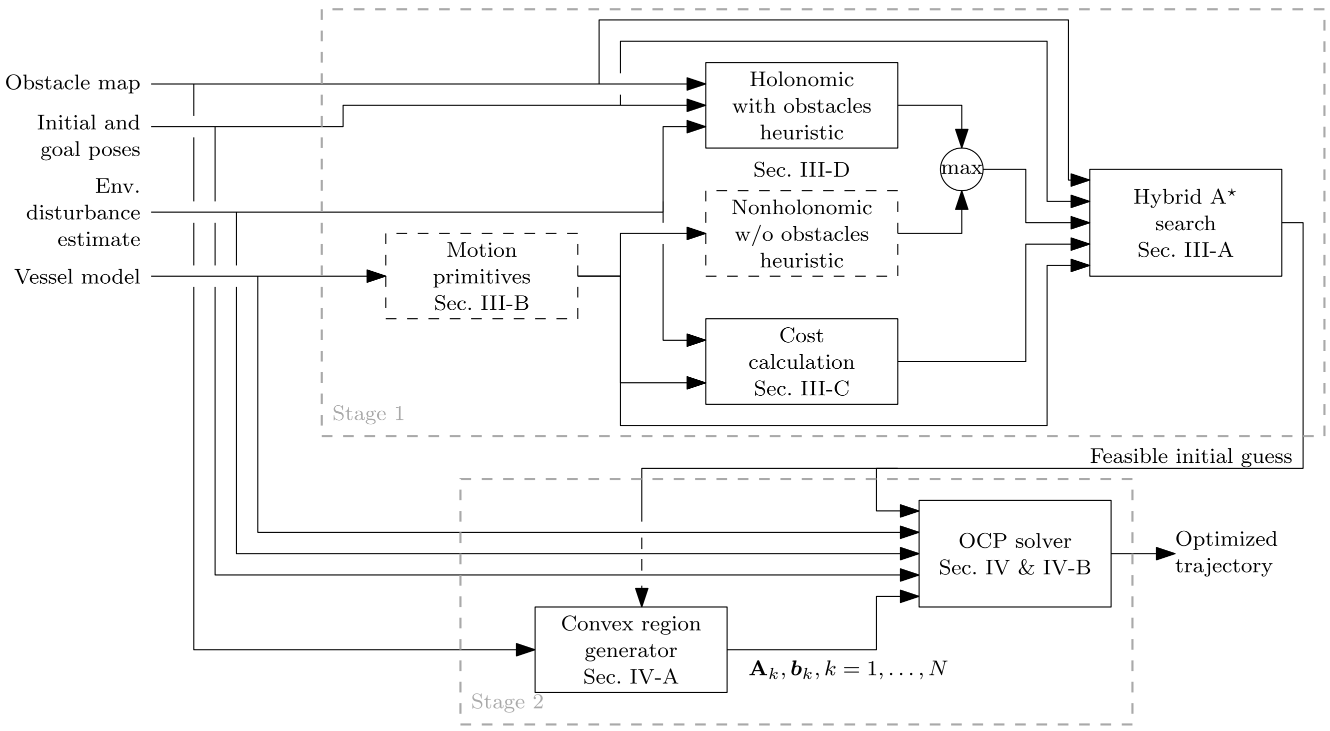
The model is concatenated to

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\tau}, \boldsymbol{V}_w) \\ &:= \begin{bmatrix} \mathbf{R}(\psi)\boldsymbol{\nu} \\ \mathbf{M}^{-1}\big[-\big(\mathbf{C}(\boldsymbol{\nu}) + \mathbf{D}(\boldsymbol{\nu})\big)\boldsymbol{\nu} + \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{env}}(\psi, \boldsymbol{\nu}, \boldsymbol{V}_w)\big] \end{bmatrix}\end{aligned} \tag{9}$$

for ease of reference when discussing OCPs later in the paper.

III. STAGE 1: GENERATING A DYNAMICALLY FEASIBLE INITIAL GUESS

As we mention in the introduction, our trajectory planning method comprises two stages. The entire method, its subcomponents, and their interconnections are illustrated in Figure 2. Each subcomponent will be described in this section and the next. Stage 1 of our method is to find a dynamically feasible trajectory using the hybrid A⋆ search.



Obstacle map
Initial and goal poses
Env. disturbance estimate
Vessel model
Holonomic with obstacles heuristic
Sec. III-D
Nonholonomic w/o obstacles heuristic
max
Motion primitives Sec. III-B
Cost calculation Sec. III-C
Hybrid A⋆ search Sec. III-A
Stage 1
Feasible initial guess
OCP solver Sec. IV & IV-B
Optimized trajectory
Convex region generator Sec. IV-A
$\mathbf{A}_k, \mathbf{b}_k, k = 1, \ldots, N$
Stage 2

FIGURE 2. **Block diagram of the trajectory planning method. Stage 1 refers to the generation of the initial guess, described in Section III, and Stage 2 refers to the trajectory optimization from Section IV.**

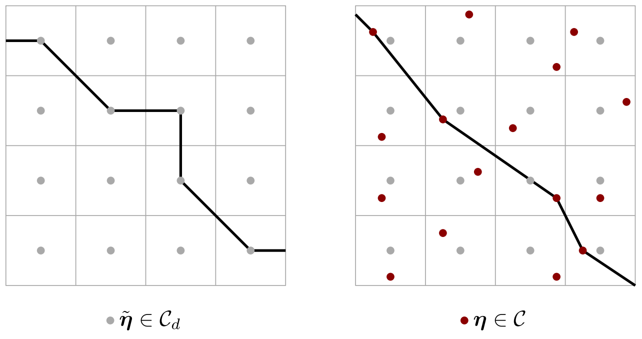


FIGURE 3. **Comparison of traditional A⋆ search space to the hybrid A⋆ search space in a two-dimensional grid. To the left is the commonly found eight-connected uniform grid associated with A⋆, where states are associated with grid cell centers. To the right is the search space of hybrid A⋆, where states can lie anywhere in the cells.**

A. HYBRID A⋆

Dolgov *et al.* [18] developed the hybrid A⋆ algorithm to plan paths for autonomous cars. Hybrid A⋆ is a variant of the well-known A⋆ algorithm that captures continuous-state data in discrete search nodes. The search space is discretized, but a continuous state is associated with each discrete node, as illustrated in Figure 3. An advantage of the hybrid A⋆ search space is that it does not require the connections between two states in different nodes to be exact, which allows us to be flexible when using motion primitives in the discrete search. A disadvantage is that the optimality from traditional A⋆ is no longer strictly guaranteed due to the merging of continuous and discrete states.

Algorithm 1 is pseudocode for the hybrid A⋆ search. Like an A⋆ search, it uses a priority queue to keep track of the open set. In Algorithm 1, that functionality is maintained by the Push and Pop functions. Push adds a key with a priority value to the open set O, while Pop removes and returns the key with the lowest associated priority. The mappings State, Cost, and Parent keep track of continuous states, cost values, and parents associated with discrete keys $\tilde{\eta} \in \mathcal{C}_d$. The mappings are updated as the search progresses. The function Primitives returns a set of motion primitives, Collision checks whether there is a collision, and Heuristics returns heuristic cost estimates. These functions are further described in sections III-B, III-D, and III-E, respectively.

B. MOTION PRIMITIVES

In the hybrid A⋆ search algorithm, new configurations are discovered by propagating motion primitives from an existing configuration. A motion primitive is a dynamically feasible state trajectory between two configurations in $\mathcal{C}$. Dynamic feasibility, as discussed in Section II-A, is inherently satisfied by using motion primitives with trajectories that satisfy (9). While we search in $\mathcal{C}$, the trajectories are in $\mathcal{X}$, which means that to feasibly connect two configurations with state trajectories in $\mathcal{X}$, they must start and end with the same velocities.

Motion primitives with varying lengths and turn angles are precomputed using an OCP. During the search, the primitives are translated and rotated to fit with the originating configuration. The motion primitives used in our results are shown in Figure 4.

Algorithm 1 Hybrid A⋆ Search Pseudocode

```
1:  function HYBRID A⋆( $\eta_0, \eta_f, V_w, \mathcal{O}$ )
2:      $\tilde{\eta}_0 \leftarrow$ KEY($\eta_0$), $\tilde{\eta}_f \leftarrow$ KEY($\eta_f$)
3:      $O \leftarrow \emptyset, C \leftarrow \emptyset$
4:      PUSH($O, \tilde{\eta}_0, 0$)
5:      STATE($\tilde{\eta}_0$) $\leftarrow \eta_0$, COST($\tilde{\eta}_0$) $\leftarrow 0$
6:      while $O \neq \emptyset$ do
7:          $\tilde{\eta} \leftarrow$ POP($O$)
8:          $C \leftarrow C \cup \{\tilde{\eta}\}$
9:          if $\tilde{\eta} = \tilde{\eta}_f$ then
10:             return sequence from $\tilde{\eta}_0$ to $\tilde{\eta}_f$
11:         $\eta \leftarrow$ STATE($\tilde{\eta}$)
12:         for all $\mathcal{P}, c, \eta_n \in$ PRIMITIVES($\eta, V_w$) do
13:             $\tilde{\eta}_n \leftarrow$ KEY($\eta_n$)
14:             if COLLISION($\mathcal{P}, \mathcal{O}$) or $\tilde{\eta}_n \in C$ then
15:                 continue
16:             $f \leftarrow$ COST($\tilde{\eta}$) $+ c$
17:             if $\tilde{\eta}_n \notin C \cup O$ then
18:                 COST($\tilde{\eta}_n$) $\leftarrow \infty$
19:             if $f <$ COST($\tilde{\eta}_n$) then
20:                 COST($\tilde{\eta}_n$) $\leftarrow f$
21:                 PARENT($\tilde{\eta}_n$) $\leftarrow \tilde{\eta}$
22:                 STATE($\tilde{\eta}_n$) $\leftarrow \eta_n$
23:                 $O \leftarrow O \setminus \{\tilde{\eta}_n\}$
24:                 $h \leftarrow f +$ HEURISTICS($\eta_n, \eta_f, V_w$)
25:                 PUSH($O, \tilde{\eta}_n, h$)
26:     return error, no path found
```

