

Non-Prehensile Mobile Agent Interactive Navigation in Cluttered Environments

by

Ninghan Zhong

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2025

© Ninghan Zhong 2025

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Ninghan Zhong was the sole author for Chapters 1, 2, 5, and 6, which were written under the supervision of Prof. Stephen L. Smith. This thesis also includes collaborative work and first-authored peer-reviewed material that has appeared in conference proceedings published by the Institute of Electrical and Electronics Engineers (IEEE).

Chapter 3 is based on the following publication.

1. Ninghan Zhong, Alessandro Potenza, and Stephen L Smith. Autonomous navigation in ice-covered waters with learned predictions on ship-ice interactions. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025. [95]

Ninghan Zhong was the main author of this publication who designed the path planning algorithm, machine learning pipeline, and evaluation procedures. Ninghan Zhong also led the physical experiment setup and evaluation, outlined in Sec. 3.7.3 and 3.7.4, with Alessandro Potenza, a visiting student at the University of Waterloo.

Chapters 4 is based on the following collaborative work.

1. Ninghan Zhong, Steven Caro, Avraiem Iskandar, Megnath Ramesh, and Stephen L Smith. Bench-NPIN: Benchmarking non-prehensile interactive navigation. *arXiv preprint arXiv:2505.12084*, 2025. [94]

Ninghan Zhong was the main author of this work. Ninghan Zhong, Steven Caro, Avraiem Iskandar, and Megnath Ramesh collaboratively developed the *Maze*, *Ship-Ice*, *Box-Delivery* and *Area-Clearing* environments in Sec. 4.3.1, designed the evaluation metrics in Sec. 4.3.3, and trained reference baselines in Sec. 4.4.1.

We acknowledge the helpful feedback and comments on the work presented in Chapters 3 and 4 from Kevin Murrant and Robert Gash at NRC Canada.

Other Publication: I have also contributed on the following work that is not presented in this thesis. This work is currently under resubmission to IEEE Transactions on Robotics.

1. Rodrigue de Schaetzen, Alexander Botros, Ninghan Zhong, Kevin Murrant, Robert Gash, and Stephen L Smith. AUTO-IceNav: A local navigation strategy for autonomous surface ships in broken ice fields. *arXiv preprint arXiv:2411.17155*, 2024.

Abstract

As autonomous mobile robots are increasingly deployed in complex, real-world environments, traditional approaches that rely solely on collision-free navigation often fall short. In many practical scenarios, physical interactions with the environment, such as pushing objects or manipulating articulated structures, are not only unavoidable but also crucial for completing tasks. This thesis studies non-prehensile interactive navigation (NPIN), where robots interact with their surroundings using non-grasping, contact-based actions. These interactions present significant challenges, including complex interaction dynamics, long-horizon planning under uncertainty, and the lack of standardized tools for evaluation.

This thesis addresses these challenges through three core contributions. First, we present a novel learning-based predictive planning algorithm for autonomous surface vehicle (ASV) navigation in ice-covered waters, a real-world instantiation of NPIN. In this scenario, the ASV must interact with dynamic ice floes to reach its goal. We propose a hybrid planning framework that integrates deep learning-based occupancy prediction of obstacle motion with a graph search-based planner. The resulting system is capable of computing safe and efficient trajectories that account for future ice movements, both in simulation and in a physical testbed.

Second, we broaden the scope and introduce Bench-NPIN, the first standardized benchmarking suite for NPIN. Bench-NPIN includes a diverse set of environments and tasks, such as maze navigation, box delivery, and area clearing, which span both navigation-centric and manipulation-centric categories. It also provides unified evaluation metrics and reference baseline algorithms to facilitate fair and reproducible comparisons across different approaches.

Third, we present a forward-looking study into long-horizon planning through generative skill chaining using diffusion models. We investigate how sequences of low-level interaction skills can be generated in one shot using a learned generative model. Furthermore, we introduce an out-of-distribution (OOD) detection mechanism to evaluate the feasibility of these skill sequences. This allows the planner to anticipate failure, reject infeasible plans, and potentially replan in a timely manner.

Together, these studies provide a comprehensive investigation into the foundations, applications, and emerging approaches for NPIN in cluttered and uncertain environments.

Acknowledgements

I would like to sincerely thank my supervisor Professor Stephen L. Smith for his invaluable guidance and support throughout my master’s studies. His insight, patience, and encouragement consistently steered me in the right direction and pushed me forward, while also giving me the freedom to explore and grow independently. I feel fortunate to have worked under his exceptional mentorship on projects that have been both technically engaging and intellectually rewarding.

I am also grateful to my friends and colleagues at the Autonomous Systems Laboratory. I would especially like to thank Rodrigue de Schaetzen for helping me get started with my research, and Shamak Dutta, Megnath Ramesh, and Avraiem Iskandar for the many discussions, collaborations, and shared experiences. The supportive and stimulating environment in the lab has made my time at Waterloo truly memorable.

Finally, I thank my family for their unwavering support and encouragement, and for always believing in me.

Dedication

To my wonderful family, my beloved parents.

Table of Contents

Author’s Declaration	ii
Statement of Contributions	iii
Abstract	iv
Acknowledgements	v
Dedication	vi
List of Figures	ix
List of Tables	xii
1 Introduction	1
1.1 Motivation and Contribution	2
1.2 Thesis Outline	4
2 Literature Review	5
2.1 Path Planning	5
2.2 Interactive Navigation	6
2.3 Robotics Benchmarks	7
2.4 Generative Models	7

3	Autonomous Ship Navigation with Learned Ship-Ice Interactions	9
3.1	Introduction	9
3.2	Related Work	11
3.3	Problem Formulation	12
3.4	Ice Floe Motion Prediction	13
3.5	Path Planning Framework	18
3.6	Algorithm Performance Bound	23
3.7	Evaluation	26
3.8	Conclusion	29
4	Benchmarking Non-prehensile Interactive Navigation	30
4.1	Motivation	30
4.2	Related Work	32
4.3	Bench-NPIN	34
4.4	Evaluating Baselines using Bench-NPIN	43
4.5	Conclusion	49
5	Generative Skill Chaining with Out-of-distribution Predictions	51
5.1	Motivation	51
5.2	Related Work	53
5.3	Preliminaries	55
5.4	OOD Based Infeasibility Predictions	60
5.5	Preliminary Studies	62
5.6	Conclusion	68
6	Conclusion	69
6.1	Discussions and Future Work	70
	References	72

List of Figures

3.1	The left panel shows a planned path from the proposed planner in simulation. The middle panels compare the predicted occupancy after path traversal with the ground truth occupancy. The right panel presents an experimental trial from the ship-ice navigation testbed at the Autonomous Systems Laboratory, University of Waterloo.	10
3.2	Ratio-based occupancy map is computed from the detected obstacles (a). Based on the ship's current pose (blue ship icon), a local occupancy observation (b) is cropped. Ship pose is encoded as footprint (c) and ship action is encoded as swath (d). U-Net is trained to predict the resulting occupancy. Predicted occupancy is stitched back into global occupancy.	15
3.3	A toy example showing occupancy differences. The motion primitives are simplified for illustration purposes. The red dotted boxes highlight the predicted motions of the obstacles. Here left swath is optimal.	20
3.4	Evaluation results from simulation across four concentrations	28
3.5	Testbed evaluations across 20%, 30%, and 40% concentrations.	29
4.1	Illustration of non-prehensile interactive navigation environments with configurable complexity levels. Upper-left shows the <i>Maze</i> environment, the mobile robot must reach the goal point among movable obstacles. Lower-left shows the <i>Ship-Ice</i> environment where an autonomous ship must navigate through ice-covered waters. Upper-right shows <i>Box-Delivery</i> environment, where the robot's objective is to push all boxes into the green receptacle. Lower-right is the <i>Area-Clearing</i> environment, where a robot needs to remove all boxes from within a clearance area outlined by the green rectangle.	31
4.2	The <i>Maze</i> environment (left) and an example egocentric observation for <i>Maze</i> (right).	37

4.3	The <i>Area-Clearing</i> environment (left) and example observation for <i>Area-Clearing</i> (right).	37
4.4	An illustration of three teleoperated paths and the corresponding performance. While the shortest path (red) has the highest efficiency score $E_{\text{nav}} = 0.86$, the excessive collisions with the movable obstacles significantly degrade the effort score $I_{\text{nav}} = 0.56$. In contrast, while the longest path (blue) is collision-free $I_{\text{nav}} = 1.00$, efficiency is largely compromised $E_{\text{nav}} = 0.54$. A balanced path (yellow) potentially offers the best trade-off.	40
4.5	Three teleoperated paths for area clearing, with each path having a task success score $S_{\text{manip}} = 2/3$. While path 1 (red) has the highest efficiency score, its effort score is low as it moves a box longer than its shortest distance to the goal (dotted lines of length l_i^*). In contrast, path 3 (blue) achieves the best effort score by pushing as little as possible, but results in a long path and a low efficiency score. Path 2 (yellow) potentially offers the best trade-off.	42
4.6	Evaluation results for PPO and SAC in <i>Maze</i>	45
4.7	Evaluation results for PPO, SAC, lattice-planning policy (Lattice), and predictive-planning policy (Predictive) in <i>Ship-Ice</i>	46
4.8	Task success scores S_{manip} for the manipulation-centric NPIN baselines included in the benchmark.	47
4.9	Evaluation results for PPO, SAC, and SAM in <i>Box-Delivery</i>	48
4.10	Evaluation results for PPO, SAC, SAM, and GTSP in <i>Area-Clearing</i>	49
5.1	An illustration of long-horizon non-prehensile interactive navigation task - <i>Delicate Transport</i> . The robot is required to push the blue box into the green goal area. However, the blue box is a delicate object and cannot collide with any other objects. We assume that the robot has a pushing bumper with protective design. The blue box is safe to be pushed by the robot, but will be damaged by collisions with other objects in the environment.	53
5.2	An illustration of defining a simple <i>push-box</i> skill. The blue box is the assumed target box to be pushed, and the red dotted circle denotes a close vicinity of the target box that the robot must reach before pushing. The green area is the goal.	57
5.3	A visual illustration of the precondition and postcondition states for motion skills π_a, π_b, π_c	63

5.4	A visualization of sampling results for diffusion models for each skill. The sampled states (both precondition and postcondition states) are represented by dots, while skill parameters are represented as arrows. 30 transitions are sampled for each diffusion model.	64
5.5	A visualization of 30 sampling results for applying Generative Skill Chaining [58] on the feasible skill sequence $\Phi_{\text{feasible}} = \{\pi_a, \pi_b, \pi_c\}$	65
5.6	A visualization of 30 sampling results for applying Generative Skill Chaining [58] on the infeasible skill sequence $\Phi_{\text{infeasible}} = \{\pi_a, \pi_b, \pi_a\}$	66
5.7	OOD scores computed using our Alg. 2 on the generated feasible sequences from Fig. 5.5 for $\Phi_{\text{feasible}} = \{\pi_a, \pi_b, \pi_c\}$ and on the generated infeasible sequences from Fig. 5.6 for $\Phi_{\text{infeasible}} = \{\pi_a, \pi_b, \pi_a\}$	67

List of Tables

3.1	Occupancy change correlations	19
4.1	Non-prehensile interactive navigation tasks in Bench-NPIN.	35
4.2	Reference baselines for each task in Bench-NPIN.	35
4.3	Actions and observations for each task.	35
4.4	Rewards each task.	36

Chapter 1

Introduction

In recent years, the field of robotics has seen remarkable growth, driven by advances in artificial intelligence, perception, and control. As a result, autonomous mobile robots are increasingly being deployed in a wide range of settings, from everyday household environments to high-stakes scenarios such as hospitals, disaster zones, and remote natural environments like polar regions. These robots offer potentials to improve efficiency, safety, and accessibility, assisting humans in both routine and extreme conditions.

However, as deployment expands, so too do the challenges: robots must now autonomously navigate and act in increasingly complex and unstructured real-world settings. A key bottleneck in this expanded deployment is the difficulty of navigating cluttered environments with movable and interactable objects.

