2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Real-Time Navigation for Autonomous Surface Vehicles In Ice-Covered Waters

Rodrigue de Schaetzen[1], Alexander Botros[1], Robert Gash[2], Kevin Murrant[2], Stephen L. Smith[1]

***Abstract*— Vessel transit in ice-covered waters poses unique challenges in safe and efficient motion planning. When the concentration of ice is high, it may not be possible to find collision-free trajectories. Instead, ice can be pushed out of the way if it is small or if contact occurs near the edge of the ice. In this work, we propose a real-time navigation framework that minimizes collisions with ice and distance travelled by the vessel. We exploit a lattice-based planner with a cost that captures the ship interaction with ice. To address the dynamic nature of the environment, we plan motion in a receding horizon manner based on updated vessel and ice state information. Further, we present a novel planning heuristic for evaluating the cost-to-go, which is applicable to navigation in a channel without a fixed goal location. The performance of our planner is evaluated across several levels of ice concentration both in simulated and in real-world experiments.**

I. INTRODUCTION

Recent successes in deploying autonomous ship navigation systems demonstrate that ship autonomy can improve marine safety and travel efficiency [1]. These benefits are especially appealing to crews tasked with high-risk missions such as arctic navigation through ice-covered waters [2], [3]. This setting typically features a significantly higher concentration of obstacles than most autonomous navigation applications [4]. In addition, the obstacles are dynamic in nature, moving in response to actions taken by the ship and contributing to the overall complexity of the problem.

In this paper, we address the problem of planning motion for an autonomous surface vehicle (ASV) operating in icy waters. We assume that the ASV is built for arctic or sub-arctic conditions but has limited ice-breaking capabilities. Depending on the ice conditions, these ASVs may need to be escorted by an icebreaker which creates a narrow channel containing a high concentration of ice of widely different sizes (e.g., navigation through the St. Lawrence Seaway during winter). Our proposed framework operates by planning a reference path to be tracked by the ASV, and then replanning in a receding horizon fashion using updated ice information.

Existing work on path planning for ASVs predominantly focuses on planning collision-free paths [1], [5]–[8]. For the conditions described above — and illustrated in Figure 1 — such paths may not exist. Thus, rather than avoiding ice, the objective should be to minimize a cost function that captures the effect of a collision with an ice piece (for example, the energy transferred during collision). In this work we propose a planning framework that utilizes such a cost function.

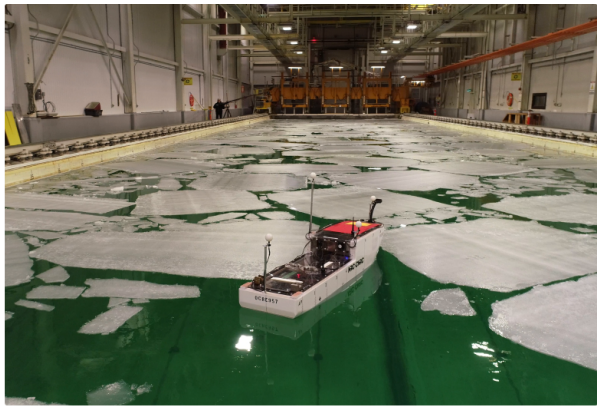

Fig. 1. Model vessel being tested in the 12 m $\times$ 76 m National Research Council Canada (NRC) ice tank in St. John's, Newfoundland and Labrador.

Contributions: For ships operating in an ice field, we propose a local path planner which we incorporate into a real-time navigation system. The underlying methodology we adopt is the familiar A* path planner over a graph of motion primitives [9]. However, to apply this methodology to ship navigation in ice, several key challenges are addressed. We adapt an existing collision model for ship-ice interactions [10] to show that a cost function modelled on *kinetic energy* efficiently penalizes head-on collisions with large ice pieces. Further, since the goal of our planner is forward progress through an ice channel, we propose an admissible closed-form heuristic where the objective is to reach a line segment as opposed to a single configuration. Finally, we demonstrate the efficacy of our approach in ice fields through experiments — both in simulation and in a physical 12 m $\times$ 76 m ice tank.

Related Work: Navigation among movable obstacles (NAMO) is considered in [11], [12]. The authors address the problem of navigating a robot through an environment where obstacles can be manipulated by the robot. The NAMO problem is similar to the problem of navigating an ASV through an ice field in that obstacles are movable in both settings. However, unlike ASV navigation the obstacles considered in the NAMO problem typically do not interact with each other once manipulated. Of perhaps greater consequence, obstacles inflict no damage to the robot.

Extensive work has been done in path planning, tracking

This work is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the National Research Council Canada (NRC).

[1]Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: {rdeschae, alexander.botros, stephen.smith}@uwaterloo.ca)

[2]National Research Council Canada (e-mail: {robert.gash, kevin.murrant}@nrc-cnrc.gc.ca)

control, and collision avoidance algorithms for ASVs. For many of these works, the obstacles considered are either other vessels or static surrounding environments [5], [13], [14] like harbors [1] and urban waterways [6]. The approaches proposed in these works are effective in their settings, but assume that collisions are forbidden. Therefore, they are not easily generalized to path planning in highly congested environments where collisions are unavoidable.