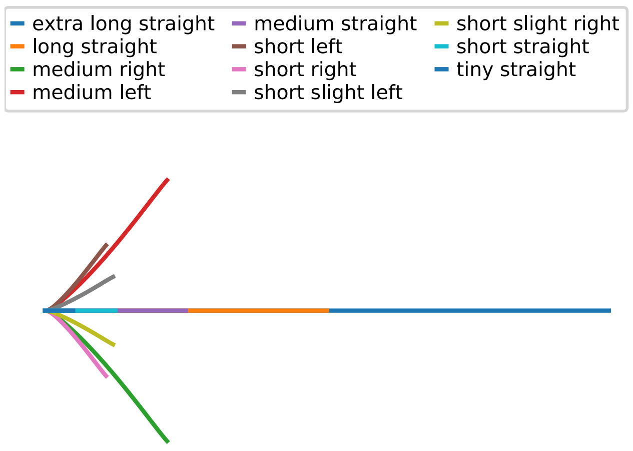


FIGURE 4. Motion primitives used in our results.

The OCP used to generate motion primitives is

$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{\tau}(\cdot)} \int_0^{t_f} F(\boldsymbol{x}(\tau), \boldsymbol{\tau}(\tau))\mathrm{d}\tau \tag{10a}$$

$$\text{subject to } \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{\tau}(t), \boldsymbol{0}_2) \quad t \in [0, t_f] \tag{10b}$$

$$\boldsymbol{x}_{lb} \le \boldsymbol{x}(t) \le \boldsymbol{x}_{ub} \quad t \in [0, t_f] \tag{10c}$$

$$\boldsymbol{\tau}_{lb} \le \boldsymbol{\tau}(t) \le \boldsymbol{\tau}_{ub} \quad t \in [0, t_f] \tag{10d}$$

$$\boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{10e}$$

$$\boldsymbol{x}(t_f) = \boldsymbol{x}_f. \tag{10f}$$

The OCP is equal for every primitive, except for the final time t_f, the state bounds (10c) and the final condition (10f), all of which depend on the motion primitive length $L > 0$ and direction angle χ. The vessel is assumed to travel with a nominal speed U_{nom}, which in our results is 1.5 m s^{-1}. For a specific primitive defined by (L, χ), the parameters of (10) are

$$t_f = L/U_{\text{nom}} \tag{11a}$$

$$\boldsymbol{x}_{lb} = \begin{bmatrix} \min(0, L\cos\chi) \\ \min(0, L\sin\chi) \\ \min(0, \chi) \\ u_{lb} \\ -v_{ub} \\ -r_{ub} \end{bmatrix} \tag{11b}$$

$$\boldsymbol{x}_{ub} = \begin{bmatrix} \max(0, L\cos\chi) \\ \max(0, L\sin\chi) \\ \max(0, \chi) \\ u_{ub} \\ v_{ub} \\ r_{ub} \end{bmatrix} \tag{11c}$$

$$\boldsymbol{\tau}_{lb} = \begin{bmatrix} X_{lb} & -Y_{ub} & -N_{ub} \end{bmatrix}^\top \tag{11d}$$

$$\boldsymbol{\tau}_{ub} = \begin{bmatrix} X_{ub} & Y_{ub} & N_{ub} \end{bmatrix}^\top \tag{11e}$$

$$\boldsymbol{x}_0 = \begin{bmatrix} 0 & 0 & 0 & U_{\text{nom}} & 0 & 0 \end{bmatrix}^\top \tag{11f}$$

$$\boldsymbol{x}_f = \begin{bmatrix} L\cos\chi & L\sin\chi & \chi & U_{\text{nom}} & 0 & 0 \end{bmatrix}^\top. \tag{11g}$$

The values u_{lb}, u_{ub}, v_{ub} and r_{ub} are velocity bounds, and X_{lb}, X_{ub}, Y_{ub}, and N_{ub} are bounds on surge force, sway force, and yaw moment, respectively. Table 1 specifies the parameters (L, χ) in our results, and Table 2 gives the boundary values.

TABLE 1. Motion primitive parameters.

Name	Length L [m]	Turn angle χ [°]
extra long straight	200	0
long straight	100	0
medium right	50	30
medium left	50	−30
medium straight	50	0
short right	25	30
short left	25	−30
short slight right	25	15
short slight left	25	−15
short straight	25	0
tiny straight	10	0

The OCP (10) contains a cost-to-go function:

$$F(\boldsymbol{x}, \boldsymbol{\tau}) = \overbrace{|\boldsymbol{\nu}|^\top \cdot |\boldsymbol{\tau}|}^{\text{energy}} + 1000\left((v/v_{ub})^2 + (r/r_{ub})^2\right) + 100\left((X/X_{ub})^2 + (Y/Y_{ub})^2 + (N/N_{ub})^2\right). \tag{12}$$

The cost's main contributor is energy spent but includes quadratic costs on velocity states and input forces. Without

TABLE 2. OCP boundary values for motion primitives.

U_{nom}	1.5	$\mathrm{m\,s^{-1}}$
u_{lb}	0	$\mathrm{m\,s^{-1}}$
u_{ub}	2.5	$\mathrm{m\,s^{-1}}$
v_{ub}	1.5	$\mathrm{m\,s^{-1}}$
r_{ub}	5	$^\circ\,\mathrm{s^{-1}}$
X_{lb}	-1000	N
X_{ub}	1000	N
Y_{ub}	1000	N
N_{ub}	1800	N m

these quadratic costs, the OCP becomes significantly harder to solve. The pure energy part of the cost function makes up ∼95% of the straight motion primitives' costs and ∼80% of the costs in turns.

The choice of length and direction parameters L and χ of the primitives are tightly connected to the resolutions defined in (3). At least one of the motion primitives must have a length longer than the diagonal of the grid cells defined by the positional resolution r_p in order to be guaranteed to traverse from one cell to another. We use a positional resolution of $r_p = 10\,\mathrm{m}$, so we need at least one primitive longer than $\sqrt{2} \cdot 10\,\mathrm{m} \approx 14.14\,\mathrm{m}$. Additionally, one of the primitives should have a length equal to r_p, so that the search does not "jump over" the goal cell. It will also ease the discrete search if the motion primitives' direction angles are multiples of the angular resolution r_h. The primitives in Table 1 include these important properties.

The positional resolution greatly affects the performance of the hybrid A⋆ search. A smaller resolution r_p makes the search space denser, which increases the computational load and time to find a solution, but improves the accuracy of the search.

The OCPs are transcribed to NLPs using direct collocation, and then solved using an interior point algorithm [26] offline prior to performing any search. The details of the transcription and solving are the same as in the main OCP-stage of our planning method – those details are found in Section IV-B.

In Algorithm 1, motion primitives from a configuration $\boldsymbol{\eta} \in \mathcal{C}$ are returned by the function PRIMITIVES. This function returns a sequence of geometrical paths $\mathcal{P} \in \mathcal{W}$, the cost of the maneuver c whose calculation is described in Section III-C, and the new neighboring state $\boldsymbol{\eta}_n \in \mathcal{C}$. The cost is dependent on the wind velocity $\boldsymbol{V}_w$. A mathematical description of the function is

$$\textsc{Primitives} : \mathcal{C} \times \mathbb{R}^2 \mapsto [\mathcal{W} \times \mathbb{R}^+ \times \mathcal{C}]_{1,\ldots,M}, \qquad (13)$$

where M is the number of motion primitives.

C. COST FUNCTION

While the OCP that generates the motion primitives uses the generic cost-to-go function (12), these OCPs are solved offline and have no information about environmental disturbances. Therefore, we need an alternative method to quickly calculate the energy usage of each maneuver online, when the disturbances are known or estimated. For calculating energy exerted to overcome environmental disturbances, we use the definition of mechanical work:

$$W_r = \int_0^{t_f} |\boldsymbol{\tau}_r|^\top \cdot |\boldsymbol{\nu}_r|\, \mathrm{d}t, \qquad (14)$$

where we use the absolute values since there is no energy regeneration in the ASV's propulsion system. In this calculation, the subscript $(\cdot)_r$ denotes *relative* values, e.g., the force needed to overcome relative wind velocity. The work required to move through the wind is

$$W_{\text{wind}} = \int_0^{t_f} |\boldsymbol{\tau}_w|^\top \cdot \left| \boldsymbol{\nu} - \mathbf{R}(\psi)^\top \begin{bmatrix} \boldsymbol{V}_w \\ 0 \end{bmatrix} \right| \mathrm{d}t, \qquad (15)$$

where $\boldsymbol{\tau}_w$ is the force needed to overcome wind effects, calculated with our wind model. We have assumed zero ocean currents for moving through the water since the vessel we are working with has a very shallow and flat hull. Additionally, we do not have access to accurate information about ocean currents in our test areas. The work required to move through the water is then

$$W_{\text{water}} = \int_0^{t_f} |\mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu}|^\top \cdot |\boldsymbol{\nu}|\, \mathrm{d}t. \qquad (16)$$

The total energy cost $c = W_{\text{wind}} + W_{\text{water}}$ is calculated by propagating the integrands of (15) and (16) over the discretized solution trajectories from (10) with the appropriate wind velocity. This relative energy formulation is inspired by [27].