Traditionally, robot navigation has focused on avoiding contact with the environment. Algorithms are designed to plan paths that steer clear of obstacles, relying on accurate mapping and sensing. While effective in structured environments, these approaches break down in more complex settings. In many real-world scenarios, interaction with the environment is not a failure, but rather a necessity. For example, a household robot might need to nudge a backpack out of the way to vacuum a floor; a search-and-rescue robot may need to push debris aside to reach a trapped victim; an autonomous ship navigating polar waters may need to traverse through broken ice fields and interact with floating ice.

This class of problems, where robots must physically interact with their surroundings during navigation, falls under the umbrella of Interactive Navigation. Unlike classical navigation, where obstacle avoidance is paramount, interactive navigation recognizes that contact with the environment can be deliberate, purposeful, and even essential.

A common approach to interactive navigation involves mobile manipulators — robots equipped with arms that can grasp and move objects. While effective in principle, these systems come with limitations. Robotic arms can be expensive, heavy, or unsuitable for small mobile platforms. Moreover, reliably grasping and manipulating irregular or dynamic objects remains an open problem in its own right. To overcome these limitations, this thesis explores a different approach: non-prehensile interactive navigation (NPIN), where robots interact with the environment through pushing, nudging, or otherwise making purposeful contact without grasping.

1.1 Motivation and Contribution

Non-prehensile interactive navigation is an emerging research area with great potential. However, it brings with it a unique set of challenges:

1. Modeling Complex Interactions

Robots operating in cluttered environments must often deal with objects of diverse shapes, sizes, and dynamic properties. Predicting the outcomes of robot-object interactions is challenging, especially when contact propagates through multiple objects (e.g., pushing one box that pushes another). This complexity undermines classical numerical methods that depend on accurate dynamics models. Recent advances in deep learning provide a promising alternative: data-driven models that can learn to approximate interaction dynamics. To this end, this thesis focuses on applying machine learning algorithms in non-prehensile interactive navigation. We explore such an approach in the context of autonomous ship navigation in ice-covered waters, where learned models of ship-ice interaction are used to guide planning in complex ice fields.

2. Lack of a Standardized Benchmark

Although interest in NPIN has grown [24, 83, 12], much of the existing work is developed in isolated domains with custom environments and evaluation methods, despite their shared problem structure. This makes it difficult to compare approaches or build upon each other’s progress. In contrast, fields such as locomotion and visual control benefit tremendously from established benchmarks (e.g., HalfCheetah, Ant, RoboSuite) that provide a standardized platform for algorithm comparison and reproducibility. These benchmarks provide unified, rapid ways to evaluate both existing and new algorithms. The field of NPIN lacks such a unifying benchmark for

systematic evaluation and reproducibility. To address this gap, this thesis presents Bench-NPIN, the first unified benchmarking suite for non-prehensile interactive navigation.

3. Planning under High Uncertainty and Long Horizons

Lastly, movable and interactable objects in the environments lead to an exponentially large state space, as different combinations of object interaction may exist. For instance, navigation among movable obstacles (NAMO), which is a subcategory of interactive navigation, is considered NP-hard [20] to be optimally solved. Further, in cluttered environments, a single push may have many downstream consequences, and the same action may yield different results depending on small variations in state. This makes long-horizon planning especially difficult, as small errors can compound and lead to large deviations. Traditional planners that make sequential decisions or rely on auto-regressive predictions may not cope well with this large search space and uncertainty. Inspired by recent work in generative modeling [48, 9, 58], this thesis explores the use of generative skill chaining to model and plan entire trajectories in one shot. We develop an approach allows the robot to reason more holistically about the feasibility and variability of its actions.

The contributions of this thesis are summarized as follows:

1. **Learning-based Predictive Path Planning for Ship Navigation in Ice-Covered Waters:** We study Autonomous Surface Vehicle (ASV) navigation in ice-covered waters, a specific application of non-prehensile interactive navigation. We present a novel framework [95] that combines deep learning and classical planning to plan obstacle-motion-aware navigation paths for the ASV.
2. **Benchmarking Framework:** We present the first benchmark for non-prehensile interactive navigation (Bench-NPIN) [94]. This benchmark provides a suite of environments in both 2D and 3D and includes new metrics tailored to the interactive nature of the tasks.
3. **Generative Skill Planning with Infeasibility Predictions:** We study generative skill learning and long-horizon skill composition for interactive tasks, focusing on infeasibility and uncertainty handling in cluttered environments. In particular, we proposed a new technique for predicting and quantifying infeasibility of a planned sequence of executions, potentially enabling failure anticipations and timely replan.

1.2 Thesis Outline

The rest of the thesis is structured as follows.

We first introduce some literature related to this thesis in Chapter 2. It surveys classical path-planning, recent advances in interactive navigation, benchmarking initiatives in robotics, and modern generative models. Chapter 2 provides a high-level overview of the key ideas, while each subsequent chapter revisits the literature most relevant to its own contributions and examines it in greater depth.

Chapter 3 then begin our NPIN discussion in practice on autonomous-surface-vehicle (ASV) navigation in ice-covered waters. Here, NPIN is not an abstract notion: the ship must physically interact with drifting ice floes to reach its destination. We introduce a learning-augmented planning framework that predicts ice motion and incorporates it into a lattice-based search, demonstrating the benefits of interaction-aware planning.

Building on insights from the ASV study, Chapter 4 broadens the scope. It presents Bench-NPIN, the first standardized benchmark for NPIN. A suite of 2-D simulated tasks, coupled with unified metrics and reference baselines, provides a common platform for comparing algorithms and accelerating progress across diverse NPIN scenarios.

Finally, Chapter 5 looks beyond current capabilities toward long-horizon decision making under uncertainty. We explore generative skill chaining based on diffusion models, showing how entire sequences of low-level interaction skills can be synthesized in one shot and vetted for feasibility with an out-of-distribution detector.

Chapter 6 concludes the thesis. It summarizes the main findings, reflects on open challenges, such as uncertainty quantification and sim-to-real transfer. The chapter also outlines promising directions for future research in non-prehensile interactive navigation.

Chapter 2

Literature Review

The work in this thesis is related to a large body of work in interactive navigation, path planning, benchmarking, and generative models. This chapter positions the thesis within this wider research landscape. In-depth summaries of the most relevant work are provided in the pertaining chapters.

2.1 Path Planning

The goal of path planning is to find a feasible to for a robot to traverse from a start configuration into a goal configuration while satisfying motion constraints. Path planners are typically graph-based or sampling-based.

Graph-search planners typically discretise the workspace into a weighted graph and search for the minimum-cost route. Dijkstra’s algorithm [16] guarantees to find the optimal path but could potentially explore all possible routes. A* algorithm [34] accelerates the search process by adding an admissible heuristic while retaining optimality. The D* algorithm [72] extends A* by enabling real-time replan to deal with new obstacles discovered online.

Sampling methods avoid explicit grid construction by taking a Monte-Carlo approach, which could be a good alternative to deal with large planning spaces due to high degree-of-freedom configuration space. Probabilistic Roadmaps (PRMs) [51] construct a graph of collision-free configurations by randomly sampling the robot’s configuration space, connecting nearby samples that remain collision-free. The roadmap can be quickly queried to find feasible paths. Rapidly-exploring Random Tree (RRT) [55] incrementally grows a

tree from the start configuration toward random samples, quickly exploring large spaces to produce a feasible path. RRT* [49] refines this idea by rewiring the tree during expansion to minimize path cost, guaranteeing asymptotic optimality while retaining much of RRT’s exploratory efficiency.

More recent methods leverage machine learning and integrate neural networks into the planning frameworks. Value-Iteration Networks [77] learn a differentiable approximation of classical value iteration and generalize to unseen grid worlds. In [32], an RRT-based planner with a neural network embedded is proposed. The network is trained to predict locomotion outcome using partial observation for long-range planning. A hierarchical approach is proposed in [81], where an A* planner is used as a global guidance, and a local Reinforcement Learning (RL) based planner outputs robot actions to account for dynamic obstacles.

These approaches assume collisions must be completely avoided. However, mobile robots are often tasked with challenges that require deliberate contact. Sec. 2.2 therefore surveys methods that embrace interaction, setting the stage for Navigation Among Movable Obstacles (NAMO) and the broader Non-Prehensile Interactive Navigation (NPIN) literature.

2.2 Interactive Navigation

Robotic navigation has evolved from purely collision-free path-finding [54] to settings where physical interaction is integral rather than accidental. Works in [74, 73, 57] studied Navigation Among Movable Obstacles (NAMO), a sub-branch of Interactive Navigation. These methods reason from a geometric perspective to compute a sequence of pushing and pulling actions to arrange objects to allow for navigation.

Other methods leverage machine learning to make estimations about obstacle interactions. For instance, [89] trained a Reinforcement Learning policy that learns the robot-object interaction dynamics. The RL policy is then integrated into a Model Predictive Path Integral (MPPI) framework to relocate obstacles out of a tracked global path. In [92], an A* planner first computes a global path, and a neural network model is trained to predict the estimated time to remove a path-blocking obstacle for deciding whether to take a detour.

Non-prehensile Interactive Navigation (NPIN) is a branch of Interactive Navigation where the robot uses non-grasping actions, such as pushing or nudging, to interact with the environments. For instance, [82] leverages Curriculum Reinforcement Learning to train

a mobile pushing platform to navigate through diverse and complex indoor environments. The mobile pusher learned to push open doors and push away obstacles blocking a navigating hallway. More in-depth discussion on works related to non-prehensile interactive navigation can be found in Chapter 4.2.

2.3 Robotics Benchmarks

Benchmarks have played an significant role in accelerating progress across robotics and embodied AI. Early efforts targeted locomotion, leveraging the MuJoCo physics engine [79] to create a suite of under-actuated agents such as *Hopper*, *HalfCheetah*, and *Ant* that remain de-facto standards for reinforcement-learning (RL) research [23, 7]. The DeepMind Control Suite extends this idea with a unified API and interpretable reward functions [78], while RoboSuite introduces high-fidelity robot arms and grippers for manipulation-centric tasks [96].

As sensing, perception and computation power improved, the community focus shifted toward navigation in 3-D environments. Gibson Env [87] and Habitat Sim [69] focused on photorealistic, scan-based indoor worlds and introduced novel evaluation metrics such as SPL (Success weighted by Path Length). Subsequent work addressed social and dynamic settings, such as SocialGym [39] for crowd-aware motion and CARLA [17] for autonomous driving in urban traffic. Benchmarks for long-horizon tasks, such as RL Bench [47] and ManiSkill [59], encourage learning generalizable policies for multi-stage tasks.

Finally, there is a growing movement toward **real-robot benchmarks** that close the sim-to-real loop. For instance, the Bridge Dataset [19] and *Real Robot Challenge* [31] take the initiatives to evaluate and improve robots performance in open-world environments.

Despite this breadth, most benchmarks implicitly penalize any robot–environment contact. The few that embrace interaction include Interactive Gibson [86] and our proposed Bench-NPIN [94], which are described in more details in Chapter 4.

2.4 Generative Models

Generative modeling provides a complementary perspective to classical planning and policy search. Early probabilistic models such as Hidden Markov models (HMMs) and Gaussian mixture models (GMMs) are examples of traditional generative models. Modern deep learning approaches fall into three broad families:

Variational auto-encoders (VAEs) [52] jointly trains an encoder–decoder pair to map data into a continuous, regularized latent distribution, so that new samples can be generated by drawing from this learned latent space. VAEs has being used to predict multi-modal human trajectories [46] for socially-aware navigation. In [8], a sliding-window VAE is used for industrial robot anomaly detection.

Generative adversarial networks (GANs) [29] employ a generator network against a discriminator in a minimax game where the generator learns to create realistic synthetic data that fools the discriminator, which simultaneously learns to distinguish generated samples from real ones, driving both networks to improve. The methods were original proposed for sharp image synthesis, but has been used in robotics for various applications. For instance, GANs have been used to learn inverse dynamics for robotic arms [67] and to generate robot actions from text [3].

Denoising diffusion probabilistic models (DDPMs) [37] have become state-of-the-art for high-dimensional generation. The general idea is to train a neural network that guides a denoising process that transforms samples of pure noise into clean data with desired properties. Recent works such as *Diffusion Policy* framework [9] and *Diffuser* [48] have shown remarkable performance in generating manipulation trajectories and flexible behaviors. More details regarding diffusion models can be found in Chapter 5.