In path planning for ships, related to our work, the authors in [7] use aggregated statistics of the ice conditions retrieved from satellite imaging for their uncertainty-based route planner. Their work plans paths – consisting of a set of waypoints – on the scale of several thousand kilometers which require a local planner (e.g., the one presented in this paper) for real-time autonomous navigation. Local path planning is considered in [15] where the authors propose a motion planner using bidirectional RRT and data gathered from marine radar imaging. Similarly, in [16] the authors construct a graph using a morphological skeleton from a post processed overhead snapshot of an ice field. They then employ A* to compute a path in the resulting graph. The techniques proposed in [15], [16] empirically work well in low-concentration ice fields. However, planned paths are piece-wise straight lines and do not take into account the ASVs' dynamics. Further, the authors assume that computed paths can and must be collision-free. Minimal turning radius constraints are considered in [8], but the authors still consider collision-free paths.

II. Problem Formulation

The problem we address is navigating a ship through a cluttered ice channel (as in Figure 2(a)) while minimizing the effects of ship-ice collisions on the ship—namely the energy transferred in the collisions. We treat the water surface on which a ship moves as a 2D surface $\mathcal{W} \subseteq \mathbb{R}^2$.

Our objective is forward progress along a channel $\mathcal{C} \subseteq \mathcal{W}$ which we model as a rectangle with length parallel to the y-axis as in 2(a). This model does not limit the applicability of our method since long, curved channels can be partitioned into smaller rectangles as in Figure 2(b). The objective is to reach a goal line segment $\mathcal{G} \subset \mathcal{C}$ with constant y-value as illustrated in Figure 2(a). Given m ice floes in $\mathcal{C}$, we treat each floe as an obstacle $O_i \subset \mathcal{W},\ i = 1, \ldots, m$ which we group into a set of obstacles $\mathcal{W}_{\text{obs}} = \{O_1, \ldots, O_m\}$. We assume that the location of each obstacle in $\mathcal{W}_{\text{obs}}$ can be estimated at any time via an onboard vision system.

To navigate a ship through an ice field, three components are required: a *reference path* $\pi(s)$ parameterized by arc-length s along the path, a *velocity profile* v that describes how the path is traversed through time, and a *controller* for tracking the path at the desired velocity. The tuple (π, v) is called a *trajectory*. To capture the effects of ice collision, we utilize a cost function u that penalizes the path length (final arc length) s_f and a *collision cost*:

$$u(\pi, v) = s_f + \underbrace{\alpha \int_0^{s_f} c_{\text{obs}}\big(\pi(s), \mathcal{W}_{\text{obs}}, v(s)\big)\, ds}_{\text{collision cost}}. \quad (1)$$

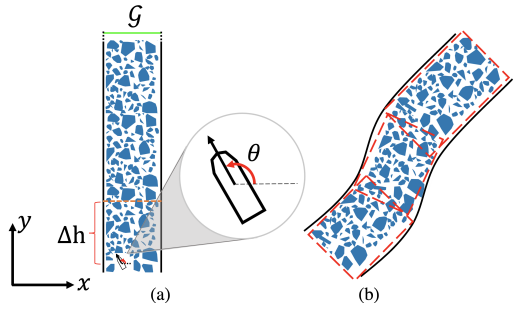

Fig. 2. (a) Depiction of the navigation problem of interest for a rectangular channel with ice (blue), ASV (black), and goal region (green line). (b) Generalization of problem to curved channel.

Where $c_{\text{obs}} \geq 0$ is described in detail in Section III and $\alpha \geq 0$ is a tuning parameter. Thus, we seek to solve the following problem.

Problem 1. *Given a start position of the ship, a goal line $\mathcal{G}$, and obstacles $\mathcal{W}_{obs}$, compute a trajectory (π, v) from the start to $\mathcal{G}$ that minimizes u.*

In what follows we propose a solution to this problem and describe how it is incorporated into a navigation system for real-time (re-)planning.

III. Navigation Framework

In this section, we discuss our navigation framework and approach to solving Problem 1. To account for the large and evolving environment, we frequently re-plan a reference path and velocity profile for a controller in real-time. Replanning is done over a moving horizon as illustrated by the part of the channel lying below the orange dotted line in Figure 2(a).

Algorithm 1 Receding Horizon Navigation Framework

Input: $\mathcal{G}, \Delta h, \Delta t$

1: **while** `True` **do**
2: $\quad \mathcal{W}_{\text{obs}} \leftarrow$ obstacles from onboard sensors
3: $\quad P, v_S \leftarrow$ current ship pose and speed
4: $\quad$ Set intermediate goal $\mathcal{G}^i$ a distance Δh ahead of P
5: $\quad$ **if** ship pose P has not crossed $\mathcal{G}$ **then**
6: $\quad\quad \pi \leftarrow$ path from P to $\mathcal{G}^i$ with *static* $\mathcal{W}_{\text{obs}}$, v_S
7: $\quad\quad v_{\text{nom}} \leftarrow$ velocity profile for π
8: $\quad\quad$ Send (π, v_{nom}) to controller to track for time Δt
9: $\quad$ **else**
10: $\quad\quad$ Exit loop