D. COLLISION CHECKING

For each solution trajectory generated by (10), the position state trajectories $x(\cdot)$ and $y(\cdot)$ make up the vessel's geometrical footprint in $\mathcal{W}$. After translating and rotating a motion primitive, the geometrical footprint is checked for overlap with $\mathcal{O}$, and a collision is reported if that is the case. The geometrical footprint is diluted by a clearance radius r_c to account for the shape of the vessel and additional clearance to keep a proper distance from obstacles. The clearance radius and footprint are illustrated in Figure 5. Our vessel is rectangular with a shape of 5 m by 2.8 m, and we use a clearance

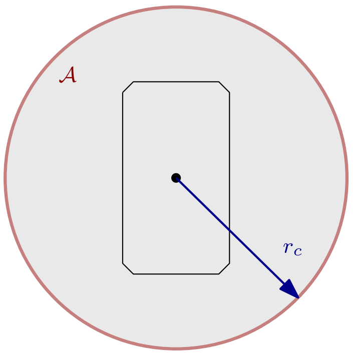


FIGURE 5. Vessel shape along with the clearance radius r_c which defines the footprint $\mathcal{A}$ used for collision checking.

radius of $r_c = 10\,\text{m}$. The Collision function in Algorithm 1 performs the collision checking:

$$\text{Collision} : \mathcal{W} \times \mathcal{W} \mapsto \{\text{true}, \text{false}\}\,. \tag{17}$$

E. SEARCH HEURISTICS

To guide the hybrid A⋆ search, we use heuristic cost functions. These are functions that estimate the remaining cost from a node in $\mathcal{C}_d$ to the goal node. The search will prioritize exploring nodes with the lowest estimated total cost. In a traditional A⋆ search, using *admissible* heuristic functions, i.e., functions that never overestimate the true cost, maintains a Dijkstra search's optimality guarantee. However, hybrid A⋆ does not have any optimality guarantees due to the merging of continuous states in discrete "bins," so the heuristic functions' admissibility is not as important.

Similar to [18], we combine two different heuristic functions. We employ a *holonomic with obstacles* heuristic that guides the search towards the two-dimensional cheapest path, and a *nonholonomic without obstacles* heuristic that avoids trajectories that the ASV cannot feasibly follow. Their designs are described in the following, and they are combined using the maximum of the two heuristics.

The description of the Heuristics function from Algorithm 1 is

$$\text{Heuristics} : \mathcal{C} \times \mathcal{C} \times \mathbb{R}^2 \mapsto \mathbb{R}^+, \tag{18}$$

where the function maps the current state $\boldsymbol{\eta}$, the goal state $\boldsymbol{\eta}_f$, and the wind velocity $\boldsymbol{V}_w$ to a positive scalar.

1) HOLONOMIC WITH OBSTACLES

The holonomic with obstacles heuristic uses a simple model that can move in any direction without the nonholonomic constraint of moving along the vessel's heading angle. It considers the obstacle map $\mathcal{O}$ and assigns costs to nodes using a breadth-first search on a two-dimensional grid with resolution r_p. Instead of the standard eight-connected graph illustrated in Figure 3, we use a 16-connected graph, as seen in Figure 6, to allow more movement angles. We use the same cost function described in Section III-C, which results in a mapping from every node in $\mathcal{C}_{d,\text{free}}$ to a positive scalar that estimates the remaining cost to navigate to the goal node. Figure 7 shows an example of the mapping near a harbor.

The 16-connected breadth-first search is limiting since it biases towards paths with the same directions as the graph

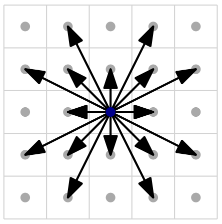


FIGURE 6. 16-connected graph. In this connectivity scheme, edges are added to all nodes two layers from the center node, unless the travel direction already exists in an inner layer.

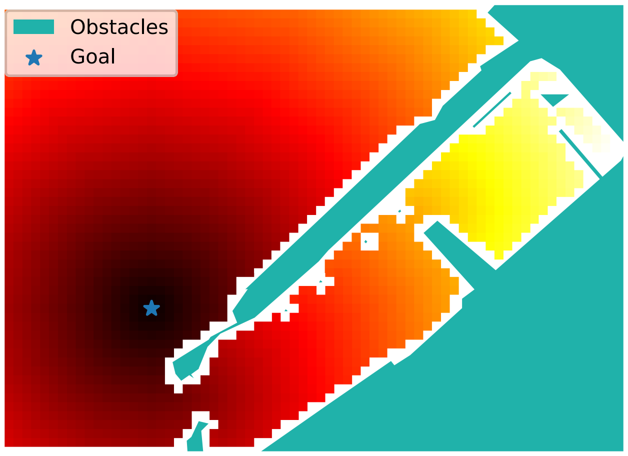


FIGURE 7. Example of the holonomic with obstacles heuristic function on a map. Brighter squares are more costly.

connectivity in Figure 6, i.e., on ∼22° increments. Without disturbances, the error between the real cost function and the heuristic averages 1.8% in an obstacle-free map of 1 km by 1 km.

Alternative heuristics include the fast marching method, which can calculate a cost function in the presence of obstacles without bias to particular directions. Standard implementations of the fast marching method [28] do not support the inclusion of a directional component in the cost function, on which we rely. Implementations of the fast marching method subject to a vector field are available [29], [30]. Furthermore, graph searches with simplified models can function as guiding heuristics, demonstrated in [15].

Since the calculation of our holonomic-with-obstacle heuristic requires information about the goal location and disturbances, the mapping has to be calculated online.

2) NONHOLONOMIC WITHOUT OBSTACLES

The dual to the holonomic with obstacles heuristic is one that considers nonholonomic movements without obstacles. This heuristic places high costs on nodes that lead to trajectories the ASV cannot feasibly follow. It utilizes the motion primitives from Section III-B and performs a breadth-first hybrid A⋆ search from every node in a limited, rectangular, collision-free grid around the origin of $\mathcal{C}_d$. This results in a mapping from the included nodes in $\mathcal{C}_d$ to a positive scalar and is precomputed offline. The mapping is translated and rotated to the desired goal node when used in the search. Figure 8 shows the heuristic mapping for different initial heading angles.

Since the environmental disturbances are unknown at the time of precomputation, we cannot say anything about the effects these disturbances have on the cost. However, this heuristic is only active in the final part of the search, and we argue that the energy-optimality criterion is less critical in this stage. Additionally, the optimization stage described

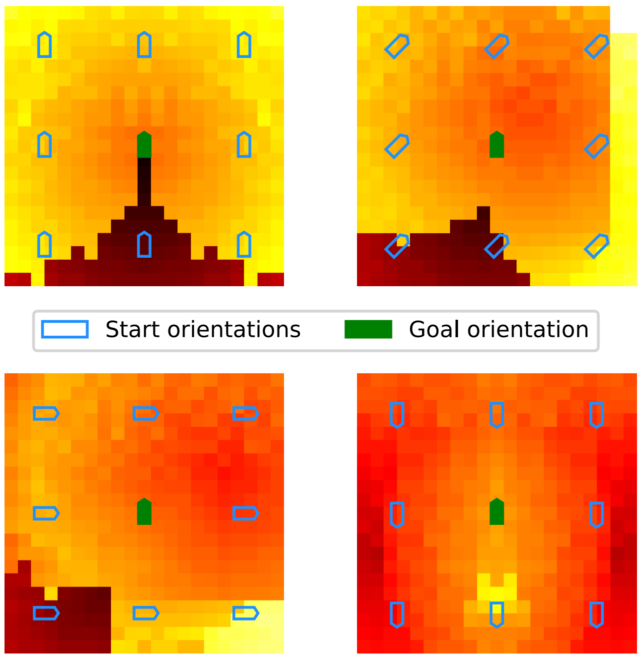


FIGURE 8. **Plot of the nonholonomic without obstacles heuristic function for initial heading angles 0°, 45°, 90° and 180°. Brighter squares are more costly.**

in Section IV locally optimizes the trajectory accounting for known or estimated disturbances.

F. SEARCH OUTPUT

A search is completed when the goal node is discovered by a motion primitive. The result is a chain of nodes from the goal node towards the start node by following their parents. This chain is reversed, and the resulting sequence of motion primitives are concatenated into forming the solution trajectories

$$x^\star : [0, t_f^\star] \mapsto \mathcal{X} \tag{19a}$$
$$\tau^\star : [0, t_f^\star] \mapsto \mathcal{T}, \tag{19b}$$

which are valid on the time interval $[0, t_f^\star]$, where $t_f^\star$ is the sum of the motion primitive durations. In practice, these mappings are a discrete sequence of points in the state and input spaces ($\mathcal{X}$ and $\mathcal{T}$), interpolated to form time-continuous trajectories. The points' density depends on the number of shooting intervals used when solving (10).

To summarize Stage 1, it consists of a hybrid A⋆ search guided by two heuristics, propagating motion primitives that lead from the start pose to the desired end pose. Since the trajectory so far consists of only the motion primitive maneuvers, it must be improved to find an optimized trajectory in the continuous search space.