Chapter 3

Autonomous Ship Navigation with Learned Ship-Ice Interactions

In this chapter, we study applying machine learning in predictive path planning for autonomous ship navigation in ice-covered waters, a non-prehensile interactive navigation in real-world application context.

3.1 Introduction

In recent years, the polar regions have been attracting international attention. The Arctic area offers shorter shipping routes and rich natural resources while tourism in Antarctica is on the rise [40]. However, navigating in ice-covered waters such as the Arctic areas poses high risks due to significantly higher ice concentrations compared to typical maritime environments [28]. Recent advancement of autonomous surface vehicles (ASVs) holds promise for safer and more efficient navigation in these icy waters. Nevertheless, current approaches still face substantial challenges. The high concentration of ice often makes a collision-free path infeasible. Additionally, the dynamic and chaotic nature of ice movements in response to ship maneuvers adds another layer of complexity to the task [14].

In this work, we address the problem of path planning for an autonomous surface vehicle in ice-covered waters. We assume the ASV is designed for standard maritime operations and has limited ship-ice collision protection [14]. Our work aims to compute a reference path for the ASV such that both ship-ice collisions and traveling distance are minimized.

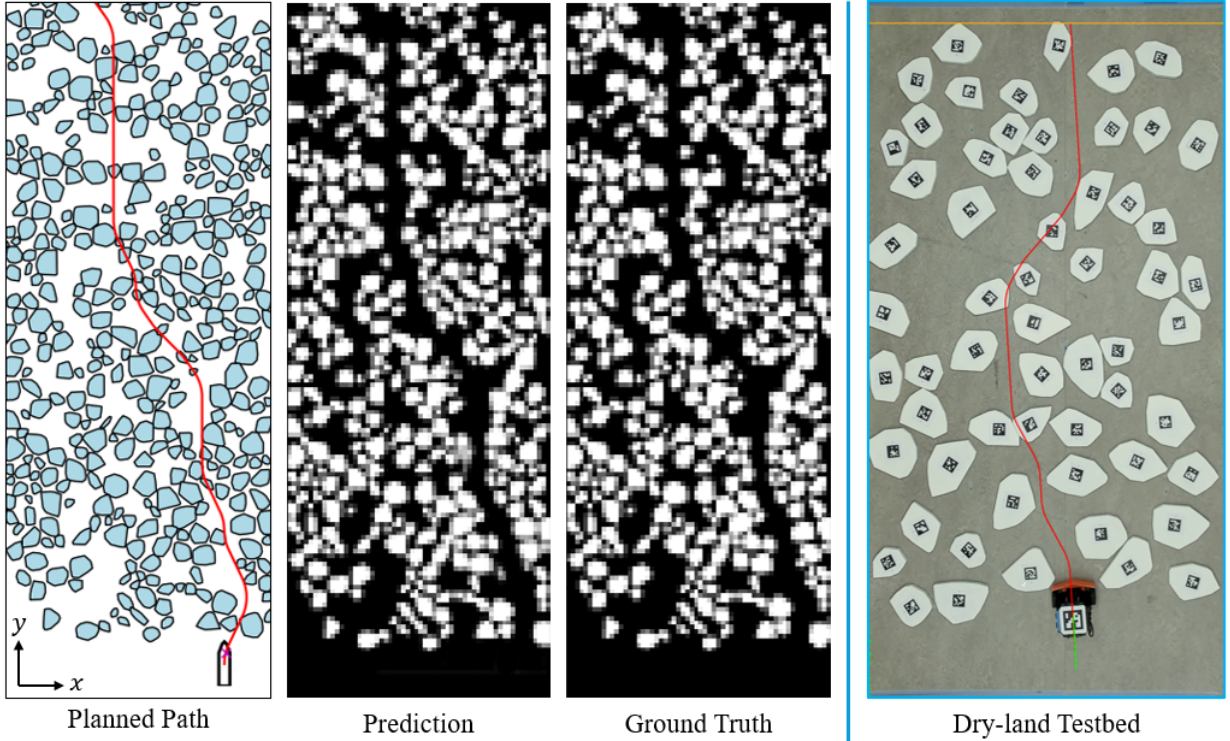


Figure 3.1: The left panel shows a planned path from the proposed planner in simulation. The middle panels compare the predicted occupancy after path traversal with the ground truth occupancy. The right panel presents an experimental trial from the ship-ice navigation testbed at the Autonomous Systems Laboratory, University of Waterloo.

Existing works on ASV path planning primarily focused on finding a collision-free path [5, 10, 53], and are not easily generalizable to environments with high ice concentrations, where collision-free paths are typically non-existent, as illustrated in Fig. 3.1. While the works in [14] and [27] address this challenge, they do not consider ice motion during planning. As ice concentration increases, the motion of the ice becomes more complex due to interactions with the ship and other ice floes, resulting in a more dynamic environment. Consequently, planners that treat ice floes as static objects become less effective. For example, the performance improvement seen with the planner proposed in [14] diminishes with 50% ice concentration. In contrast, our approach incorporates predicted ice motion into the path planning process, ensuring robustness in both high and low ice concentration environments.

Contributions In this work, we propose a deep learning framework that predicts the coarse dynamics of the ice motion in response to the ship actions through occupancy estimation. The learning process is guided with a novel physics-derived loss function tailored to our occupancy formulation. To leverage the prediction results for planning, we present a simple yet empirically effective cost function based on occupancy maps to penalizes collisions. Further, we propose a graph search planner that seamlessly incorporates the learned model by caching the intermediate prediction results. Finally, the proposed planner is evaluated both in simulation and in a physical testbed, and superior performance is demonstrated.

3.2 Related Work

While path planning for ASV ice navigation has been extensively studied, much of the existing literature focuses on computing a path on a global scope [11, 56, 71]. For instance, the planned paths in [11] and [56] span hundreds of kilometers while the paths considered in [71] take days to traverse. While these works demonstrated good performance, local planners are required to achieve full ship autonomy. In ASV local planning, however, most work aims to compute a collision-free path [5, 10, 53, 41], which does not generalize to environments where a collision-free path does not exist.

Perhaps the most comparable works to this work are [27] and [14]. In [27], the authors leverage morphological skeletons to represent the open-water areas given an overhead image of the ice field. An A* algorithm is then applied to find a path in the resulting graph. In [14], a lattice-based planner with a kinetic-energy based cost function is proposed to generate paths with minimal collisions. While [27] and [14] apply to our settings, the authors treat obstacles as static during planning.

Ice Prediction Models: Ice motion predictions, together with ice-ice and ship-ice interactions, have been primarily studied with numerical methods and empirical formulas [44, 76, 36, 43, 42]. While these approaches demonstrate high accuracy, numerical simulations are computationally intensive, limiting their real-time applicability. On the other hand, empirical formulas are computationally efficient, but often require environmental data that is not available online from ASV onboard sensors.

More recent works leverage machine learning, but the majority of these approaches have considered ice motion prediction on a global scale for sea ice forecasting. Using satellite images, convolutional LSTM neural networks have been used to predict ice motion for several days in the future [63, 93]. In [38], linear regression and CNN models are used to perform present-day Arctic sea ice motion prediction.

Few works address local ice floe motion with learning approaches. In [90], a multi-step deep-learning based perception pipeline is proposed to track the motion and velocity of river ice, but no prediction is performed. In [61], a physics-informed neural network incorporating attention mechanisms is proposed to predict ice floe longitudinal motion in response to waves. While this work addresses ice motion in the scale of our setting, it is not applicable to ASV planning as ship maneuvers are not considered. To the best of our knowledge, our work provides the first ice motion prediction model designed for real-time ASV local navigation in icy conditions.

3.3 Problem Formulation

In this section we introduce the problem of ship navigation through an ice-covered waters with minimal collisions.

3.3.1 Environment

We define the water surface in which the ship navigates as a 2D surface $\mathcal{W} \subseteq \mathbb{R}^2$. We further restrict our navigation environment as a rectangular channel $\mathcal{C} \subset \mathcal{W}$ where the length is parallel to the y-axis, as shown in Fig. 3.1 (left). Note that this restriction does not limit the applicability of planning, as longer and curved channels can be partitioned into rectangular sections [14]. The objective is to navigate the ship to pass a goal line $\mathcal{G} \subset \mathcal{C}$ with a constant y-value. We assume all ice pieces can be detected by some perception setup at any given time and we treat each ice piece as an obstacle. We group the obstacles into a set $W_{obs} = \{p_1, \dots, p_m\}$ with $p_i \in W_{obs}$ denoting an individual obstacle and m the total number of detected ice pieces.

3.3.2 Ice Motion Prediction

To address the dynamic nature of ice floes in navigation, our objective is to predict ice field evolution as a result of ship maneuvers. Given the obstacles W_{obs} , we can compute a representation, denoted as O , that captures the state of the entire obstacle field. In this work, we use an occupancy map, but other representations could be used. Let $\mathcal{O}$, $\mathcal{S}$, and $\mathcal{A}$ be the state space of the obstacle field, ship state space, and ship action space, respectively, the prediction task is formulated as finding a function $f : \mathcal{O} \times \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{O}$. Specifically,

$$f(O, s, a) = O' \quad (3.1)$$

takes as inputs the current state of the obstacle field $O \in \mathcal{O}$, the state of the ship $s \in \mathcal{S}$, and a ship action $a \in \mathcal{A}$ to predict the resulting state of the obstacle field $O' \in \mathcal{O}$ after the ship action is taken. In Sec. 3.4, we detail the representation used for each of the arguments.

Note that the input O and the prediction O' being in the same space $\mathcal{O}$ allows making predictions on a sequence of actions. Specifically, given the current obstacle field state O_1 and a paired sequence of n ship actions and ship state transitions $\{(s_1, a_1), (s_2, a_2), \dots, (s_n, a_n)\}$, the resulting obstacle field state O'_n after taking the n actions can be predicted as

$$f(\dots f(f(O_1, s_1, a_1), s_2, a_2) \dots, s_n, a_n) = O'_n. \quad (3.2)$$

We assume that the primary driving influence on ice motion in static waters is ship maneuvers. However, we acknowledge that other factors, such as wind and ship waves, also affect ice motion. The inclusion of these additional factors is left for future work.

3.3.3 Path Planning

Given a reference path Π and let $d(\Pi)$ be the total path length of Π , we compute a cost function u that penalizes both path length and collision as

$$u(\Pi) = d(\Pi) + \alpha \cdot C(\Pi) \quad (3.3)$$

where $C(\Pi) \geq 0$ is the collision cost function described in Section 3.5.2, and $\alpha \geq 0$ is a tunable scaling parameter. Given a ship start position, a goal line $\mathcal{G}$, and a set of detected obstacles W_{obs} , we seek a reference path Π from start to $\mathcal{G}$ that minimizes u .

3.4 Ice Floe Motion Prediction

In this section, we present our proposed deep learning pipeline to predict ice motions for ship planning.

3.4.1 Space Representations

To account for the complex motions of ice floes, we leverage occupancy as a coarse representation of obstacle positions and orientations, shown in Fig. 3.2a. To obtain this representation, the channel $\mathcal{C}$ is first discretized into a grid map M . Given the set of obstacles W_{obs} detected from some perception system, we compute a global occupancy map O_{global} of the same resolution as M where each occupancy grid $O_{\text{global}}[x, y] \in [0, 1]$ represents the ratio between the obstacle-occupied area and the grid area at grid $M[x, y]$. Note that our occupancy map differs from the typical form in the autonomous navigation literature, where occupancy maps are usually encoded as binary maps where each grid is either occupied or empty. By leveraging occupancy ratio, this coarse obstacle representation preserves the boundary and separation information of the obstacles.

The state of the ship s in Eq. 3.1 is represented as the ship footprint to encode spatial features such as size and shape, shown in Fig. 3.2c. Given the current pose η of the ship, we define footprint $\mathcal{F}(\eta)$ to be the set of grid cells occupied by the ship body in map M .