The navigation framework is summarized in Algorithm 1, which takes as input a final goal line segment $\mathcal{G}$, receding horizon parameter Δh, and control tracking parameter Δt. At the start of each iteration we obtain updated ice data $\mathcal{W}_{\text{obs}}$ from onboard sensors (Line 2) along with the ship's current position in $\mathbb{R}^2$ and orientation (called a *pose* P) and speed (Line 3). We compute an intermediate goal line $\mathcal{G}^i$ a distance Δh ahead of the ship (Line 4; see Figure 2(a)). If the ship has not yet reached the final goal $\mathcal{G}$ (Line 5), we run our proposed planning algorithm (see Section III-A), returning a path from P to $\mathcal{G}^i$ (Line 6). A nominal

velocity profile is then computed for the path (Line 7). In our planner, we use a constant nominal speed based on the concentration of ice, following established guidelines such as [2]. The planned trajectory (π, v_{nom}) is sent to a controller (Line 8) which tracks it for a time Δt before the process repeats. We employ a Dynamic Positioning (DP) controller described in Section IV. The following sections describe how paths π are computed.

Note that at each iteration, ice is treated as static when computing a trajectory (π, v_{nom}) but is updated frequently to account for collisions and ice movement (Line 2). This approach maintains low planning run-time for each iteration while still accounting for complex ice movement across all iterations (this is explored in the evaluation in Section V).

A. Path Planning using a State Lattice

To solve Problem 1, we use a form of lattice planner [9], where paths are generated using a finite set of motion primitives.

1) Model: For the purposes of path planning, we treat the ship as a 2D rigid body with three degrees of freedom [17]: planar position $(x, y) \in \mathbb{R}^2$ and heading $\theta \in [0, 2\pi)$. We refer the tuple (x, y, θ) as a *configuration*. We adopt a unicycle model commonly used in marine navigation [18]–[20]: $\dot{x}(s) = \cos(\theta), \dot{y}(s) = \sin(\theta), \dot{\theta}(s) = \kappa,\ |\kappa| \leq \kappa_{\max}$, where derivatives $(\dot{})$ are taken with respect to arc length s, κ is the *curvature*, and $\kappa_{\max}$ is a maximum curvature dictated by the physical limits of the ship. A path $\pi(s)$ is *feasible* if it adheres to the unicycle model above.

2) Lattice planning: In lattice-based path planning, the set of all configurations is discretized into a regularly repeating grid called a *lattice* L. Configurations in the lattice are called *vertices*. A set (called a *control set*) of feasible paths between lattice vertices (called *motion primitives*) is pre-computed offline and used as an action set during an online search. The key observation in lattice-based path planning is that motion primitives may be rotated, translated, and concatenated to form complex paths (observe that motion primitives *are* paths between lattice vertices).

For a control set $\mathcal{P}$, we can construct a graph $G^{\mathcal{P}} = (L, E, \tilde{u})$ the vertices of which are the lattice vertices L while edges are pairs of vertices (p^1, p^2) such that there exists a motion primitive $\pi \in \mathcal{P}$ that can be rotated and translated so that $\pi(0) = p^1, \pi(s_f) = p^2$. Given the ship's current speed v_S (Line 3 of Algorithm 1) and $\mathcal{W}_{\text{obs}}$ (Line 2), the cost of an edge (p^1, p^2) with associated motion primitive π is given by $\tilde{u}((p^1, p^2)) = u(\pi, v_S)$ from (1). The problem of computing a feasible path between lattice vertices reduces to computing a cost-minimizing path in $G^{\mathcal{P}}$.

We generate a state lattice L by discretizing the plane $\mathbb{R}^2$ into a uniform grid and the headings θ into uniformly spaced angles around the unit circle. To define our motion primitives, we compute shortest paths between (x, y, θ) configurations for the unicycle model, known as Dubin's paths [21]. These paths are comprised of sequences of straight lines and circular arcs of radius $r_{\min} = \kappa_{\max}^{-1}$. The control set $\mathcal{P}$ is generated using the method proposed in [22] which computes a control set that balances a tradeoff between $|\mathcal{P}|$ and path optimality.

We use the lattice framework to plan a path π (Line 6) from the ship's current pose P (Line 3) to the current goal $\mathcal{G}^i$ (Line 4) with the lattice origin aligned with P. This is accomplished by searching over the graph $G^{\mathcal{P}}$ using the A* search algorithm [23] with a line segment as the objective as opposed to a single configuration. As a final step in Line 6, we run a smoothing algorithm on π introduced in [22] as a post-processing step to remove excessive oscillations. In Section (1) we finalize the cost function u from (1) while in Section III-C we present a heuristic to improve the performance of A*.

B. Cost Function

We describe the cost function u from (1) given a feasible path π, obstacles $\mathcal{W}_{\text{obs}}$, and the ship's current speed v_S. In detail, we use the notion of *kinetic energy* to derive a function $c_{\text{obs}}(\pi(s), \mathcal{W}_{\text{obs}}, v_S)$ that captures the severity of ship-ice collisions in terms of the energy transferred during collision.