IV. STAGE 2: TRAJECTORY OPTIMIZATION

The second stage of the trajectory planner is to solve an OCP that describes the trajectory planning problem. Stage 2 in Figure 2 shows the subcomponents of this trajectory optimization. The OCP is similar to (10) in Section III-B. The initial and final conditions are different, we include external disturbances, and we have added obstacle avoidance constraints. Additionally, the final time is a free optimization variable. We restate the OCP, including the stated changes:

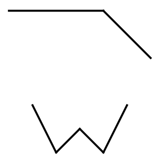
$$\min_{\boldsymbol{x}(\cdot),\boldsymbol{\tau}(\cdot),t_f} \int_0^{t_f} F(\boldsymbol{x}(\tau), \boldsymbol{\tau}(\tau))\mathrm{d}\tau \tag{20a}$$
$$\text{subject to } \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{\tau}(t), \boldsymbol{V}_w) \quad t \in [0, t_f] \tag{20b}$$
$$\boldsymbol{x}_{lb} \le \boldsymbol{x}(t) \le \boldsymbol{x}_{ub} \quad t \in [0, t_f] \tag{20c}$$
$$\boldsymbol{\tau}_{lb} \le \boldsymbol{\tau}(t) \le \boldsymbol{\tau}_{ub} \quad t \in [0, t_f] \tag{20d}$$
$$\boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{20e}$$
$$\boldsymbol{x}(t_f) = \boldsymbol{x}_f \tag{20f}$$
$$\mathbf{A}_k \cdot \begin{bmatrix} x(t_k) & y(t_k) \end{bmatrix}^\top \le \boldsymbol{b}_k - r_c \quad k = 1, \ldots, N \tag{20g}$$
$$0 \le t_f \le t_f^\star. \tag{20h}$$

The initial and final conditions are replaced with the initial and final desired pose, with zero velocities. The cost-to-go function is the same, as are the velocity and force bounds. Equation (20g) encodes obstacle avoidance constraints, which will be described in Section IV-A. Since the final time is a free variable, we place bounds on it in (20h). The transcription and solution process is described in Section IV-B.

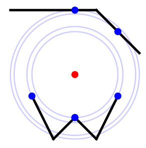
A. CONVEX COLLISION AVOIDANCE CONSTRAINTS

The OCP contains obstacle avoidance constraints in the form of halfspaces in the matrix-vector form (20g). The halfspaces are defined for the points in time t_k, $k = 1, \ldots, N$, where N is the number of shooting intervals used in the transcription of (20). With $h = t_f/N$ being the shooting interval duration, we have $t_k = h \cdot k$. The convex regions that define the obstacle avoidance constraints are generated along the solution of the hybrid A⋆ trajectory, i.e., the initial guess. The positional part of the state trajectory $\boldsymbol{x}^\star(\cdot)$ from (19a) is denoted $\boldsymbol{p}^\star(\cdot) = [x^\star(\cdot), y^\star(\cdot)]^\top$. For the points in time $t_k, k = 1, \ldots, N$, the parameters $\mathbf{A}_k \in \mathbb{R}^{m_k \times 2}$ and $\boldsymbol{b}_k \in \mathbb{R}^{m_k}$ are generated based on the obstacle map $\mathcal{O}$ with $\boldsymbol{p}^\star(t_k)$ being the *generator points*.

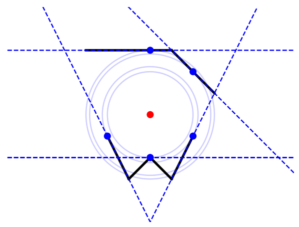
To create the convex region constraints, we use an algorithm that calculates an inner approximation of the obstacle map based on the polygons' edges in that map. The process is summarized as follows: Given a generator point $\boldsymbol{p}^\star(t_k)$, grow a circle centered at $\boldsymbol{p}^\star(t_k)$ until it reaches a point $\boldsymbol{p}_{c,k}$ where it touches an obstacle, and then create a constraint tangent to the expansion circle at the point at $\boldsymbol{p}_{c,k}$. Continue growing and create constraints until no further growth is possible. The process is illustrated in Figure 9. The parameters $\mathbf{A}_k$ and $\boldsymbol{b}_k$



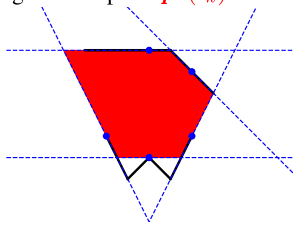
(a) Line segments of polygonal obstacles.



(b) Compute closest points $\boldsymbol{p}_{c,k}$ to generator point $\boldsymbol{p}^\star(t_k)$.



(c) Find normal lines to the line segments from $\boldsymbol{p}^\star(t_k)$ to $\boldsymbol{p}_{c,k}$, that pass through $\boldsymbol{p}_{c,k}$.



(d) Final convex inner approximation.

FIGURE 9. Illustration of how to compute the convex spatial constraints.

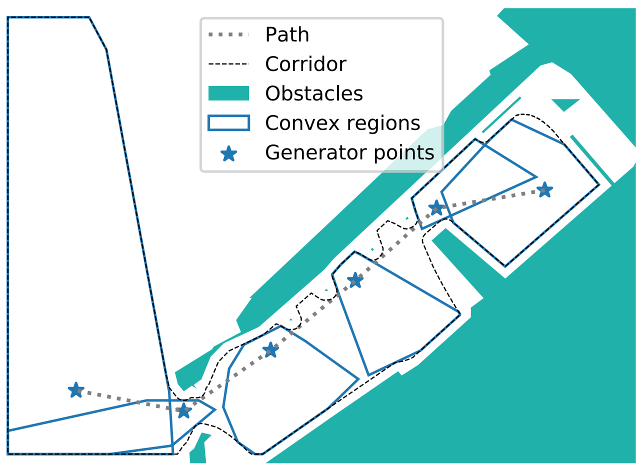


FIGURE 10. Example of convex regions along an arbitrary path with generator points spaced by 100 m. In the OCP the spacing would be ~1.5 m, causing dense overlapping, resulting in a corridor as depicted in the figure.

are defined by

$$\underbrace{\begin{bmatrix} \frac{(\boldsymbol{p}_{c,k,1}-\boldsymbol{p}^\star(t_k))^\top}{\|(\boldsymbol{p}_{c,k,1}-\boldsymbol{p}^\star(t_k))\|_2} \\ \frac{(\boldsymbol{p}_{c,k,2}-\boldsymbol{p}^\star(t_k))^\top}{\|(\boldsymbol{p}_{c,k,2}-\boldsymbol{p}^\star(t_k))\|_2} \\ \vdots \\ \frac{(\boldsymbol{p}_{c,k,m_k}-\boldsymbol{p}^\star(t_k))^\top}{\|(\boldsymbol{p}_{c,k,m_k}-\boldsymbol{p}^\star(t_k))\|_2} \end{bmatrix}}_{\mathbf{A}_k} \boldsymbol{p} \leq \underbrace{\begin{bmatrix} \frac{(\boldsymbol{p}_{c,k,1}-\boldsymbol{p}^\star(t_k))^\top \boldsymbol{p}_{c,k,1}}{\|(\boldsymbol{p}_{c,k,1}-\boldsymbol{p}^\star(t_k))\|_2} \\ \frac{(\boldsymbol{p}_{c,k,2}-\boldsymbol{p}^\star(t_k))^\top \boldsymbol{p}_{c,k,2}}{\|(\boldsymbol{p}_{c,k,2}-\boldsymbol{p}^\star(t_k))\|_2} \\ \vdots \\ \frac{(\boldsymbol{p}_{c,k,m_k}-\boldsymbol{p}^\star(t_k))^\top \boldsymbol{p}_{c,k,m_k}}{\|(\boldsymbol{p}_{c,k,m_k}-\boldsymbol{p}^\star(t_k))\|_2} \end{bmatrix}}_{\mathbf{b}_k}. \tag{21}$$

A point $\boldsymbol{p} \in \mathbb{R}^2$ is inside the convex region if the inequality constraints are satisfied, which is (20g) in the OCP. The number of halfspaces that make up a specific region is denoted m_k, $k = 1, \ldots, N$, and has an upper limit, in our case 12. The unit dimension of this inequality is distance, and a subtraction of the right-hand side of (21) shrinks the convex regions, implicitly increasing the clearance by, e.g., r_c, which is the clearance radius from Figure 5, used in (20g).

Figure 10 shows an example of convex regions using an arbitrary path as the basis for generator points. For each point in time t_k, the OCP may freely adjust the ASV's position inside the respective convex region. With dense overlapping, this allows the ASV to travel inside a corridor along the initial guess.

The convex regions constrain only a discrete set of points in the state trajectory $(\boldsymbol{x}(t_k), k = 1, \ldots, N)$. This limitation means that the points in between can violate the collision avoidance constraints. However, the vessel's dynamics restrict the trajectory's velocity, thus limiting the movement in a neighborhood around $\boldsymbol{x}(t_k)$. Having a short shooting interval duration h gives satisfactory collision avoidance behavior. In our results, we use a density of $h \approx 1$ s.