We adopt the control set from [14] as the action set of the ship, which is a set of motion primitives where each motion primitive is a short feasible path of the ship. In our setup, the motion primitives' lengths range from approximately 1 - 6 ship length. Given a control set $\mathcal{P}$, the ship action a in Eq. 3.1 is effectively a motion primitive $\pi \in \mathcal{P}$, and is represented as the swath of the motion primitive. Specifically, we define the swath $\mathcal{T}(\pi)$ as the set of grid cells in M that are swept by the ship footprint after executing the motion primitive π , as shown in Fig. 3.2d.

Note that the global occupancy map O_{global} , footprint $\mathcal{F}(\eta)$, and swath $\mathcal{T}(\pi)$ are all computed with the same resolution as the grid map M such that the spatial relations between the obstacles, ship pose, and motion primitive are preserved in the representations. Further, given a control set $\mathcal{P}$, a path can be constructed by concatenating a sequence of motion primitives. The resulting occupancy after the entire path is traversed can then be predicted in a sequential manner, as outlined in Eq. 3.2. An illustration of such sequential predictions for a concatenated path is presented in Fig. 3.1 middle panels.

3.4.2 Occupancy Prediction

Observe that (Fig. 3.2d and Fig. 3.3) each motion primitive within the control set is a short path relatively local to the ship position with respect to the entire ice field. Hence a motion primitive is likely to only trigger ice motions within a local neighborhood. With this observation, we perform predictions within a locally cropped region $O \subseteq O_{\text{global}}$ with a fixed

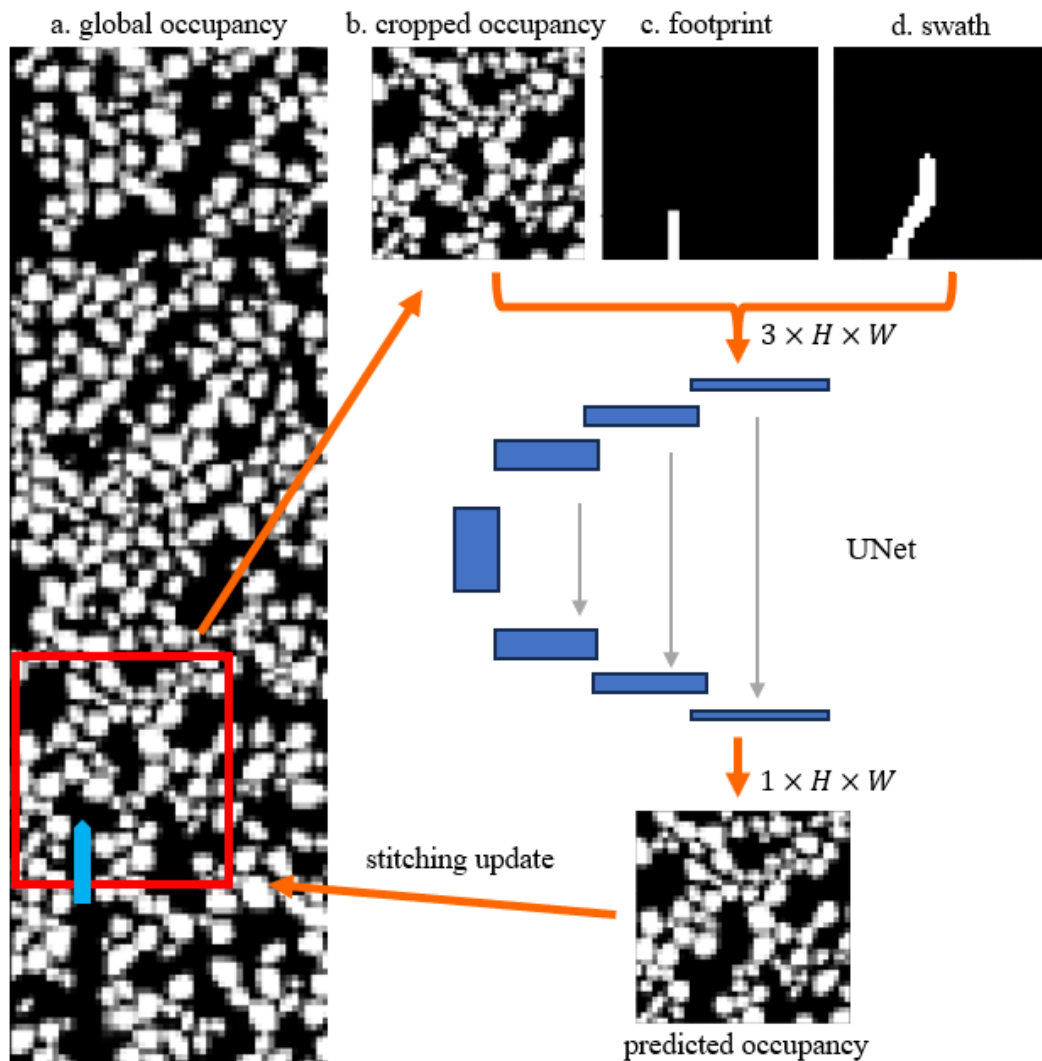


Figure 3.2: Ratio-based occupancy map is computed from the detected obstacles (a). Based on the ship's current pose (blue ship icon), a local occupancy observation (b) is cropped. Ship pose is encoded as footprint (c) and ship action is encoded as swath (d). U-Net is trained to predict the resulting occupancy. Predicted occupancy is stitched back into global occupancy.

window size $H \times W$ from the global occupancy map based on the current ship position, as illustrated in Fig. 3.2b. The footprint $\mathcal{F}(\eta)$ and swath $\mathcal{T}(\pi)$ are also spatially aligned with the cropped occupancy O . After each prediction, we update the global occupancy map by simply stitching back the locally predicted occupancy. This local prediction approach allows our model to concentrate inference power to motion-rich areas. Special care is required to select an appropriate window size $H \times W$, as a window that is too small might not capture all ship-ice and ice-ice collisions, resulting in obstacles being pushed out of its boundary and hence inconsistency in global occupancy map updates.

The neural network model input $X \in \mathbb{R}^{3 \times H \times W}$ is a 3-channel image where one channel is a local occupancy map observation $O \subseteq O_{\text{global}}$, one channel is the footprint $\mathcal{F}(\eta)$ and one channel is the swath $\mathcal{T}(\pi)$. Note that the footprint and swath channels are binary channels as footprint and swath are encoded as binary images. However, the occupancy channel is a gray-scale channel as each occupancy grid $O[x, y]$ is a value from $[0, 1]$.

Our model architecture is based on the U-Net [68]. In particular, our model has an encoder stack consisting of convolutional downsampling layers and a decoder stack with transposed convolutional upsampling layers and skip connections. Inspired by [2], residual convolution is used for the bottleneck to reduce model complexity. The model outputs a single channel of the predicted occupancy map with the same spatial dimension as the input occupancy map.

3.4.3 Model Training

Training Data

We obtain our training data from the open-sourced ship-ice navigation simulator (<https://github.com/rdesc/AUTO-IceNav>) from [14] in a self-supervised manner. The training set covers 20%, 30%, 40%, and 50% ice concentration environments and for each concentration, a total of 2,000,000 entries are collected. Each training entry is a tuple $(O, \mathcal{F}(\eta), \mathcal{T}(\pi), O')$ where the initial occupancy observation O , footprint $\mathcal{F}(\eta)$ and swath $\mathcal{T}(\pi)$ are the model input, and O' is the ground truth resulting occupancy after the ship executing the motion primitive π at pose η . In other words, O and O' are “snapshots” of the occupancy map at the start and end of executing a motion primitive, respectively. To obtain a diverse set of ship-ice and ice-ice interaction samples, we deploy a random navigation policy for the ship during the data collection process. At each step, the ship randomly samples a motion primitive $\pi \in \mathcal{P}$ to execute.

Loss Functions

We define the occupancy loss as a Huber loss [45] between the predicted occupancy $\hat{O}'$ and the ground truth O' given its robustness to outliers.

To guide the learning process toward more physically-plausible predictions, we present a novel physics-based loss tailored to our occupancy formulation. The core assumption is that the total mass of the ice floes is conserved before and after the collisions from the ship executing a maneuver. Let m_p represent the mass of an ice floe p , and let $W_{obs}^i = \{p_1^i, \dots, p_m^i\}$ and $W_{obs}^f = \{p_1^f, \dots, p_n^f\}$ be the sets of obstacle ice floes before and after ship executing a motion primitive, respectively. By conservation of mass, we have

$$\sum_{p^i \in W_{obs}^i} m_{p^i} = \sum_{p^f \in W_{obs}^f} m_{p^f}. \quad (3.4)$$

Similar to [14], we assume that the ice floes are 3D polygons with uniform density ρ and thickness d . It follows that the mass of an ice floe is $m_p = \sigma_p \rho d$, where σ_p denotes the surface area of an ice floe p . We rewrite Eq. 3.4 as

$$\begin{aligned} \sum_{p^i \in W_{obs}^i} \sigma_{p^i} \rho d &= \sum_{p^f \in W_{obs}^f} \sigma_{p^f} \rho d \\ \sum_{p^i \in W_{obs}^i} \sigma_{p^i} &= \sum_{p^f \in W_{obs}^f} \sigma_{p^f}. \end{aligned} \quad (3.5)$$

Recall that each grid in our occupancy map is a ratio between the obstacle-occupied area and the grid area. Let s_g denote the grid size of the occupancy map, and let o_p be the total occupancy value the ice floe p contributes to the occupancy map, it follows that $\sigma_p = s_g o_p$. Further, note that the sum of all ice floes' occupancy o_p is simply the sum of occupancy values in the occupancy map, namely $\sum_{p^i \in W_{obs}^i} o_{p^i} = \text{Sum}(O)$ and $\sum_{p^f \in W_{obs}^f} o_{p^f} = \text{Sum}(O')$. With these observations, Eq. 3.4 can be rewritten as

$$\begin{aligned} \sum_{p^i \in W_{obs}^i} o_{p^i} s_g &= \sum_{p^f \in W_{obs}^f} o_{p^f} s_g \\ \sum_{p^i \in W_{obs}^i} o_{p^i} &= \sum_{p^f \in W_{obs}^f} o_{p^f} = \\ \text{Sum}(O) &= \text{Sum}(O'). \end{aligned} \quad (3.6)$$

In other words, the sum of all cell values from the occupancy map before and after executing a motion primitive is conserved. Hence, given a training set entry $(O, \mathcal{F}(\eta), \mathcal{T}(\pi), O')$ and

a prediction $\hat{O}'$, we define a conservation loss as $\text{MSE}(\text{Sum}(O), \text{Sum}(\hat{O}'))$. The final loss is constructed as

$$\mathcal{L} = \text{Huber}(O', \hat{O}') + \lambda \text{MSE}(\text{Sum}(O), \text{Sum}(\hat{O}')), \quad (3.7)$$

which is a linear combination of the occupancy loss and conservation loss scaled by λ . In practice, the inclusion of the conservation loss prevents the neural network predictor from expanding or shrinking the obstacles in the occupancy map predictions, preserving obstacle shapes and sizes.

3.5 Path Planning Framework

We integrate a lattice-based planner with the trained neural network predictor via a simple yet empirically effective cost function based on occupancy map difference. We reduce the number of network inferences by caching intermediate prediction results.

3.5.1 Lattice Planning with Motion Primitives

We adopt the lattice-based planner from [14]. The robot state space is discretized into a state lattice and a pre-computed set of motion primitives, called a control set $\mathcal{P}$, is used to repeatedly sample feasible motions. Each motion primitive $\pi \in \mathcal{P}$ is a path that connects two lattice states. The total planned path $\Pi = \{\pi_1, \pi_2, \dots, \pi_n\}$ is then a concatenated sequence of motion primitives π_i where $i = 1, \dots, n$. Finally, the planning problem is reduced to searching for a sequence of motion primitives that minimizes the cost function at Eq. 3.3 from the start state to the goal line.

3.5.2 Collision Cost with Occupancy

To leverage ice motion prediction, our key observation is that the change in ice floe occupancy map is strongly correlated with our physical quantities of interests, such as the impulse and kinetic energy loss from ship-ice collisions. We define an occupancy difference function $\text{diff} : \mathcal{O} \times \mathcal{O} \rightarrow \mathbb{R}$ to compute a scalar difference between two occupancy maps. Let O and O' be the ice floe occupancy maps before and after the ship executes a motion primitive, respectively.