We restrict our attention to convex, polygon-shaped ice and assume uniform ice density and thickness in the individual ice floes [10], [24]. Ship-ice collisions are modeled and empirically validated in [10], and we adopt a simplified version of their 2D inelastic-collision method to model ship-ice collisions. Specifically, we treat the ship and ice as disks rather than polygons and the ice as static prior to collision. This simplified model lets us efficiently capture sufficient detail in our collision cost and similar approximations have been made in existing work [25], [26].

With this collision model, the change in kinetic energy ΔK_{sys} in the system is given by

$$\Delta K_{\text{sys}} = \frac{1}{2} M_{\text{eq}} V_{\text{eq}}^2, \tag{2}$$

where M_{eq} and V_{eq} are the effective inertial mass and velocity, respectively, at the moment of collision [10]. For the case of two disk-shaped bodies, the effective mass M_{eq} is constant and is defined as:

$$M_{\text{eq}} = \frac{m_S m_I}{m_S + m_I}, \tag{3}$$

where m_S, m_I are the ship and ice masses, respectively. We assume that m_S is known and m_I can be calculated as the product of the area, thickness, and density [10]. The effective velocity V_{eq} is given by [10]:

$$V_{\text{eq}} = v_S \cos(\theta), \tag{4}$$

where v_S is measured in Line 3 of Algorithm 1 and θ is the angle between the ship's heading and the collision normal $\vec{n}$ (Figure 3). Next, we isolate the change in kinetic energy of the ship ΔK_S.

The goal of our collision cost will be to minimize the ship's kinetic energy loss from ship-ice collisions. Since the ice is initially static, it must hold that $\Delta K_I > 0$ where ΔK_I is the change in kinetic energy of the ice. Further,

$$\Delta K_{\text{sys}} = \Delta K_I - \Delta K_S. \tag{5}$$

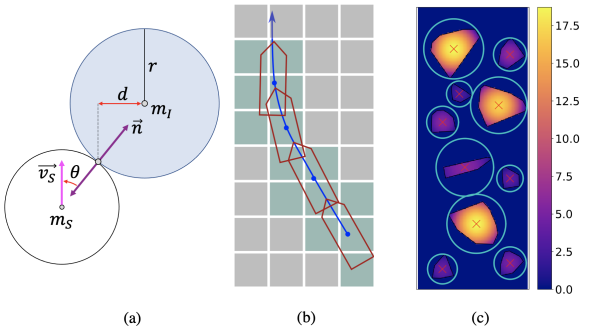

Fig. 3. (a) Collision scenario between two disk-shaped bodies. (b) Example path (blue) with ship footprint (red) at regular intervals of arc length with exaggerated costmap resolution to illustrate the swath (green). (c) Sample costmap where the color bar indicates the cost. Obstacle centroids are in red and bounding circles are in light blue.

We treat the ice as having a velocity post-collision equal to V_{eq} – a reasonable approximation if m_S/m_I is large. Therefore, $\Delta K_I = 0.5 m_I V_{\text{eq}}^2 = 0.5 m_I (v_S \cos(\theta))^2$ and from (2), (5):

$$|\Delta K_S| = \frac{m_I^2}{2(m_S + m_I)} \left(v_S \cos(\theta)\right)^2. \tag{6}$$

From Figure 3(a), observe that $\sin(\theta) = d/r$ where d is the lateral distance between the collision point and the center of the ice and r is the radius of the ice. Therefore from (6), we may write $|\Delta K_S|$ as a function of d, r, m_I, v_S:

$$\Delta K_S(d, r, m_I, v_S) = \frac{1}{2} \frac{m_I^2}{m_S + m_I} \left[v_S \cos\left(\arcsin\left(\frac{d}{r} \right) \right) \right]^2$$
$$= \frac{v_S^2 m_I^2}{2(m_S + m_I)} \left(\frac{r^2 - d^2}{r^2} \right), d \in [0, r]. \tag{7}$$

Observe that the function ΔK_S is large if collisions are head-on ($\theta = d = 0$) or if m_I/m_S is large. Thus ΔK_S has the benefit of effectively penalizing collisions with ice that are head-on and or involve large ice relative to the ship. Using ΔK_S, we describe the cost function u from (1).

To efficiently compute collision costs across an ice channel $\mathcal{C}$, we use a costmap representation of the planar environment as done in [9]. In particular, the channel $\mathcal{C} \subseteq \mathbb{R}^2$ is discretized into a square grid where each grid cell is assigned an identifying tuple $k \in \mathbb{N}^2$ corresponding to the cells' center and a cost $c_{obs}(k, \mathcal{W}_{\text{obs}}, v_S)$ given obstacles $\mathcal{W}_{\text{obs}}$ (determined in Line 2 of Algorithm 1), and v_S (Line 3). Each cell is occupied by at most one obstacle. The set of costmap cells that are occupied from the area of a ship as it traverses a path π is called the *swath* of π and the area occupied by the ship is called the *footprint* [9] (see Figure 3(b)).