B. TRANSCRIPTION AND SOLVER

To solve the continuous OCP (20), we discretize it into an NLP. We use direct collocation with three Legendre collocation points per shooting interval to discretize the dynamics (20b). Both the state and input trajectories are encoded as polynomials over N shooting intervals. In our results, the number of shooting intervals is determined by the estimated final time $t_f^\star$ from the hybrid A* results in Section III-F. An initial shooting interval duration of $h^\star = 1$ s determines $N = \lfloor t_f^\star / h^\star \rfloor + 1$, while since the final time t_f is a free variable with upper bound $t_f^\star$, the actual shooting interval duration can be shorter. The cost function is determined by propagating the quadrature integral (20a) along the state and input polynomials. The resulting NLP is

$$\min_{\boldsymbol{w}} \phi(\boldsymbol{w}) \tag{22a}$$
$$\text{subject to } \boldsymbol{w}_{lb} \leq \boldsymbol{w} \leq \boldsymbol{w}_{ub} \tag{22b}$$
$$\boldsymbol{g}_{lb} \leq \boldsymbol{g}(\boldsymbol{w}) \leq \boldsymbol{g}_{ub}. \tag{22c}$$

The decision variables $\boldsymbol{w}$ include states and inputs at all collocation points, and the final time t_f. The bounds (22b) are box bounds on all the decision variables and encode the state and input constraints (20c) through (20f), and (20h). The function $\boldsymbol{g}$ and its bounds in (22c) encode the dynamics (20b) in addition to the obstacle avoidance constraints (20g).

The NLP is solved using the interior point algorithm "Ipopt" by Wächter and Biegler [26]. Since the initial guess provided by the hybrid A* algorithm results in minimal violations of the constraints, the initial value of the auxiliary boundary parameter μ in Ipopt is set quite low to 1×10^{-6}, compared to its default value of 1×10^{-1}. This reduction causes fast convergence of the solution.

Solving (22) provides the optimal decision variables $\boldsymbol{w}^{\diamond}$. These are converted to optimal trajectories

$$\boldsymbol{x}^{\diamond} : [0, t_f^{\diamond}] \mapsto \mathcal{X} \tag{23a}$$

$$\boldsymbol{\tau}^{\diamond} : [0, t_f^{\diamond}] \mapsto \mathcal{T}, \tag{23b}$$

where $t_f^{\diamond}$ is the optimal final time. Accurate interpolation of the discrete values returned from the solver is achieved by using the polynomial definition of the state and input trajectories.

C. METHOD SUMMARY

Figure 2 illustrates how all the subcomponents of our method are connected. Stage 1 performs a discrete search with continuous states using the hybrid A* algorithm guided by two heuristics and propagating the states with motion primitives. This results in a dynamically feasible initial guess for an energy-optimized trajectory between the start and goal poses. The resulting trajectory consists of a sequence of the motion primitives from Section III-B, limiting the search space to only those maneuvers. Therefore the trajectory cannot be optimal with respect to our cost functional. Stage 2 is a trajectory optimization step that uses the initial guess for two purposes: 1) To provide a sequence of convex and smooth polygonal constraints that represent a collision-free corridor from start to goal, and 2) to warm-start the OCP solver. The convex polygonal constraints are constructed with the process shown in Figure 9 and allow the OCP solver to handle the inherently nonconvex obstacle avoidance problem easily. Combined, this gives us a fast solution to (20), which is a locally optimal and dynamically feasible trajectory between the start and goal poses.

V. SIMULATION RESULTS

In this section, we describe the simulation and control setup used to evaluate our planning method and present the evaluation itself. We evaluate our method by performing planning and simulation in various scenarios and wind conditions and comparing our planner to other methods.

A. SIMULATOR AND CONTROL SYSTEM

The different trajectory planning methods are tested in a software-in-the-loop vessel simulator. The simulator comprises dynamic models of the vessel, its actuators, and its control systems. The vessel model is described in Section II-B, and the simulator performs Runge-Kutta 4 integration to propagate the differential equations. Additionally, the actuators' propeller and azimuth dynamics are simulated, whose models are available in [25].

The vessel's control system for trajectory tracking is divided into two layers, as seen in Figure 11: A trajectory-tracking dynamic positioning (DP) controller and a thrust allocation algorithm. The DP controller consists of a PID feedback term and a model-based feed-forward term for velocity and acceleration. Its details are available in [31, Section 3.4]. The controller sends the desired force

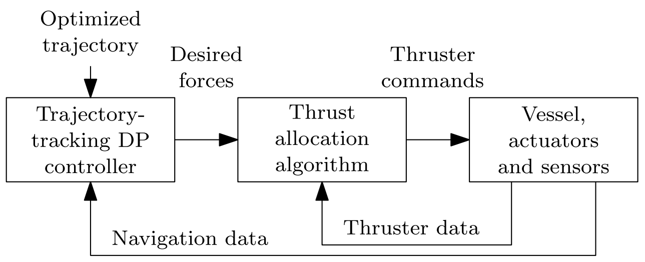

Optimized trajectory
Desired forces
Thruster commands
Trajectory-tracking DP controller
Thrust allocation algorithm
Vessel, actuators and sensors
Navigation data
Thruster data

FIGURE 11. Vessel control system architecture.

output to the thrust allocation algorithm, which in turn sends thruster commands to the vessel's actuators. This thrust allocation algorithm is described in [32].

For evaluation, energy use is measured by integrating the simulated power output, similar to the energy-part of (12):

$$E = \int_{t_0}^{t_f} |\boldsymbol{\nu}(t)|^\top \cdot |\boldsymbol{\tau}(t)| \, \mathrm{d}t. \tag{24}$$

B. EVALUATING THE EFFECT OF INCLUDING DISTURBANCE INFORMATION

One of the goals while developing the method was the ability to include known or estimated disturbance effects in both planning stages. To magnify the effects of wind on planning, we have designed a scenario where the starting point and goal are far apart, and the vessel is under the influence of crosswinds. Figure 12 shows the scenario where the plan is to sail from south to north.

The scenario is planned twice. Once when no wind information is included in the search, assuming that the wind velocity is $\boldsymbol{V}_w = [0, 0]$ when it is, in fact, $\boldsymbol{V}_w = [0, 3]\,\mathrm{m\,s^{-1}}$, and once using the correct wind velocity. The warm start solutions from the hybrid A* search differ significantly in the two cases, as shown Figure 12. However, the optimized trajectories of the two plans are nearly identical. Additionally, the power outputs from the simulated trackings are not that different – the total energy use for the two scenarios are 170 W h when not accounting for wind in the planning, compared to 164 W h when including wind information, a mere 3.5% improvement, attributed mainly to a difference in heading during transit.

In practice, models of how wind affects a ship are uncertain. For such a low improvement, it might not be beneficial to include wind effects when planning a long-term trajectory. Adding this information may worsen the result if the wind model or wind velocity estimates are erroneous. Including environmental disturbances may be more appropriate for other types of vessels or other types of disturbances, such as waves and ocean currents.

C. COMPARISONS TO OTHER TRAJECTORY PLANNING METHODS

Our method is compared to two other trajectory planning methods by planning a trajectory in the same scenario with all three methods. The two other planning methods are a

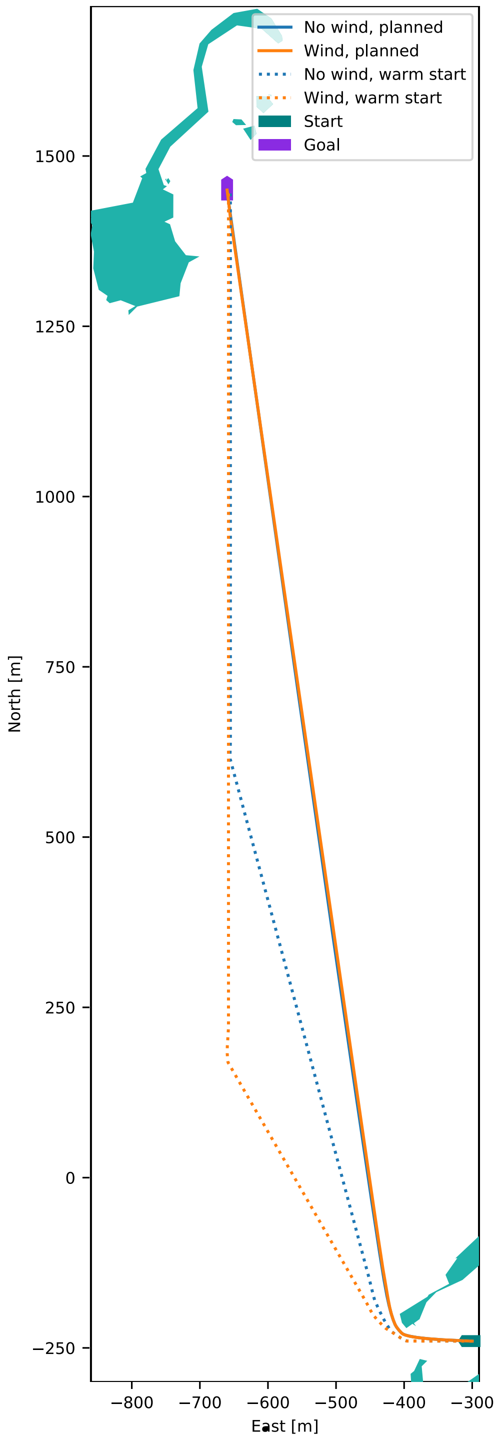


FIGURE 12. Comparison of trajectories planned without and with knowledge of simulated wind conditions. Described in Section V-B.

warm-started optimization scheme developed in [14], labeled C1 in the plots, and an optimal control-based complete cell decomposition method from [23] labeled C2. Our method is labeled TP. These two methods are selected for comparison because they are both optimization-based methods. C1 is similar in terms of the warm-starting methodology, and C2 is interesting because of the map discretization's completeness.