We begin by presenting our empirical analysis on how the occupancy change, denoted as $\text{diff}(O, O')$, that results from a ship motion primitive relates to the ship's kinetic energy

	KE Loss	Impulse	W_{approx}
<i>MSE</i>	0.77	0.95	0.96
<i>-SSIM</i>	0.70	0.88	0.90
<i>EMD</i>	0.66	0.88	0.89

Table 3.1: Occupancy change correlations loss, collision impulse and approximated work done (W_{approx} described in Sec. 3.7.1) during the execution of the motion primitive. For this analysis, we use occupancy maps that match the input/output dimensions of our learned predictor.

We examined three different methods to evaluate the function $\text{diff}(O, O')$ that measures occupancy differences: mean-squared error (MSE), negated structural similarity (-SSIM), and earth-mover distance (EMD). Experiments of ship-ice collisions are generated from 100 trials of random ship navigations in each of the 20%, 30%, 40%, and 50% concentration environments. These experiments provide collision entries that contain the occupancy changes measure by the three difference methods and the physical quantities of interests after the ship executing a motion primitive. The correlations between the occupancy changes and the collision metrics are presented in Table 3.1. As shown, occupancy changes from all difference-measurement methods are highly correlated with the kinetic energy loss, impulse and approximated work. This strong correlation is intuitive because the occupancy difference correlates with ice field changes. A larger ice field change indicates a greater displacement of ice floes by the ship maneuvers, which results in heavier ship-ice collisions and greater ship navigation efforts.

Given this observation, we formulate our collision cost based on occupancy difference, assuming that minimizing occupancy changes minimizes collisions. Specifically, let $\pi \in \mathcal{P}$ be a motion primitive, the collision cost c_π of taking this motion primitive is computed as

$$c_\pi = \text{diff}(O, O'), \quad (3.8)$$

where O and O' are the ice floe occupancy maps before and after the ship executes the motion primitive π , respectively. For the rest of this work, we adopt MSE as the method for computing occupancy difference, given its highest empirical correlation with our collision metrics and its computational efficiency. Fig. 3.3 shows an illustrative example of this cost.

Finally, given the collision cost from Eq. 3.8 and a concatenated total path $\Pi = \{\pi_1, \pi_2, \dots, \pi_n\}$ from a sequence of motion primitives π_i where $i = 1, \dots, n$, we define the collision cost function C in Eq. 3.3 for the entire path as

$$C(\Pi) = \sum_{\pi \in \Pi} \text{diff}(O_\pi, O'_\pi), \quad (3.9)$$

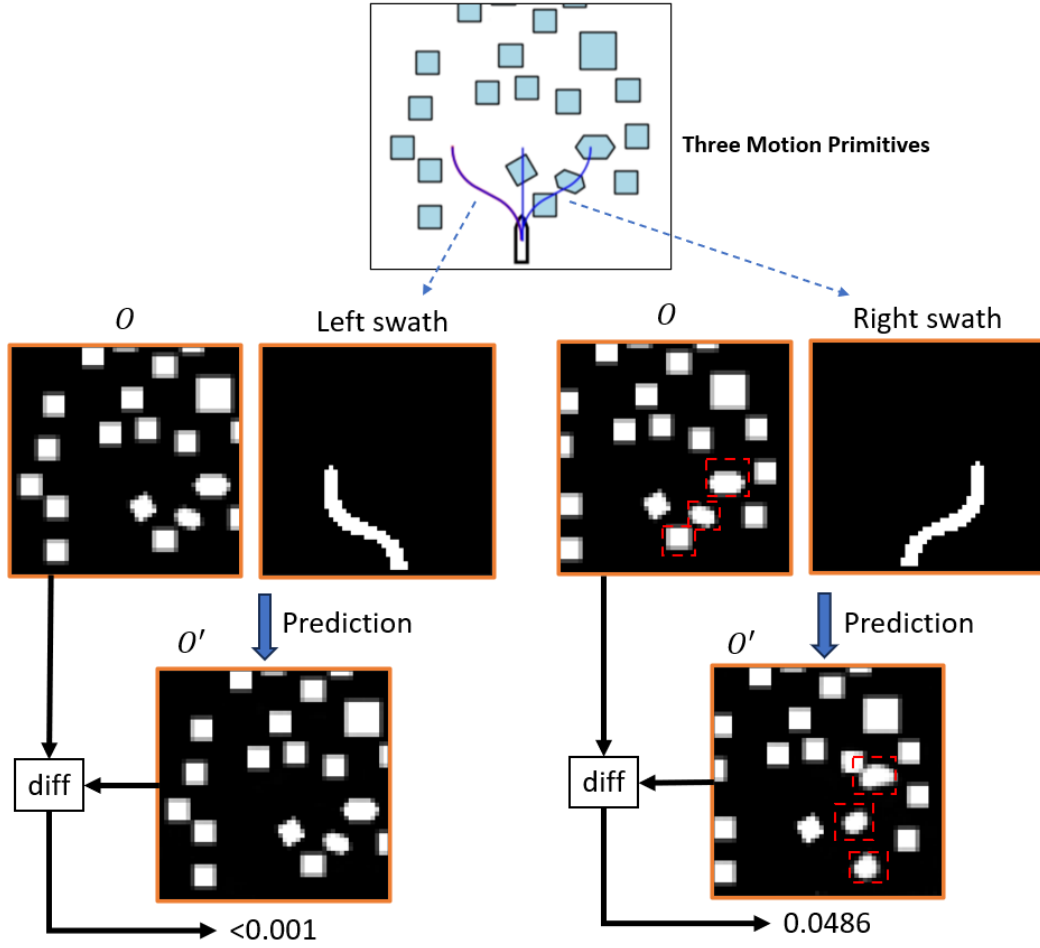


Figure 3.3: A toy example showing occupancy differences. The motion primitives are simplified for illustration purposes. The red dotted boxes highlight the predicted motions of the obstacles. Here left swath is optimal.

where O_π and O'_π is the ice floe occupancy before and after the ship executing motion primitive π , respectively.

3.5.3 A* Search with Occupancy Estimation

Presented in Alg. 1, our lattice-based planner is based on the A* search algorithm [34] and integrates the predictive component to be obstacle-motion-aware during planning. The algorithm takes as input the current 2D pose of the ship as the start node $n_s = \langle x_s, y_s, \theta_s \rangle$, the current global occupancy observation O_s^{global} , and a goal line y_{goal} .

During each iteration, the current node n is expanded to its neighbor n' via a feasible motion primitive π , denoted by the overloaded $+$ operator at Line 11. A local occupancy O is cropped from the global occupancy O^{global} based on the ship pose at node n (Line 13) and is then used for occupancy prediction (Line 14). The edge cost of π is then computed as a linear combination of the occupancy-difference collision cost discussed in 3.5.2 scaled by α and the motion primitive length $d(\pi)$ (Line 15). The predicted local occupancy O' is stitched back for an updated global occupancy map O'^{global} (Line 16), which is then cached if the neighbor n' is visited for the first time or revisited with a better path (Lines 22). In a future iteration when the node n' is being expanded, the global occupancy at n' predicted previously is retrieved for further predictions (Line 9). Additionally, by leveraging a GPU in practice, the network inference can be further optimized by performing batched predictions by constructing the inferences from node n to all its neighbors n' as a batch.

3.5.4 Algorithm Analysis

Here we analyze Alg. 1 runtime and performance.

Runtime

While Alg. 1 has the same structure as A^* or Dijkstra’s algorithm, given that the predictor f is a deep neural network, the bottleneck operation is the prediction in Line 14, rather than the priority queue operation in typical cases. Note that in Alg. 1, each node is expanded at most once and is then added to the closed set CLOSED (Line 8). This implies that Alg. 1 runtime scales linearly with the number of edges, or motion primitives (Line 10), explored during the search. Let $|E|$ be the number of edges in the expanded search graph from Alg. 1, it follows that the runtime is $O(|E|P)$, where P is the time for each forward pass of the predictor.

Algorithm 1: A* Search with Occupancy Prediction

Input: $n_s, O_s^{\text{global}}, y_{\text{goal}}$
1 OPEN $\leftarrow$ min priority queue containing n_s
2 CLOSED $\leftarrow$ empty set
3 $g \leftarrow$ dict that stores cost-so-far
4 $\text{occMemo}[n_s] \leftarrow O_s^{\text{global}}$ // cache first global occupancy
5 **while** OPEN not empty **do**
6 $n \leftarrow \text{OPEN.extractMin}()$
7 **if** n reaches y_{goal} **then** reconstruct and **return** path
8 CLOSED.add(n)
9 $O^{\text{global}} \leftarrow \text{occMemo}[n]$ // retrieve cached prediction
10 **for all** π in $\mathcal{P}$ **do**
11 $n' \leftarrow n + \pi$ // get neighbor
12 **if** $n' \in \text{CLOSED}$ **then continue**
13 $O \leftarrow \text{crop}(O^{\text{global}}, n)$ // crop local occupancy
14 $O' \leftarrow f(O, \mathcal{F}(n), \mathcal{T}(\pi))$ // prediction, Sec. 3.4
15 $\text{cost} \leftarrow g(n) + \alpha \cdot \text{diff}(O, O') + d(\pi)$
16 $O'^{\text{global}} \leftarrow \text{stitch}(O^{\text{global}}, O')$ // stitch prediction
17 **if** $n' \notin \text{OPEN}$ **or** $\text{cost} < g(n')$ **then**
18 $g(n') \leftarrow \text{cost}$
19 $F \leftarrow g(n') + h(n')$ // h heuristic from [14]
20 OPEN.enq(n', F) // enqueue or update n'
21 Update n' parent pointer
22 $\text{occMemo}[n'] \leftarrow O'^{\text{global}}$ // cache prediction
23 **return** False

3.5.5 Performance

We note that by considering obstacle motions, the path finding problem lacks optimal substructures. Namely, in an arbitrary optimal path $n_s \rightsquigarrow n \rightsquigarrow n_{\text{goal}}$ from start n_s to the goal n_{goal} that passes through node n , the sub-path $n_s \rightsquigarrow n$ is not necessarily the optimal path from n_s to n . This could be the cases where, by following the sub-optimal path $n_s \rightsquigarrow n$ to n , the ship pushes the ice floes into a desirable configuration, resulting in the entire path $n_s \rightsquigarrow n \rightsquigarrow n_{\text{goal}}$ being optimal. Further, optimally solving the navigation among movable obstacle problem is generally considered as NP-hard [20]. Consequently, Alg. 1 lacks theoretical optimality guarantee. Hence, we present the following constant-factor guarantee to upper-bound the performance of Alg. 1.

Theorem 1. *Given a problem instance $I = (n_s, O_s, y_{\text{goal}})$, let $\text{OPT}(I)$ denote the optimal path and let $\text{Alg}(I)$ denote the path returned by Alg. 1 assuming perfect occupancy predictions. Let $l_{\min}$ be the distance of the shortest action primitive, the cost of $\text{Alg}(I)$ is upper-bounded as*

$$u(\text{Alg}(I)) \leq (1 + \frac{\alpha}{l_{\min}}) \cdot u(\text{OPT}(I)) \quad (3.10)$$

where u and α are described in Eq. 3.3.

We present the proof of Theorem. 1 in the next section. We acknowledge that Theorem. 1 may not hold under imperfect predictions, and we leave the incorporation of prediction error uncertainty to future work.

3.6 Algorithm Performance Bound

Recall that given an edge $e = (n_1, n_2)$ that connects two nodes n_1 and n_2 , the cost of the edge $u(e)$ is computed as

$$u(e) = d(e) + \alpha \cdot \text{MSE}(O_1, O_2), \quad (3.11)$$

where O_1 and O_2 are the occupancy maps at nodes n_1 and n_2 , respectively (Alg. 1, Line 15). This is a linear combination of the edge length $d(e)$ and the occupancy difference from taking the edge. For simplicity we use $u(\cdot)$ to denote both the cost of a path and an edge. By noting that each cell value of an occupancy map O is a value from $[0, 1]$, we present the following observation.