For our planning task, the costmap is effectively a lookup table for computing the collision cost associated with a particular path between a pair of lattice vertices, given an appropriate mapping from the configuration space to the costmap. For each polygonal obstacle $O_j \in \mathcal{W}_{\text{obs}}$, let C_j denote the position of the centroid of O_j, r_j the radius of its bounding circle centered at C_j (Figure 3(c)), and m_j its mass. The function $c_{obs}(k, \mathcal{W}_{\text{obs}}, v_S)$ for a cell k and current ship speed v_S uses the ship kinetic energy loss from (7):

$$c_{\text{obs}}(k, \mathcal{W}_{\text{obs}}, v_S) = \begin{cases} \Delta K_S(q, r_j, m_j, v_S), & k \cap O_j \neq \emptyset \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

where $q = ||k - C_j||$, the Euclidean distance from the cell to the centroid of the obstacle. This function is well defined since each cell can be occupied by at most one obstacle. A sample costmap is illustrated in Figure 3(c).

Finally, with $c_{\text{obs}}(k, \mathcal{W}_{\text{obs}}, v_S)$ given in (8) we define the cost $u(\pi, v_S)$ for a candidate path π (and ships current speed v_S) with final arc-length s_f using the discretization of (1):

$$u(\pi, v_S) = s_f + \alpha \sum_{k \in \text{swath of } \pi} c_{\text{obs}}(k, \mathcal{W}_{\text{obs}}, v_S). \tag{9}$$

C. Heuristic

Given the current ship pose P (Line 3), we present an admissible heuristic with a closed form to improve the runtime of the path planning step in Line 6. We omit the final expression for brevity, but details can be found in the extended version [27].

At a high-level, we compute the shortest path from P to an infinite line $\mathcal{G}_\infty$ that is colinear with the intermediate goal line $\mathcal{G}^i$ (Line 4), subject to the unicycle model described in Section III-A.1. This heuristic is admissible to our graph search implementation since the cost function u is lower bounded by path length and $\mathcal{G}^i \subset \mathcal{G}_\infty$ (thus the shortest path to $\mathcal{G}_\infty$ is no longer than the shortest path to $\mathcal{G}^i$). To characterize the proposed heuristic, we offer the following result:

Proposition III.1 (Closed-form Heuristic)**.** *The shortest path from P to the infinite line $\mathcal{G}_\infty$ with minimum turning radius $r_{\min}$ is a Dubin's path. Referencing Figure 4,*

1) *if $\mathcal{G}_\infty$ lies above the point o, the path is of the form CS (circular arc C of radius $r_{\min}$, followed by a straight line S) and S intersects $\mathcal{G}_\infty$ at a right angle (Fig. 4(a));*
2) *otherwise, the path is of the form C (Fig. 4(b)).*

In what follows we provide just a sketch of the proof. The complete proof is given in the extended version [**?**].

Proof. (Sketch of Proof) Let $P = (P_x, P_y, P_\theta)$, and let $P^g = (P_x^g, P_y^g)$ be any point on $\mathcal{G}_\infty$. Since the heading of P^g is not specified, the shortest path from P to P^g is of the form CS [28], [29] where S intersects $\mathcal{G}_\infty$ at an angle θ. Therefore, determining the shortest path from P to $\mathcal{G}_\infty$ reduces to computing $\min_\theta L(C) + L(S)$ where $L(\cdot)$ denotes length. It is easily verified that this is solved by $\theta = \pi/2$. If, however, $\mathcal{G}_\infty$ lies below o, then the result follows from [28], [29]. □

In the next section, we detail the integration of our proposed planner with the experimental platform.

IV. Details on Experimental Platform

We validated our proposed approach by integrating our navigation framework with the NRC ice tank facility shown

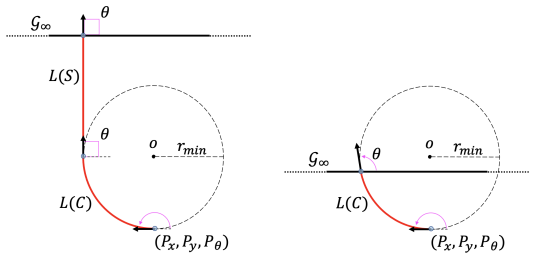

Fig. 4. Diagram depicting the geometry of the Dubins path to an infinite line $\mathcal{G}_\infty$ for two possible cases.

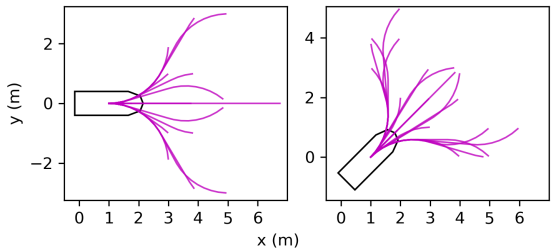

Fig. 5. Axis aligned (left) and non-axis aligned (right) motion primitives generated for a 1:45 scale ASV model.

in Figure 1, which is complete with a model vessel and real-time overhead vision system [16].

Environment and Physical Model: The NRC ice basin is 12 m $\times$ 76 m with ice thickness up to 200 mm. We used the same 1:45 scale platform supply vessel (PSV) model deployed in [4] shown in Figure 1.