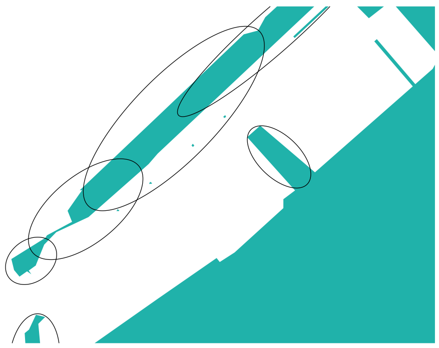


FIGURE 13. Elliptic obstacles that approximately match our map. Used when planning with the C1 method in Section V-C1.

1) C1: ALTERNATIVE WARM-STARTED OPTIMIZATION METHOD

The method developed in [14] uses a similar approach to our method. The main difference is how the warm start is generated and how the obstacles are represented in the optimization stage. C1 uses a standard A* search on an 8-connected uniformly discretized grid to search for the *shortest* path. That search results in a piecewise linear path which is converted to a trajectory by smoothing the connections with circular arcs and adding artificial dynamic information. The trajectory is not dynamically feasible with respect to the ASV's model, but it is used as the initial guess for an OCP solver. The OCP solver represents obstacles as inequalities in the form of ellipses, which are smooth representations, suitable for an optimization problem, but cannot accurately represent polygonal maps.

To compare TP to C1, we adjust the cost-to-go function in [14] to be equivalent to (12). Additionally, we have created elliptic obstacles to approximately match the polygonal obstacles which define our map, seen in Figure 13. We plan and simulate with zero wind, and with the initial and final poses, as shown in Figure 14. From the figure, we see that the resulting trajectories differ only slightly, mainly due to the different obstacle constraints. In the simulation, the trajectories give equal energy consumption, both at 52 W h. Figure 15 shows significant positional tracking error at the start and end of the transit, for both TP and C1. The vessel and its control system cannot track the acceleration that happens from and to a standstill. The models used in trajectory planning do not consider actuator dynamics or the control system, which probably is the cause of these errors. The positional tracking errors are comparable between TP and C1 for the remainder of the transit, with an error of 1 m around the turn and negligible error for the straights. Similar deviations are also evident in the heading, due to the coupling between linear and angular velocities. The positional error in Figure 15 is calculated as $||[x(t), y(t)] - [x^\diamond(t), y^\diamond(t)]||_2$, while the heading error is $\psi(t) - \psi^\diamond(t)$.

2) C2: COMPLETE OPTIMIZATION-BASED CELL DECOMPOSITION

Martinsen *et al.* [23] have developed an optimization-based trajectory planner that searches for a trajectory by

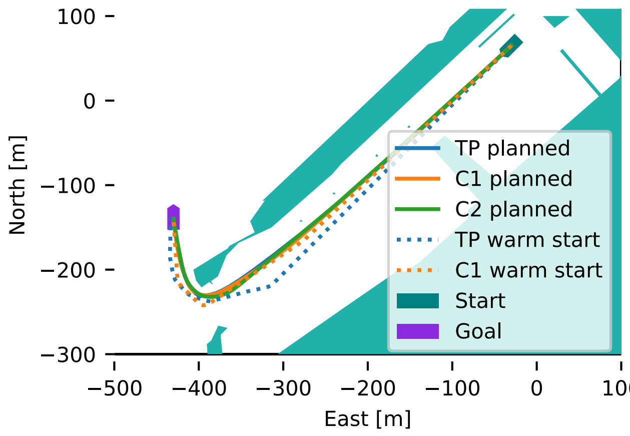


FIGURE 14. Comparison of trajectories planned with different methods from Section V-C. Results from our method are labeled TP, while C1 and C2 denote the other planning methods.

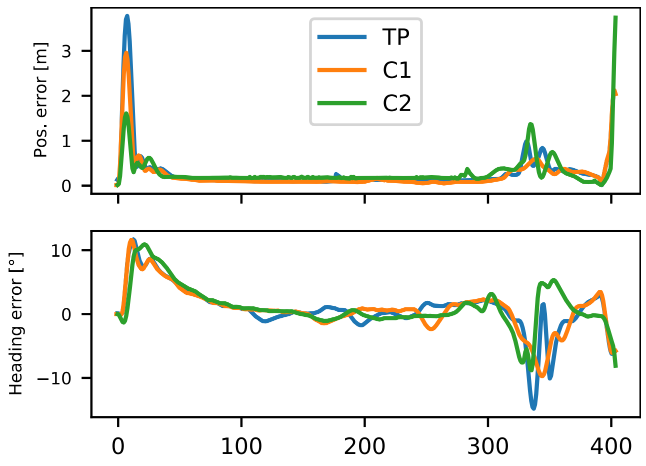


FIGURE 15. Tracking errors from the simulation comparisons. Results from our method are labeled TP, while C1 and C2 denote the other planning methods.

considering sequences of collision-free triangles from a constrained Delaunay triangulation of the workspace. C2 finds a globally optimal trajectory for linear models regardless of the inherent non-convex obstacles due to the cell decomposition by triangulation.

A trajectory with the same initial and final positions, as described in Section V-C1, is generated using a similar cost-to-go function and a simplified dynamic model. Figure 14 shows that also with C2, the trajectory difference is minimal. The differing cost-to-go function and dynamic model may cause the small differences we see. The slight difference may be caused by the differing cost-to-go function and the dynamic model used. As TP and C1, C2 gives an energy consumption of 52 W h. The tracking errors are similar between TP and C2, with better performance at the start of the trajectory for C2, as we see in Figure 15.

D. COMPLEX SCENARIO

The previous planning scenarios have been simple, with obvious routing choices. Figure 16 shows a more complex

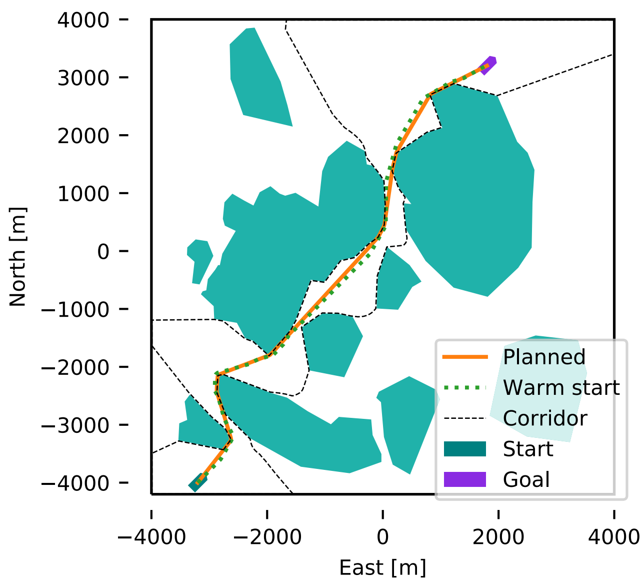


FIGURE 16. Planning in a map of Sjernarøyane, Norway – a more complex scenario with multiple routing options. Described in Section V-D.

scenario with multiple routing options. Our method was able to find the most direct and energy-efficient routing and optimized a trajectory from start to goal in 75 s. The figure also shows the corridor composed of the union of convex regions that allow the OCP to optimize freely. The resulting trajectory is dynamically feasible and adheres to the obstacle clearance constraints.

E. CONCLUSION

Including available wind information when planning a scenario yielded negligible improvements, shown in Section V-B. Only minor differences are found in the state trajectories. We conclude that there is no benefit to energy consumption for our application and vessel model when including wind estimates in trajectory planning. This conclusion is supported by the fact that there will be significant uncertainties in both wind estimates and wind force models.

Compared to two other optimization-based trajectory planning methods in Section V-C, we have shown that our method produced a similar trajectory with equal energy consumption. This similarity verifies that our method can find a desirable optimized trajectory with good energy performance. Significant improvements in runtime are achieved by using our method in this scenario, highlighted in Table 3.

TABLE 3. Performance comparisons for simulated planning scenarios in Section V-C.

Method	Energy usage [W h]	Planning time [s]
TP	52	**31**
C1	52	57
C2	52	785

As we show in Section V-D, our method can find the most reasonable trajectory in a complex routing environment, which is a major challenge when using purely optimization-based trajectory planning methods.

VI. EXPERIMENTAL VALIDATION

To validate that our method will produce collision-free, dynamically feasible trajectories, we have applied it in a full-scale experiment with *milliAmpere*, an experimental autonomous ferry developed at NTNU, depicted in Figure 17. The specifications of *milliAmpere*, its sensors, and control systems are found in Table 4, and it uses the same control setup as described in Section V. We tested the planning method and tracking capabilities in the Trondheim harbor area, using the same scenario as in Section V-C. On the day of testing, we measured a light breeze from North-northeast, but we were shielded by a breakwater for most of the route, causing us to experience almost no wind. We tested planning with zero wind and with the measured wind, finding a difference in measured energy use of 2%, which we deem insignificant given the measurement uncertainties, and thus we only present the results from planning with zero wind. Energy use is measured by integrating power as determined by the voltage and current measured on both the azimuth thrusters' propeller motors:

$$E = \int_{t_0}^{t_f} \left(|I_1(t) \cdot U_1(t)| + |I_2(t) \cdot U_2(t)|\right) \mathrm{d}t, \tag{25}$$

where I_i and U_i are motor current and voltage, respectively, for $i \in \{1, 2\}$.