Observation 1 (Maximal and Minimal Edge Weights). *For each edge e , the edge cost is bounded as $d(e) \leq u(e) \leq d(e) + \alpha$.*

Let $I = (n_s, O_s, y_{\text{goal}})$ denote an arbitrary problem instance and let $\text{OPT}(I) = n_0 \rightarrow n_1 \rightarrow \dots \rightarrow n_{k-1} \rightarrow n_k$ be the optimal path with $k \geq 1$ edges, where $n_0 = n_s$ and $n_k = n_{\text{goal}}$. We let e_i with $i \in \{1, \dots, k\}$ denote the directed edge on the optimal path from node n_{i-1} to node n_i .

Recall that in our problem setup the goal is defined to be a line y_{goal} . So here n_k is an arbitrary goal node n_{goal} whose y-value is greater than y_{goal} , and the multiplicity of the goal nodes does not affect our proof. We now prove the following Lemma.

Lemma 1. *For all nodes n_i with $i = \{1, \dots, k\}$ on the optimal path, after running Alg. 1, the cost-so-far for n_i found by Alg. 1, denoted as $g(n_i)$ (Alg. 1 Line 18), is upper-bounded by*

$$g(n_i) \leq d(e_1) + \dots + d(e_i) + \alpha \cdot i. \quad (3.12)$$

Proof. We prove Lemma 1 by induction on the node n_i with $i \in \{1, \dots, k\}$.

Base Case $n_i = n_1$: There are two cases:

1. After the execution of Alg. 1, n_1 's parent pointer points to n_0 . This implies that $g(n_1) = u(e_1) \leq d(e_1) + \alpha$ by Observation 1.
2. After the execution of Alg. 1, n_1 's parent pointer does not point to n_0 . This implies Alg. 1 finds a cheaper path to n_1 than following the edge e_1 . It follows that $g(n_1) \leq u(e_1) \leq d(e_1) + \alpha$. Thus, the base case holds.

Inductive Hypothesis: Lemma 1 holds for all nodes $n \in \{n_1, \dots, n_{i-1}\}$.

Inductive Step: By Inductive Hypothesis, it follows that after the execution of Alg. 1, we have

$$g(n_{i-1}) \leq d(e_1) + \dots + d(e_{i-1}) + \alpha \cdot (i - 1). \quad (3.13)$$

We again proceed with the same case analysis.

1. After the execution of Alg. 1, n_i 's parent pointer points to n_{i-1} . This implies that $g(n_i) = g(n_{i-1}) + u(e_i)$. By Eq. 3.13 and Observation 1, we have

$$\begin{aligned}
g(n_i) &= g(n_{i-1}) + u(e_i) \\
&\leq d(e_1) + \dots + d(e_{i-1}) + \alpha \cdot (i-1) + d(e_i) + \alpha \\
&= d(e_1) + \dots + d(e_i) + \alpha \cdot i.
\end{aligned} \tag{3.14}$$

2. After the execution of Alg. 1, n_i 's parent pointer does not point to n_{i-1} . This implies Alg. 1 finds a cheaper path to n_i than following the edge e_i from node n_{i-1} . Consequently, $g(n_i) \leq g(n_{i-1}) + u(e_i)$. Applying the same reasoning from Eq. 3.14, we have

$$g(n_i) \leq d(e_1) + \dots + d(e_i) + \alpha \cdot i. \tag{3.15}$$

Thus, by induction, Lemma 1 holds. $\square$

With Lemma 1 established, note that $n_k = n_{\text{goal}}$. This implies that after its execution, Alg. 1 is guaranteed to find a path to n_{goal} that is cheaper than $d(e_1) + \dots + d(e_k) + \alpha \cdot k$. Let $\text{Alg}(I)$ denote the path returned by Alg. 1 on instance I , it follows that

$$\begin{aligned}
u(\text{Alg}(I)) &\leq d(e_1) + \dots + d(e_k) + \alpha \cdot k \\
&= d(\text{OPT}(I)) + \alpha \cdot k.
\end{aligned} \tag{3.16}$$

Recall that k is the number of edges in the optimal path $\text{OPT}(I)$, and each edge in Alg. 1 is an action primitive π . In this work, all distances $d(\cdot)$ are measure in meters. Let $l_{\min}$ denotes the distance of the shortest action primitive in meters. This implies that $k \leq \frac{d(\text{OPT}(I))}{l_{\min}}$. By expanding $u(\cdot)$ using its definition from Eq. 3.3, it follows that

$$\begin{aligned}
&d(\text{OPT}(I)) + \alpha \cdot k \\
&\leq d(\text{OPT}(I)) + \alpha \cdot k + \alpha \cdot C(\text{OPT}(I)) \\
&\leq d(\text{OPT}(I)) + \frac{\alpha}{l_{\min}} \cdot d(\text{OPT}(I)) + \alpha \cdot C(\text{OPT}(I)) \\
&= \left(1 + \frac{\alpha}{l_{\min}}\right) \cdot d(\text{OPT}(I)) + \alpha \cdot C(\text{OPT}(I)) \\
&\leq \left(1 + \frac{\alpha}{l_{\min}}\right) \cdot (d(\text{OPT}(I)) + \alpha \cdot C(\text{OPT}(I))) \\
&= \left(1 + \frac{\alpha}{l_{\min}}\right) \cdot u(\text{OPT}(I)).
\end{aligned} \tag{3.17}$$

It follows that Theorem 1 holds.

3.7 Evaluation

We evaluate the performance of the proposed planner both in simulation and in a physical dry-land ship-ice navigation testbed. We compare our planner (*predictive*) against three baselines. The first baseline (*straight*) is a naive planner that returns a straight path from the ship’s position to the goal. The second baseline (*skeleton*) adopts the shortest open-water planner proposed in [27]. The third baseline (*lattice*) is the lattice planner proposed in [14]. Each trial involves navigating the ship across an ice channel toward a goal line $\mathcal{G}$ using a specified planner. Codes and demos are available at <https://github.com/IvanIZ/predictive-asv-planner>.

3.7.1 Simulation Setup

We conduct evaluations on the autonomous ship-ice navigation simulator from [14]. The setup is consistent with the experimental platform at the National Research Council Canada ice tank in St. John’s, NL, featuring a 1:45 scale model vessel. Tests are conducted at 20%, 30%, 40%, and 50% concentrations, with 200 randomly generated environments per concentration. Fig. 3.1 (left) shows a 40% concentration ice field. We evaluate all four planners on the same set of trials, giving a total of 4 concentrations $\times$ 200 trials $\times$ 4 planners = 3200 experiments.

Planners are evaluated with four metrics - travel distance, a running total kinetic energy loss by the ship, impulse due to ship-ice collisions, and an approximate of work done by the ship due to ice pushing. The total kinetic energy loss and impulse are computed from the simulator, and the approximated work W_{approx} is computed as an aggregation of products of the ice mass m_{p_i} with the arc length s_{p_i} taken by the ice floe $p_i \in W_{\text{obs}}$ as $W_{\text{approx}} = \sum_{p_i \in W_{\text{obs}}} m_{p_i} s_{p_i}$. While W_{approx} does not directly compute work, we found that it correlates with work and provides a direct assessment of the extent of ice field changes due to ship maneuvers.

3.7.2 Simulation Results

The evaluation results from simulation are presented in Fig. 3.4. As shown, the *predictive* planner gives the best performance across all collision metrics in all concentrations.

In general, the *predictive* planner and the *lattice* are the most competitive planners in collision minimization, especially in 20% and 30% concentrations. This could potentially

be explained by the fact that both *predictive* and *lattice* planners are designed to handle ship-ice navigation scenarios where a collision-free path does not exist.

Observe that while the collision minimization performances of *predictive* and *lattice* planner are comparably competitive in low ice concentrations (20% and 30%), the performance gain from *skeleton* and *lattice* planners over the naive *straight* planner diminishes significantly as concentration increases (40% and 50%). Since *skeleton* and *lattice* planner do not account for ice floe motion during planning, the increasingly rich and complex ice motion in higher concentrations renders the planning strategies less effective. However, the *predictive* planner maintains a robust performance across all concentrations.

Lastly, note that *predictive* planner outperforms *lattice* while maintaining comparable distances. This suggests that the improved collision avoidance of the *predictive* planner is not simply due to compromising distance by taking longer detours, but stems from a more informed reasoning process enabled by the additional knowledge about ice motion.

3.7.3 Physical Testbed Setup

We further validate the proposed planner in a physical testbed that simulates autonomous ship-ice navigation at the University of Waterloo Autonomous System Laboratory. The testbed features a $2.8\text{ m} \times 1.6\text{ m}$ navigation environment, shown in Fig. 3.1 (right). A TurtleBot3 Burger is used as a model vessel, and is attached with V-shape hanger as the ship bow. Ice obstacles are made from randomly shaped foam boards. The vision system is an overhead camera that tracks the robot pose at 10 Hz and detects obstacles at 3 Hz. The robot navigates at a constant nominal velocity of 0.03 m/s, with a PID controller tracking the reference path at 10 Hz.

3.7.4 Physical Testbed Evaluation

We perform testbed evaluations in 20%, 30%, and 40% concentration environments. For each concentration, 40 trials are performed for each planner, giving a total of 3 concentrations $\times$ 40 trials $\times$ 4 planners = 480 experiments. W_{approx} and travel distance are used as metrics to quantify collision minimization and travel efficiency, respectively. Demonstrations of the testbed setup and experiments can be found in the supplemental video.

Fig. 3.5 presents the evaluation results from the testbed, showing similar trends. Notably, the *predictive* planner achieves the lowest W_{approx} across all three concentrations while maintaining comparable travel distances in the 20% and 30% cases. However, a

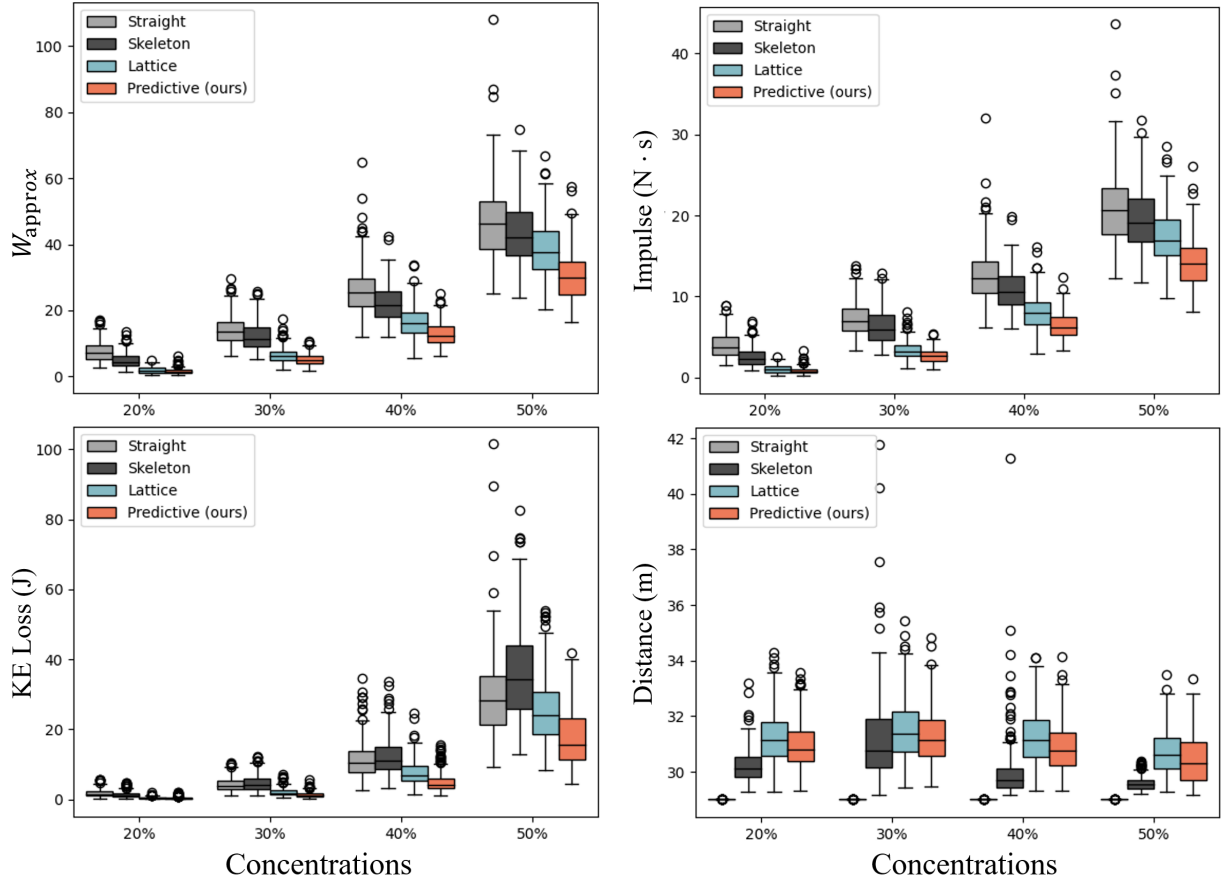


Figure 3.4: Evaluation results from simulation across four concentrations

trend of increased travel distance from *predictive* is observed at the 40% concentration, likely due to greater errors in ice motion prediction.