Vision System: The facility contains 20 ceiling cameras sending ice information at a frequency of 1 Hz. The ASV configuration is computed at 50 Hz using the tracking system and an on-board inertial measurement unit (IMU) [4], [16]. Given the two update rates, we set $\Delta t = 1$ in Algorithm 1.

Controller: We employed a Dynamic Positioning (DP) controller – widely used in marine navigation [30], [31] – which generates thruster/propeller commands to regulate position and heading. We used a constant nominal velocity of 0.3 m/s and a minimum turning radius $r_{\min} = 2$ m, computed according to a full-scale vessel.

Planner Parameters: Discretization was set to 1×1 m for planar position and 8 equally spaced values for heading. In Figure 5 we show the two sets of different motion primitives generated for the axis-aligned and non-axis aligned directions which consist of 15 and 19 primitives respectively.

Costmap grid resolution was 0.25 m, finite horizon distance $\Delta h = 20$ m in Algorithm 1, and the cost function tuning parameter $\alpha = 10$ in (1).

V. Results

We demonstrate the efficacy of the proposed approach through simulation and physical experiments. Each trial consisted of navigating a ship across the length of an ice channel to the specified goal $\mathcal{G}$. Our simulations were done in Python and the 2D physics library Pymunk [32] was used as our backbone for the simulation experiments.

A. Simulation Setup

The simulation setup matches that of the experimental platform (e.g., NRC ice tank dimensions, controller, 1:45 vessel model). Given a specified target ice concentration (expressed as a value in [0, 1] where 1 is maximal ice cover), we populated the map with randomly generated non-overlapping polygons subject to the following constraints: $R_{\min} = 0.5$m, $R_{\max} = 2$m, $y_{\min} = 5$m, $y_{\max} = 70$m, where $R_{\min}, R_{\max}$ are constraints on the radii of the primal circles from which the polygons stem from while the y constraints enforce the polygon vertices to be within a specified region of the environment. We randomized the ship starting x position, and fixed the ship starting y position and heading to 2 m and $\pi/2$ respectively. The goal line segment $\mathcal{G}$ was set to 72 m in the y-axis.

B. Simulation Results

In simulation, we explored our framework's effectiveness as a function of ice concentration via comparisons to Algorithm 1 using other planning schemes (Line 6). Two baseline planning algorithms were considered which we refer to as *straight* and *skeleton*. The former is simply a planner that returns a constant straight path from the ship's current position to the goal $\mathcal{G}$ and the latter refers to the shortest open-water path routing approach described in [16] and [4]. Their approach constructs morphological skeletons based on the ice environment to generate paths. We refer to the three different versions of our navigation framework based on their planning scheme, i.e. *lattice*, *straight*, and *skeleton*. We performed the same set of trials across all three of these navigators. In total we ran 3 navigators $\times$ 50 trials $\times$ 4 ice concentrations $=$ 800 experiments. The ice concentrations considered were 0.2, 0.3, 0.4, and 0.5. Note, we used a 1st order Nomoto model [17] to describe our vessel dynamics in simulation.

1) Metrics: We computed a running total of the kinetic energy lost by the ship ΔK_S due to collisions with ice, using the physics simulator. To better interpret this metric, we used the ship total kinetic energy loss in the straight navigator to normalize the values for each trial for lattice and skeleton. Further, we captured the average tracking error across all simulations of the lattice and skeleton navigators to gauge how easy each reference path was to track. Finally, we logged the mass of the ice collided with for each collision that occurred in each simulation and obtained a probability density function for each navigator. This effectively approximates the probability of colliding with an ice floe of any size for each navigator.

2) Results: We present two figures that capture the main results from our simulations. Figure 6 illustrates the improved performance of our approach (lattice) with regards to the total kinetic energy lost by the ship from collisions with ice. Our mean and median are consistently lower across all 4 ice concentrations. For the trials done in the most dense ice fields (i.e. ice concentration $=$ 0.5) we achieved a mean of 0.39 vs. 0.43 for skeleton. In other words, our approach achieved a 9% decrease in terms of kinetic

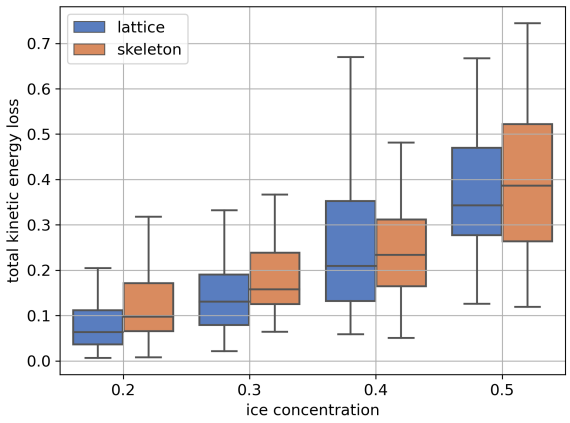

Fig. 6. Total kinetic energy loss by ship (normalized by straight navigator) as a function of the ice concentration for our navigation framework using lattice planning (ours) and skeleton.