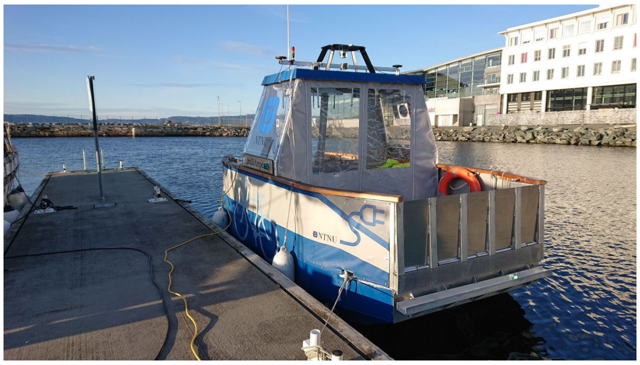


FIGURE 17. Picture of the experimental autonomous ferry *milliAmpere*.

TABLE 4. *milliAmpere* specifications.

Dimensions	5 m by 2.8 m symmetric footprint
Position and heading reference system	Vector VS330 dual GNSS with RTK capabilities
Thrusters	Two azimuth thrusters on the center line, 1.8 m aft and fore of center
Existing control modules	Trajectory-tracking DP controller and thrust allocation system

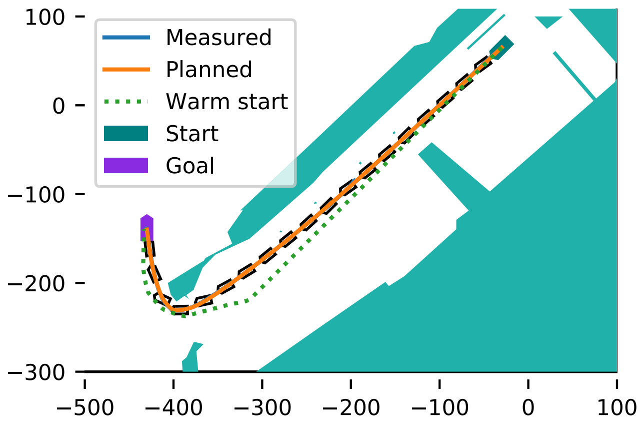


FIGURE 18. Measured and planned trajectories from validation experiment in Section VI.

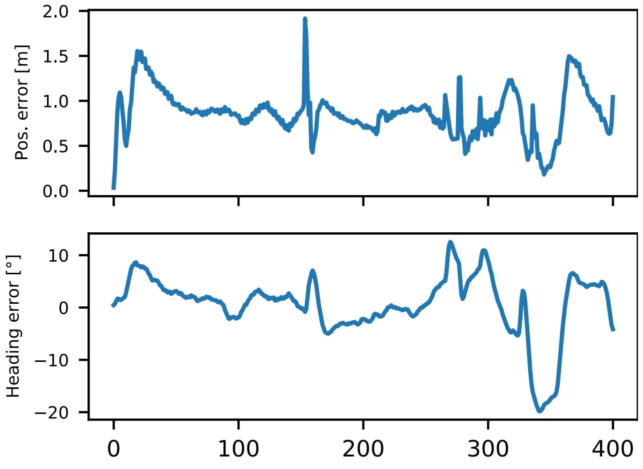


FIGURE 19. Tracking errors from the validation experiment in Section VI.

Figure 18 shows an overview of the scenario, including the measured and planned trajectories, and the warm start. The planned trajectory was naturally equivalent to the one in Section V-C, as nothing in the scenario was changed.

Figure 19 shows tracking errors for the tests. The positional error stayed within 1.5 m, with an exception at 160 s, which stems from a jump in the GNSS measurement experienced during the experiment. The positional tracking error was large at the beginning of the experiment, where *milliAmpere* and its control system struggled to follow the trajectory's acceleration. The unmodeled thruster dynamics and control systems can explain this initial lag. Large positional errors were also induced during the turn near the end of the experiments, which can be explained by the control system's poor tracking performance. Heading errors also occurred during the beginning of the experiment, as well as during the turn. These errors can be attributed to the coupling between linear and angular velocity, which is unaccounted for in the control system. *milliAmpere*'s physical properties, with its shallow and flat hull, make it hard to control its heading.

Figure 20 shows the distances from the measured and planned positions to the nearest obstacle. Of course,

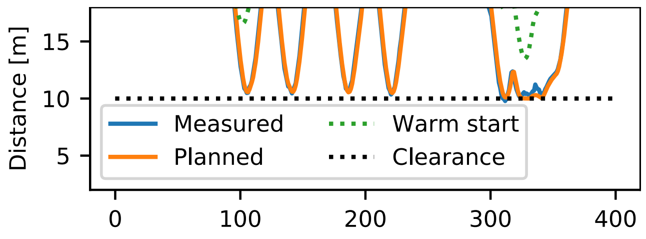


FIGURE 20. Measured and planned obstacle distances from the validation experiments in Section VI.

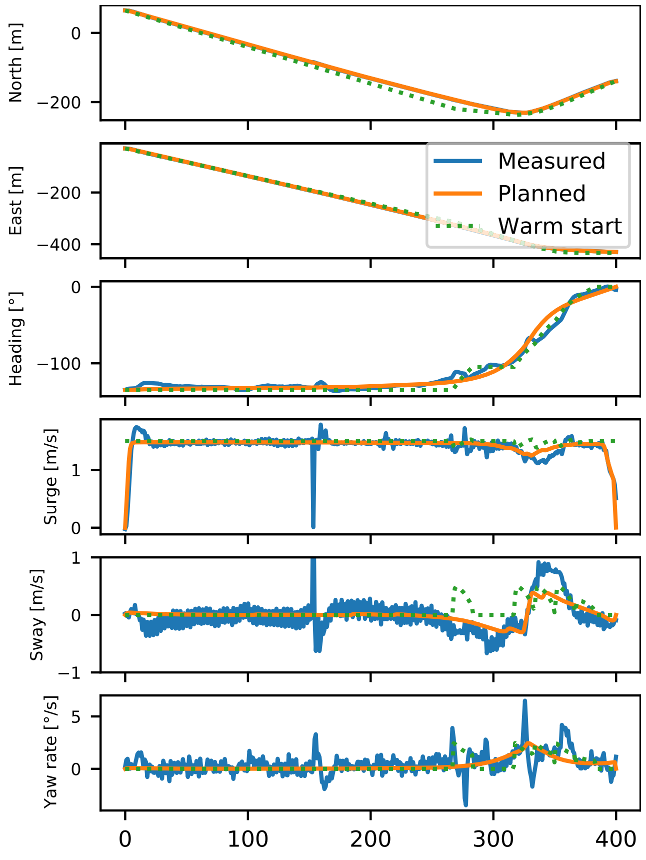


FIGURE 21. Measured and planned state trajectories from the validation experiment in Section VI.

the planned distance is more than the clearance of $r_c = 10$ m away from obstacles at all times. The measured distance was 9.9 m from the closest obstacle at the nearest, which we consider safe.

Figure 21 shows the measured and planned state trajectories. The referenced jump in GNSS measurement at 160 s is clearly visible in this plot, where the error propagates into the velocity estimates. While the plots show jumps in velocity, no such jump was experienced during the experiment – this is a measurement error. The sway velocity and yaw rate measurements oscillate, making hard to determine tracking error for these states. This oscillation may be due to the ship's natural frequency – it is not filtered out in the onboard navigation system, but we can see that the measurements lie around the planned trajectories.

During the experiment, the measured energy use was 245 W h, which is almost five times the simulated energy estimate. This discrepancy is due to the completely different approaches used to estimate the energy in simulation and experiments.

VII. CONCLUSION

Solving the continuous optimal control problem for trajectory planning is difficult and requires an initial guess close to the globally optimal solution to be a feasible option. Moreover, since the trajectory planning problem is inherently nonconvex, some clever encoding of obstacles is needed to reduce complexity, especially when dealing with polyhedral shapes with discontinuous gradients. We propose a continuous, model-based method for energy-optimized trajectory planning for ASVs that leverages a discrete search's desirable advantages to generate a good initial guess and performs convex encoding of obstacles to achieve collision avoidance. Our method is based on continuous optimal control, and the warm starting is provided by the hybrid A* algorithm. We have compared the method with an optimal control-based complete cell decomposition method with a similar cost function to find comparable performance in terms of optimality and significantly improved computational time. A comparison with a warm-started optimal control-based method from earlier work by us has shown improved performance, in addition to being able to use more general obstacle representations.

There are several areas where we can improve our method in further work:

- The search space can be extended to include surge velocity. This extension would allow the hybrid A* search to look for variations in the speed profile that could benefit energy efficiency.
- Including velocity in the search space will require modifications to the heuristic functions. Additionally, to prevent the search from always choosing the slowest velocity which would be energy optimal, we need to limit the trajectory's maximum duration. This constraint could be introduced by computing a map with the shortest path, similar to the holonomic with obstacles heuristic, and constrain the search from selecting nodes that cannot reach the goal within the time limit with the highest velocity. The fast marching method is appropriate for this distance map.
- The hybrid A* search is currently a naive Python implementation and contributes significantly to computation time. Improvements to this implementation would increase the performance of our method. This issue is also the case for the construction of the NLP.
- In our work, we have included external disturbances in terms of wind velocity. We have found that it does not affect the optimized trajectory or energy consumption in a significant manner. Other environmental effects, such as waves or ocean currents, can have more of an impact on energy consumption and should be explored.

- Including the COLREGs in trajectory planning for marine vessels should also be a priority to further work on this topic.

ACKNOWLEDGMENT

The authors thank Emil H. Thyri at NTNU for his help during the experiments with *milliAmpere*. Glenn Bitar also thanks Marius Thoresen at the Norwegian Defence Research Establishment for helpful discussions during algorithm implementation.

REFERENCES

[1] M. Ludvigsen and A. J. Sørensen, ''Towards integrated autonomous underwater operations for ocean mapping and monitoring,'' *Annu. Rev. Control*, vol. 42, pp. 145–157, Dec. 2016.