We acknowledge the lab-to-real gap in our testbed, particularly the absence of factors such as ocean current and fluid dynamics. However, the testbed effectively captures core aspects of autonomous navigation in ice-covered waters, focusing on scenarios where a collision-free path is unattainable and where ship-ice and ice-ice collisions are prominent.

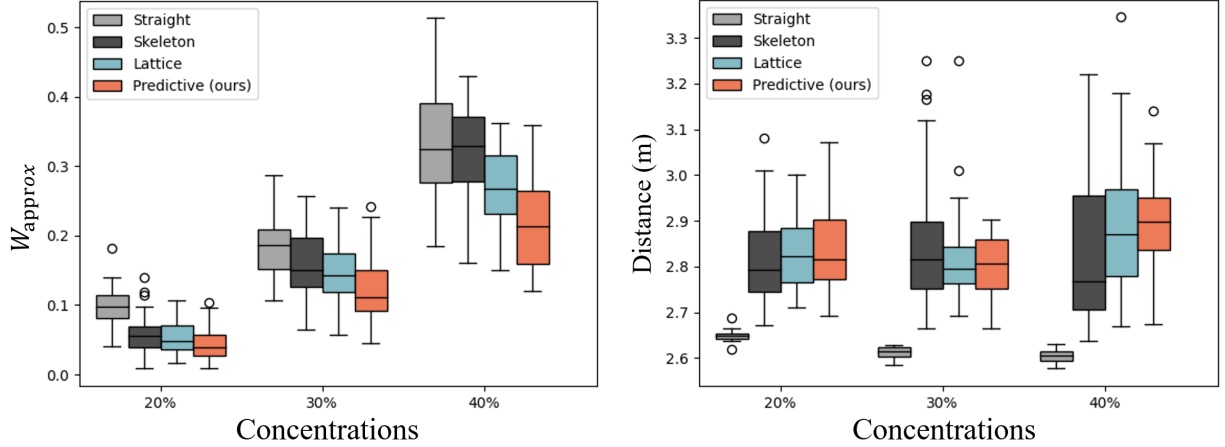


Figure 3.5: Testbed evaluations across 20%, 30%, and 40% concentrations.

3.8 Conclusion

We present an integrated framework for ASV navigation in ice-covered waters, combining a deep learning model for ice motion prediction with a lattice planner to plan obstacle-motion-aware paths. Evaluations in simulation and a physical testbed show that predicting the motion of ice significantly improves the navigation performance. The performance gain is especially notable in high ice concentrations with increased ice motion, underscoring the value of anticipating obstacle motions during planning. In future work, we aim to incorporate additional environment factors into our prediction, such ship waves and ocean currents, for more robust inferences.

Chapter 4

Benchmarking Non-prehensile Interactive Navigation

In this chapter, we move from ASV navigation to a boarder scope of non-prehensile interactive navigation and introduce Bench-NPIN, a unified benchmarking suite that evaluates algorithms across different environments and tasks.

4.1 Motivation

When mobile robots are deployed in cluttered and unstructured environments, such as homes, hospitals, or disaster sites, the ability for robots to interact with objects and obstacles can be essential for task completion. Traditional approaches to robot navigation focus on computing collision-free paths. However, in environments where obstacles block all feasible routes, robots must interact with obstacles to progress, leading to a class of problems known as Navigation Among Movable Obstacles (NAMO) [20]. Interactive navigation expands upon NAMO by including scenarios where interactions with objects in the environment are not only inevitable but essential for task completion, for example, pushing objects to designated locations while avoiding unnecessary collisions.

Historically, interactive navigation and NAMO have been studied with the assumption that mobile robots are equipped with manipulators to move obstacles out of the way [74, 64, 75, 82]. However, manipulators are not always viable due to cost considerations, structural limitations, or task-specific constraints. Moreover, prehensile manipulation of objects with complex dynamics remains a significant challenge. This has spurred increasing interest

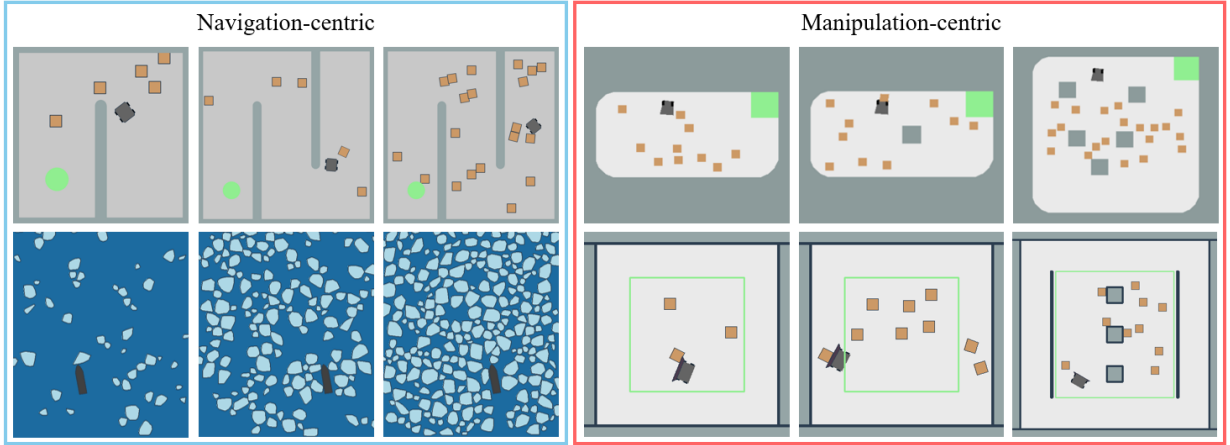


Figure 4.1: Illustration of non-prehensile interactive navigation environments with configurable complexity levels. Upper-left shows the *Maze* environment, the mobile robot must reach the goal point among movable obstacles. Lower-left shows the *Ship-Ice* environment where an autonomous ship must navigate through ice-covered waters. Upper-right shows *Box-Delivery* environment, where the robot’s objective is to push all boxes into the green receptacle. Lower-right is the *Area-Clearing* environment, where a robot needs to remove all boxes from within a clearance area outlined by the green rectangle.

in non-prehensile interactive navigation (NPIN), where robots interact with objects with non-grasping actions [12, 91, 83, 21]. Despite the growing body of work, the field still lacks a unified framework for systematic evaluation and reproducibility, as no standardized benchmarking tools exist.

To address this gap, we present Bench-NPIN, the first comprehensive suite of tools for standardized training and evaluation of algorithms for non-prehensile interactive navigation. Bench-NPIN is made available as an open-source Python library. In this benchmark, we define two classes of non-prehensile interactive navigation tasks. *Navigation-centric NPIN* refers to the tasks where the objective is to navigate the robot to a goal location; *Manipulation-centric NPIN* refers to the tasks where the primary objective is to manipulate movable objects using a non-prehensile manipulator (e.g., pushed using the robot’s front bumper) to reach a desired goal configuration.

Our key contributions are as follows. First, we introduce Bench-NPIN, a benchmarking tool, that consists of a range of simulated environments for both navigation-centric and manipulation-centric NPIN tasks. Each environment is user-configurable to modify the difficulty of the environments and the tasks. The included environments are as follows:

- navigating a maze among movable obstacles,
- autonomous ship navigation in ice-covered waters,
- delivering boxes to a receptacle, and
- clearing boxes from an area.

Secondly, we propose a set of novel metrics that are included in Bench-NPIN to evaluate and compare policies that tackle navigation-centric and manipulation-centric NPIN tasks. The proposed metrics capture properties unique to interactive navigation such as efficiency and interaction trade-offs, and partial task completions.

Finally, we demonstrate Bench-NPIN using established baseline algorithms for each task and present their evaluation results. The baseline implementations are included in Bench-NPIN as reference implementations for users to configure algorithms of their choosing. For our baselines, we include implementations of widely used Reinforcement Learning (RL) algorithms, such as Soft Actor-Critic (SAC) [33] and Proximal Policy Optimization (PPO) [70]. In addition to general RL baselines, we also include task-specific state-of-the-art methods, such as ship path planners [14, 95] for autonomous ship navigation and Spatial Action Maps [83] for manipulation-centric tasks.

4.2 Related Work

In this section, we review related work in NPIN and the most closely related benchmarks.

4.2.1 Non-prehensile Interactive Navigation

Interactive navigation [86] refers to scenarios where a mobile robot must interact with the environment to complete its tasks. Non-prehensile interactive navigation (NPIN) is a subset of this problem, where robots interact with obstacles and objects without using manipulators or grasping actions. NPIN tasks can be further classified into two categories: navigation-centric NPIN and manipulation-centric NPIN.

Navigation-centric NPIN focuses on reaching a goal location while engaging in inevitable environment interactions, such as pushing path-blocking obstacles aside. Prior work has

explored various strategies to tackle this challenge. For instance, [91] investigated reinforcement learning-based navigation in cluttered environments with movable obstacles, proposing a local path-planning framework for dynamic interactions. [12] introduced an adaptive dynamics model for common indoor movable objects that enables robots to anticipate and adjust to object movements. Beyond terrestrial robots, [14, 95] studied autonomous surface vehicle (ASV) navigation in ice-covered waters, where ship-ice interactions are inevitable, and proposed planning frameworks that reduce collisions during navigation.

Manipulation-centric NPIN, on the other hand, emphasizes manipulating the movable objects in the environment to reach a desired configuration. Examples include pushing objects to predefined locations or clearing debris from a designated area. In [83], Spatial Action Map (SAM) is proposed as a novel action representation for a multi-object pushing problem, where a mobile robot must transport scattered objects into a goal receptacle. This work was subsequently extended in [85], where the robot uses an onboard pneumatic blower to clear up scattered objects. In [21], a Deep Reinforcement Learning (DRL) framework with task-space decomposition is proposed to address long-distance non-prehensile object delivery. In [22], curriculum reinforcement learning is deployed to learn multi-robot non-prehensile heavy object transportation.

4.2.2 Benchmarks for Robot Navigation

Various benchmarks have been proposed for mobile robot navigation. Traditional benchmarks focus primarily on collision-free navigation. [62] proposed a benchmark for ground mobile robot navigation in cluttered environments. [88] and [50] extend beyond static environments by considering both static and dynamic obstacles. [39] introduced SOCIAL-GYM for robots navigating dynamic human environments. However, these benchmarks primarily assess navigation performance under constraints that discourage environmental interaction, classifying collisions as failures rather than controlled contacts.

The Interactive Gibson Benchmark [86] represents an advancement in benchmarking interactive navigation, introducing scenarios where environmental interaction is necessary. Our work differs from [86] by focusing on non-prehensile interaction navigation. Further, we extend beyond navigation-centric scenarios considered in [86] by covering a broader range of tasks, including manipulation-centric scenarios such as non-prehensile object delivery via pushing.

Evaluation metrics in existing navigation benchmarks [62, 88] primarily focus on navigation success rates, collision count, and path efficiency. However, these metrics are

insufficient for assessing interactive navigation tasks, where controlled interactions are required rather than avoided. [86] proposed a novel set of metrics for interactive navigation that captures both efficiency and environment disturbance from the interaction. These metrics are geared toward navigation-centric NPIN, and are not suitable for manipulation-centric cases where modifying the environment is part of the task (i.e., box delivery). In contrast, our benchmark introduces separate evaluation metrics for navigation-centric and manipulation-centric NPIN, ensuring that interaction quality and efficiency are appropriately measured in both classes.