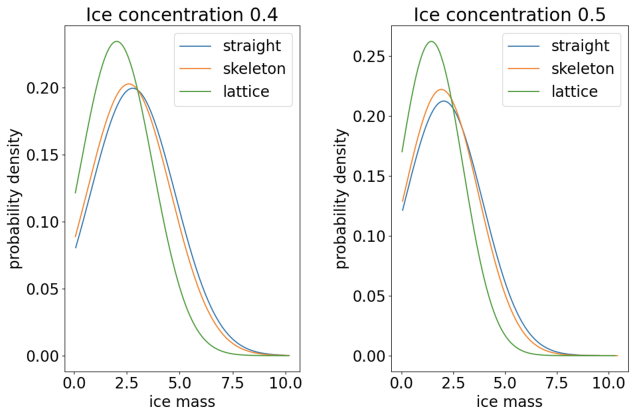

Fig. 7. Frequency of ice collision by mass for the three planning methods, smoothed using kernel density estimation. Note, ice mass is dimensionless here.

energy lost. In addition, our maximum (outliers not shown in plot) is significantly lower both in the 0.2 and the 0.5 ice concentrations, e.g., for 0.5 concentration our max was 1.0 and skeleton was 1.6.

In Figure 7, we show the probability density functions for collisions across different navigators [10]. In the 0.4 ice concentration scenario, the mean mass of ice colliding with the ship was 2.0 for the proposed navigator, 2.6 for the skeleton, and 2.8 for straight while in the 0.5 ice concentration scenario, 1.4 for the proposed, 1.9 for the skeleton, and 2 for the straight. Therefore, on average the proposed approach collided with ice that was 24% smaller in mass than the skeleton navigator, and 29% smaller than the straight.

Further, the tracking error for paths using the proposed approach was, on average 50% lower than that of the skeleton navigator. Note, path planning with our approach took an average of 90 ms with a max of 127 ms which are both comparable to the skeleton navigator. Finally, graph search using our heuristic from III-C expanded an average of 15% (max of 48%) fewer nodes than a euclidean distance baseline.

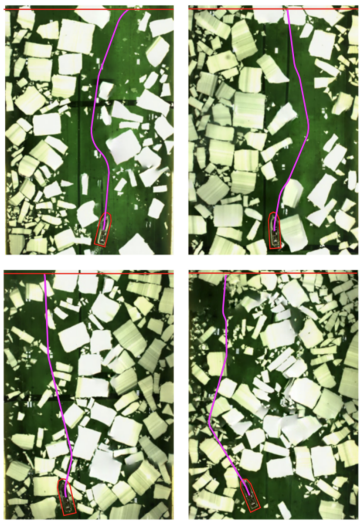

Fig. 8. Snapshots of four different trials done in the 12 m $\times$ 76 m ice tank at varying levels of ice concentration.

C. Real World Results

We conclude with an overview of the experiments ran with our planner in the physical NRC ice tank. We ran a total of 8 trials across two different ice concentrations (medium and high). This series of tests proved to be the first successful attempts at fully autonomous navigation across the entire length of the ice basin in the NRC ice tank facility following partial successes in preliminary testing done in [4].

We show four representative snapshots (Figure 8) taken during the experiments and make a series of observations that highlight our planner features. The advantage of having a non-distinct goal somewhere along a line segment is clearly shown across each snapshot. We also see the smoothness of the turn-constrained planned paths that result in better tracking behavior than any-angle approaches such as the skeleton planner. Most importantly, the collisions that occur along the paths are consistent with our design decisions captured by our proposed cost function (9). Namely, avoiding larger ice floes over smaller ones, and colliding with ice such that the ship loses minimal kinetic energy. These last two points are most apparent in the 2 snapshots taken from the high ice concentration experiments (bottom left and bottom right in Figure 8).

VI. Conclusions

In this work, we proposed an autonomous real-time navigation framework for ASVs through ice-covered waters. Our method tailored well-known lattice-based path planning and receding horizon-based planning to produce an effective navigation strategy for this environment. A key component of our framework is the proposed cost function, which captures the energy lost by the ship during a collision and heavily penalizes head-on collisions with larger ice floes. Our planner achieved better overall performance than existing planning solutions designed for navigation in icy waters. In future work, we intend to incorporate a predictive ice-motion model into our planner.

REFERENCES

[1] K. Bergman, O. Ljungqvist, J. Linder, and D. Axehill, "An optimization-based motion planner for autonomous maneuvering of marine vessels in complex environments," in *IEEE Conference on Decision and Control*, 2020, pp. 5283–5290.

[2] Government of Canada, "Ice navigation in Canadian waters," May 2019. [Online]. Available: https://www.ccg-gcc.gc.ca/publications/icebreaking-deglacage/ice-navigation-glaces/page05-eng.html

[3] F. Goerlandt, J. Montewka, W. Zhang, and P. Kujala, "An analysis of ship escort and convoy operations in ice conditions," *Safety science*, vol. 95, pp. 198–209, 2017.

[4] K. Murrant, R. Gash, and J. Mills, "Dynamic path following in ice-covered waters with an autonomous surface ship model," in *IEEE OCEANS*, San Diego–Porto, 2021, pp. 1–4.

[5] H.-T. L. Chiang and L. Tapia, "COLREG-RRT: An RRT-based COLREGS-compliant motion planner for surface vehicle navigation," *IEEE Robotics & Automation Letters*, vol. 3, no. 3, pp. 2024–2031, 2018.