[2] B.-O.-H. Eriksen, G. Bitar, M. Breivik, and A. M. Lekkas, ''Hybrid collision avoidance for ASVs compliant with COLREGs rules 8 and 13–17,'' *Frontiers Robot.*, vol. 7, pp. 1–11, Feb. 2020.

[3] M. L. Seto, *Marine Robot Autonomy*. New York, NY, USA: Springer-Verlag, 2013.

[4] S. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. Eng, D. Rus, and M. Ang, ''Perception, planning, control, and coordination for autonomous vehicles,'' *Machines*, vol. 5, no. 1, p. 6, Feb. 2017.

[5] S. M. LaValle, ''Motion planning. Part I: The essentials,'' *IEEE Robot. Autom. Mag.*, vol. 18, no. 1, pp. 79–89, 2011. [Online]. Available: https://ieeexplore.ieee.org/document/5751929, doi: 10.1109/MRA.2011.940276.

[6] A. Wolek and C. A. Woolsey, ''Model-based path planning,'' in *Sensing and Control for Autonomous Vehicles*, T. I. Fossen, K. Y. Pettersen, and H. Nijmeijer, Eds. Cham, Switzerland: Springer, 2017, pp. 183–206.

[7] P. Hart, N. Nilsson, and B. Raphael, ''A formal basis for the heuristic determination of minimum cost paths,'' *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, 1968.

[8] M. Candeloro, A. M. Lekkas, and A. J. Sørensen, ''A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels,'' *Control Eng. Pract.*, vol. 61, pp. 41–54, Apr. 2017.

[9] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, ''Probabilistic roadmaps for path planning in high-dimensional configuration spaces,'' *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Dec. 1996.

[10] S. M. LaValle, ''Rapidly-exploring random trees: A new tool for path plannning,'' Iowa State Univ., Ames, IA, USA, Tech. Rep., 1998.

[11] R. Bellman, *Dynamic Programming*. New York, NY, USA: Courier Dover, 2003.

[12] Q. Gong, L. R. Lewis, and I. M. Ross, ''Pseudospectral motion planning for autonomous vehicles,'' *J. Guid., Control, Dyn.*, vol. 32, no. 3, pp. 1039–1045, May 2009.

[13] G. Bitar, M. Breivik, and A. M. Lekkas, ''Energy-optimized path planning for autonomous ferries,'' in *Proc. 11th IFAC CAMS*, 2018, pp. 389–394.

[14] G. Bitar, V. N. Vestad, A. M. Lekkas, and M. Breivik, ''Warm-started optimized trajectory planning for ASVs,'' in *Proc. IFAC Conf. Control Appl. Mar. Syst., Robot. Veh. (IFAC CAMS)*, 2019, vol. 52, no. 21, pp. 308–314.

[15] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, ''Autonomous parking using optimization-based collision avoidance,'' in *Proc. IEEE Conf. Decision Control (CDC)*, Dec. 2018, pp. 4327–4332.

[16] K. Bergman, O. Ljungqvist, J. Linder, and D. Axehill, ''An optimization-based motion planner for autonomous maneuvering of marine vessels in complex environments,'' 2020, *arXiv:2005.02674*. [Online]. Available: https://arxiv.org/abs/2005.02674

[17] J. Chen, W. Zhan, and M. Tomizuka, ''Autonomous driving motion planning with constrained iterative LQR,'' *IEEE Trans. Intell. Veh.*, vol. 4, no. 2, pp. 244–254, Jun. 2019, doi: 10.1109/TIV.2019.2904385.

[18] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, ''Practical search techniques in path planning for autonomous driving,'' Amer. Assoc. Artif. Intell., Tech. Rep. 1001.48105, 2008.

[19] Y. Zhang, H. Chen, S. L. Waslander, J. Gong, G. Xiong, T. Yang, and K. Liu, ''Hybrid trajectory planning for autonomous driving in highly constrained environments,'' *IEEE Access*, vol. 6, pp. 32800–32819, 2018.

[20] Y. Meng, Y. Wu, Q. Gu, and L. Liu, ''A decoupled trajectory planning framework based on the integration of lattice searching and convex optimization,'' *IEEE Access*, vol. 7, pp. 130530–130551, 2019.

[21] G. Casalino, A. Turetta, and E. Simetti, ''A three-layered architecture for real time path planning and obstacle avoidance for surveillance USVs operating in harbour fields,'' in *Proc. Oceans-Europe*, May 2009, pp. 1–8.

[22] P. Svec, B. C. Shah, I. R. Bertaska, J. Alvarez, A. J. Sinisterra, K. von Ellenrieder, M. Dhanak, and S. K. Gupta, ''Dynamics-aware target following for an autonomous surface vehicle operating under COLREGs in civilian traffic,'' in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 3871–3878.

[23] A. B. Martinsen, A. M. Lekkas, and S. Gros, ''Optimal model-based trajectory planning with static polygonal constraints,'' 2020, *arXiv:2010.14428*. [Online]. Available: https://arxiv.org/abs/2010.14428

[24] T. I. Fossen, *Handbook Mar. Craft Hydrodynamics Motion Control*. Hoboken, NJ, USA: Wiley, 2011.

[25] A. A. Pedersen, ''Optimization based system identification for the milliAmpere ferry,'' M.S. thesis, Norwegian Univ. Sci. Technol., Trondheim, Norway, 2019. [Online]. Available: http://hdl.handle.net/11250/2625699

[26] A. Wächter and L. T. Biegler, ''On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,'' *Math. Program.*, vol. 106, no. 1, pp. 25–57, May 2005, doi: 10.1007/s10107-004-0559-y.

[27] T. Lee, H. Chung, and H. Myung, ''Multi-resolution path planning for marine surface vehicle considering environmental effects,'' in *Proc. OCEANS IEEE*, Jun. 2011, pp. 1–9.

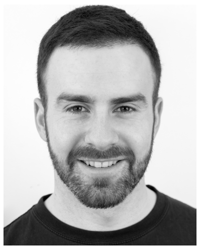
[28] J. A. Sethian, ''A fast marching level set method for monotonically advancing fronts,'' *Proc. Nat. Acad. Sci. USA*, vol. 93, no. 4, pp. 1591–1595, 1996.

[29] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, and D. Lane, ''Path planning for autonomous underwater vehicles,'' *IEEE Trans. Robot.*, vol. 23, no. 2, pp. 331–341, Apr. 2007.

[30] S. Garrido, L. Moreno, F. Martín, and D. Álvarez, ''Fast marching subjected to a vector field–path planning method for mars rovers,'' *Expert Syst. Appl.*, vol. 78, pp. 334–346, Jul. 2017.

[31] G. Bitar, A. B. Martinsen, A. M. Lekkas, and M. Breivik, ''Trajectory planning and control for automatic docking of ASVs with full-scale experiments,'' in *Proc. 1st Virtual IFAC World Congress*, 2020, pp. 1–5.

[32] T. R. Torben, A. H. Brodtkorb, and A. J. Sørensen, ''Control allocation for double-ended ferries with full-scale experimental results,'' in *Proc. 12th IFAC Conf. Control Appl. Mar. Syst., Robot. Veh. (IFAC CAMS)*, 2019, pp. 556–563.



GLENN BITAR received the B.S. degree in computer science and industrial automation from the Telemark University College, Porsgrunn, Norway, in 2015, and the M.S. degree in cybernetics and robotics from the Department of Engineering Cybernetics, NTNU, Trondheim, Norway, in 2017, where he is currently pursuing the Ph.D. degree.

His research interests include energy-optimized autonomous motion control for ships, automatic docking, and optimization-based path and trajectory planning.

ANDREAS B. MARTINSEN received the M.S. degree in engineering cybernetics from NTNU, Trondheim, Norway, in 2018, where he is currently pursuing the Ph.D. degree in engineering cybernetics.

His research interests include reinforcement learning, optimal control, and machine learning, with a focus on marine and maritime applications.

ANASTASIOS M. LEKKAS (Member, IEEE) received the M.S. degree in mechanical and aeronautical engineering from the University of Patras, Greece, in 2008, and the Ph.D. degree in engineering cybernetics from NTNU, Trondheim, Norway, in 2014.

From 2015 to 2017, he has worked as a Science Officer at JPI OCEANS, Brussels, Belgium. He is currently an Associate Professor of autonomous systems with the Department of Engineering Cybernetics, NTNU, where he is also affiliated with the Centre for Autonomous Marine Operations and Systems. He participates in projects focusing on the autonomy of marine vehicles (Autoferry, SEAVENTION) and is the Project Manager of the EXAIGON Project, where the main goal is to develop explainable AI methods for safety- and business-critical applications. His research interest includes merging artificial intelligence with control engineering in order to develop cyber-physical systems with increased autonomy.

MORTEN BREIVIK (Member, IEEE) received the M.S. and Ph.D. degrees in engineering cybernetics from the Department of Engineering Cybernetics, NTNU, Trondheim, Norway, in 2003 and 2010, respectively.

He is currently the Head of the Department of Engineering Cybernetics, NTNU. He is also an Associate Researcher with the Centre for Autonomous Marine Operations and Systems. He has previously worked as an Assistant Professor and a Researcher with NTNU, a Scientific Advisor for Maritime Robotics, and a Principal Engineer and the Department Manager in applied cybernetics at Kongsberg Maritime. His research interests include nonlinear and adaptive motion control of unmanned vehicles in general and in autonomous ships in particular. He is also a member of the Norwegian Board of Technology.