[6] T. Shan, W. Wang, B. Englot, C. Ratti, and D. Rus, "A receding horizon multi-objective planner for autonomous surface vehicles in urban waterways," in *IEEE Conference on Decision and Control*, 2020, pp. 4085–4092.

[7] M. Choi, H. Chung, H. Yamaguchi, and K. Nagakawa, "Arctic sea route path planning based on an uncertain ice prediction model," *Cold Regions Science and Technology*, vol. 109, pp. 61–69, 2015.

[8] V. Aksakalli, D. Oz, A. F. Alkaya, and V. Aydogdu, "Optimal naval path planning in ice-covered waters," *International Journal of Maritime Engineering*, vol. 159, no. A1, 2017.

[9] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[10] C. Daley, S. Alawneh, D. Peters, and B. Colbourne, "GPU-event-mechanics evaluation of ice impact load statistics," in *OTC Arctic Technology Conference*, 2014.

[11] M. Stilman and J. Kuffner, "Planning among movable obstacles with artificial constraints," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1295–1307, Nov. 2008.

[12] Hai-Ning Wu, M. Levihn, and M. Stilman, "Navigation among movable obstacles in unknown environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Oct. 2010, pp. 1433–1438.

[13] J.-y. Zhuang, Y.-m. Su, Y.-l. Liao, and H.-b. Sun, "Motion planning of usv based on marine rules," *Procedia Engineering*, vol. 15, pp. 269–276, 2011.

[14] Y. Kuwata, M. T. Wolf, D. Zarzhitsky, and T. L. Huntsberger, "Safe maritime autonomous navigation with COLREGS, using velocity obstacles," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 110–119, 2013.

[15] T.-H. Hsieh, S. Wang, H. Gong, W. Liu, and N. Xu, "Sea ice warning visualization and path planning for ice navigation based on radar image recognition," *Journal of Marine Science and Technology*, vol. 29, no. 3, pp. 280–290, 2021.

[16] R. M. Gash, K. A. Murrant, J. W. Mills, and D. E. Millan, "Machine vision techniques for situational awareness and path planning in model test basin ice-covered waters," in *IEEE Global Oceans*, 2020, pp. 1–8.

[17] T. I. Fossen, *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[18] H. Kim, D. Kim, J.-U. Shin, H. Kim, and H. Myung, "Angular rate-constrained path planning algorithm for unmanned surface vehicles," *Ocean Engineering*, vol. 84, pp. 37–44, 2014.

[19] X. Liang, P. Jiang, and H. Zhu, "Path planning for unmanned surface vehicle with Dubins curve based on GA," in *IEEE Chinese Automation Congress*, 2020, pp. 5149–5154.

[20] J.-B. Caillau, S. Maslovskaya, T. Mensch, T. Moulinier, and J.-B. Pomet, "Zermelo-Markov-Dubins problem and extensions in marine navigation," in *IEEE Conference on Decision and Control*, 2019, pp. 517–522.

[21] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[22] A. Botros and S. L. Smith, "Multi-start n-dimensional lattice planning with optimal motion primitives," *arXiv preprint arXiv:2107.11467*, 2021.

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

[24] Government of Canada, "Ice glossary," Nov 2020. [Online]. Available: https://www.canada.ca/en/environment-climate-change/services/ice-forecasts-observations/latest-conditions/glossary.html

[25] Y. N. Popov, O. Faddeev, D. Kheisin, and A. Yakovlev, "Strength of ships sailing in ice," Army Foreign Science and Technology Center Charlottesville Va, Tech. Rep., 1969.

[26] C. Daley, "Energy based ice collision forces," in *International Conference on Port and Ocean Engineering under Arctic Conditions*, vol. 2, 1999, pp. 674–686.

[27] R. de Schaetzen, A. Botros, R. Gash, K. Murrant, and S. L. Smith, "Real-time navigation for autonomous surface vehicles in ice-covered waters," *arXiv preprint arXiv:2302.11601*, 2023.

[28] X.-N. Bui, J.-D. Boissonnat, P. Soueres, and J.-P. Laumond, "Shortest path synthesis for Dubins non-holonomic robot," in *IEEE International Conference on Robotics and Automation*, 1994, pp. 2–7.

[29] X. Ma and D. A. Castanon, "Receding horizon planning for Dubins traveling salesman problems," in *IEEE Conference on Decision and Control*, 2006, pp. 5453–5458.

[30] H. Ahani, M. Familian, and R. Ashtari, "Optimum design of a dynamic positioning controller for an offshore vessel," *Journal of Soft Computing and Decision Support Systems*, vol. 7, no. 1, pp. 13–18, 2020.

[31] M. Mehrzadi, Y. Terriche, C.-L. Su, M. B. Othman, J. C. Vasquez, and J. M. Guerrero, "Review of dynamic positioning control in maritime microgrid systems," *Energies*, vol. 13, no. 12, p. 3188, 2020.

[32] V. Blomqvist, "Pymunk: A easy-to-use pythonic rigid body 2d physics library (version 6.2.1)," 2007. [Online]. Available: https://www.pymunk.org