4.3 Bench-NPIN

Bench-NPIN consists of three main components – (i) environment, (ii) policy, and (iii) evaluation metrics. In this section, we detail the specifics of these environments, explain the implementation of custom policies in Bench-NPIN and introduce the metrics used to evaluate policies. First, we clarify some of the terminology used in this section.

Environment: An environment is a custom-built Gymnasium [80] interface, which contains the full world transition dynamics. Each environment has multiple variations with increasing complexity (see Fig. 4.1). Environments can be run independent of a training session, and the robot can be manually operated by a user.

Task: We refer to a task as the robot’s objective in a given environment. Specifically, we consider two types of tasks:

- *Navigation-centric*, where the robot must reach a specified goal region in the environment.
- *Manipulation-centric*, where the robot must move objects in the environment to specified areas.

Policy: A policy is a set of rules used to control the robot in an environment. It takes a sequence of observations as an input and outputs a robot action. In Bench-NPIN, we include several baseline policies as reference implementations to users, which are listed in Tables 4.1 and 4.2.

Metric: A metric is a score used to evaluate the effectiveness of a policy in completing a task in a given environment. In Section 4.3.3, we introduce novel metrics that we included in Bench-NPIN to evaluate and compare policies.

Table 4.1: Non-prehensile interactive navigation tasks in Bench-NPIN.

Environments	Task Class	Variations
<i>Maze</i>	Navigation-centric	maze complexity, obstacle count
<i>Ship-Ice</i>	Navigation-centric	ice concentrations
<i>Box-Delivery</i>	Manipulation-centric	w/o static obstacles, box count
<i>Area-Clearing</i>	Manipulation-centric	w/o static obstacles, box count

Table 4.2: Reference baselines for each task in Bench-NPIN.

Environments	Baselines Evaluated
<i>Maze</i>	SAC [33], PPO [70]
<i>Ship-Ice</i>	SAC [33], PPO [70], ASV planning [14, 95]
<i>Box-Delivery</i>	SAC [33], PPO [70], SAM [83]
<i>Area-Clearing</i>	SAC [33], PPO [70], SAM [83], GTSP (Sec. 4.4.1)

Table 4.3: Actions and observations for each task.

Environments	Action	Observation
<i>Maze</i>	Angular vel.	occupancy + footprint + goal DT
<i>Ship-Ice</i>	Angular vel.	occupancy + footprint + goal DT + heading encoding
<i>Box-Delivery</i>	Heading	occupancy + footprint + egocentric DT + goal DT
<i>Area-Clearing</i>	Heading	occupancy + footprint + egocentric DT + goal DT

Table 4.4: Rewards each task.

Environments	Rewards
<i>Maze</i>	collision + distance decrement + terminal
<i>Ship-Ice</i>	collision + heading + terminal
<i>Box-Delivery</i>	collision + box completion + box displacement
<i>Area-Clearing</i>	collision + box completion + box displacement

4.3.1 Bench-NPIN Environments

We now introduce the navigation-centric and manipulation-centric environments and tasks included in Bench-NPIN. For each environment, we provide details about the actions and observations available to the robot and the rewards the robot receives. Tables 4.3 and 4.4 summarize the action space, observation space, and reward functions for each environment.

Navigating maze with movable obstacles (*Maze*)

The *Maze* environment features a static maze structure with randomly initialized obstacles. The robot’s task is to navigate from a starting position to a goal location (green circle in Fig. 4.2) while minimizing path length and obstacle collisions. Variations of the environment include different maze structures, obstacle geometries, and obstacle locations and densities.

The robot in this environment is configured to move with a constant forward speed and receives an angular velocity as action input from the policy. At each timestep, the robot makes a local egocentric observation consisting of four channels that correspond to (i) static obstacle occupancy, (ii) movable obstacle occupancy, (iii) robot footprint, and (iv) a distance transform (DT) originating from the goal (goal DT). Fig. 4.2 shows an example of the robot’s observations. The robot receives rewards for moving closer to the goal, penalties for colliding with obstacles, and a terminal reward when reaching the goal.

Autonomous ship navigation in icy waters (*Ship-Ice*)

The *Ship-Ice* environment simulates an autonomous ship navigating in a channel of ice-covered waters (see Fig. 4.1). The ship’s task is to reach a horizontal goal line ahead of the ship while minimizing collisions with broken ice floes in the channel [14]. The ice floes

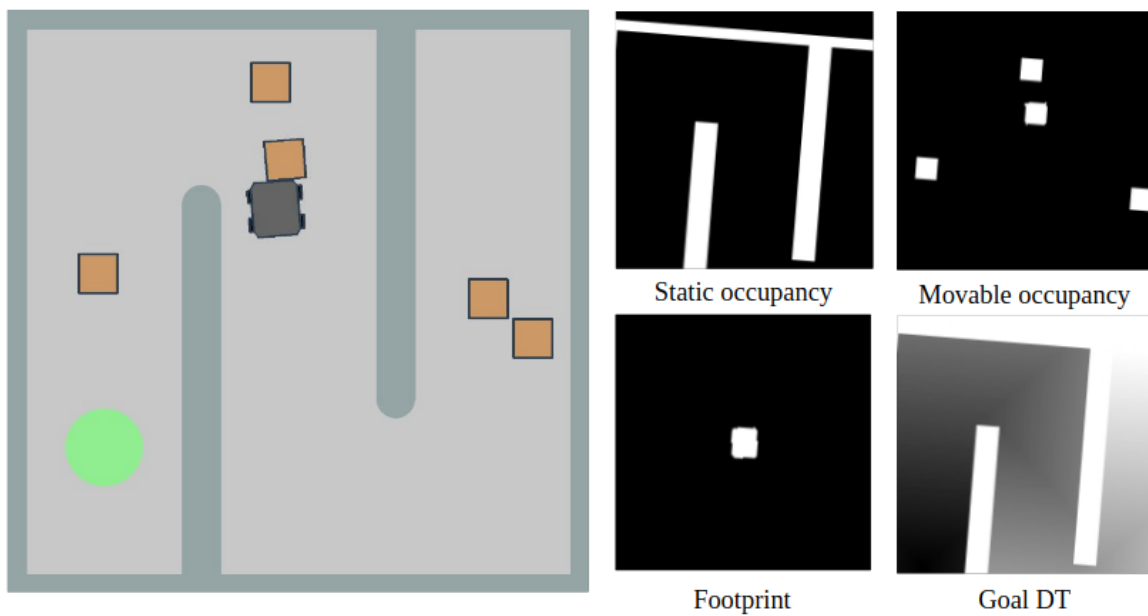


Figure 4.2: The *Maze* environment (left) and an example egocentric observation for *Maze* (right).

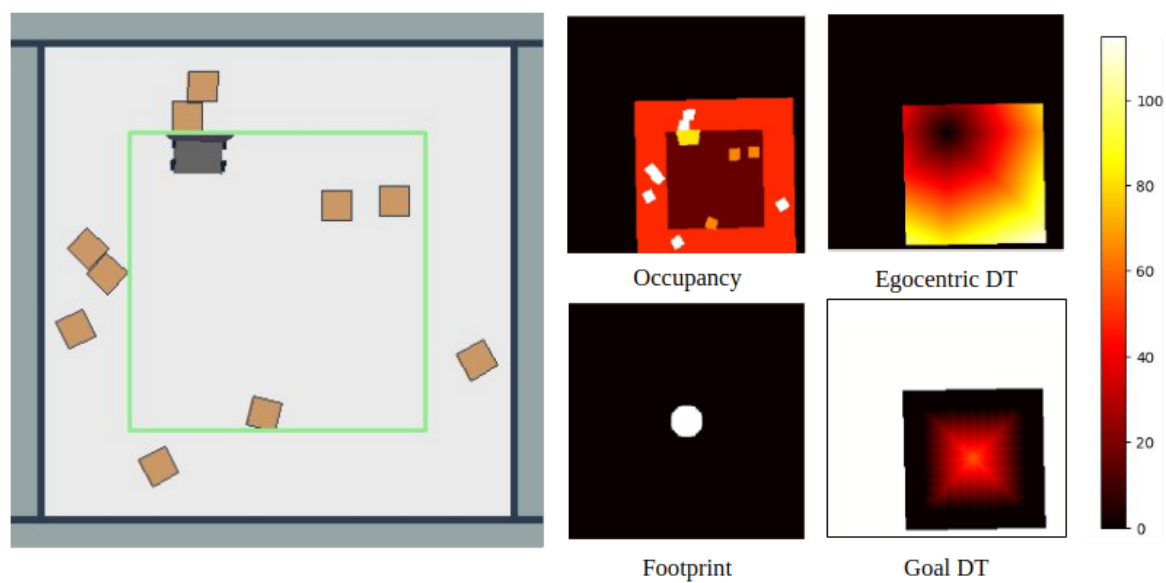


Figure 4.3: The *Area-Clearing* environment (left) and example observation for *Area-Clearing* (right).

are represented as convex polygons with ice concentrations varying between 0% and 50%. As concentrations exceed approximately 20%, collision-free paths become non-existent, so the ship must minimize kinetic energy loss and maintain momentum.

Similar to the *Maze* environment, the ship moves with a constant forward speed and receives an angular velocity as action input. The ship makes the observations similar to *Maze* but with an additional channel that encodes the heading of the ship as a single-pixel-width line. This channel facilitates fine-grained movement around clusters of ice floes. The ship’s reward function combines penalties for collisions with ice floes, a heading reward to encourage approaching the goal line, and a terminal reward for reaching the goal.

Delivering boxes to a receptacle (Box-Delivery)

The *Box-Delivery* environment consists of a set of movable boxes and a designated *receptacle*. The robot is tasked with delivering all boxes to the receptacle using a non-prehensile manipulator (e.g. front bumper) to push the boxes. The boxes and robot starting location are randomly generated within an environment. Further variations are possible by including static obstacles (e.g., columns) and changing the number and size of movable objects.

At each step, the robot receives a heading as its action input, in which case the robot travels for a fixed distance along that heading. The robot’s observation is egocentric and includes (i) occupancy of the environment with static and movable objects encoded in different values, (ii) robot footprint, (iii) DT originating from the robot (egocentric DT), and (iv) DT originating from the receptacle (goal DT). The robot receives rewards for moving boxes toward the receptacle or delivering a box, and penalties when colliding with static obstacles or moving boxes away from the receptacle.

Clearing boxes from an area (Area-Clearing)

The *Area-Clearing* environment consists of a set of movable boxes and a *clearance area*. The task of the robot is to remove the boxes from within this clearance area, with no constraints on the final positions of the cleared boxes. Variations of this environment can be created in Bench-NPIN by changing the number of objects to be removed, the number of static obstacles, and workspace constraints (e.g., walls blocking clearance area close to the boundary).

The robot actions and observations in *Area-Clearing* are identical to those of *Box-Delivery*. Fig. 4.3 illustrates the environment and corresponding robot observations. The

robot (grey) pushes movable objects (brown) out from within a clearance area (inside light green square). In the occupancy observation, cleared boxes are shown in a distinct color from those still inside the area, visually indicating task progress. The reward function is constructed similar to the *Box-Delivery*, where the robot receives a reward when moving any object inside the clearance area closer to the boundary, and when pushing a box out of the boundary.

4.3.2 Bench-NPIN Implementation

Bench-NPIN is designed with a modular architecture to support flexible customization and ease of use. The benchmark has been designed with a one-line install to allow researchers to get up-and-running quickly. We highlight two core aspects of Bench-NPIN’s implementation, which enable users to integrate custom policies of their choice to tackle the discussed NPIN tasks.

Gymnasium Integration

All environments detailed in Sec. 4.3.1 are implemented on top of the Pymunk 2D [6] physics engine. We integrate these environments within the Gymnasium [80] interface to standardize policy development and evaluation. The implementation supports a wide range of environment parameters through either script-level configuration or command-line arguments.

Extensible Policy Class

Bench-NPIN provides a standardized, extensible policy template to simplify the incorporation of new algorithms. This template follows a plug-and-play philosophy: a user only needs to implement the required APIs within a policy class. Bench-NPIN then handles inference, evaluation, and logging processes.

By way of illustration, Bench-NPIN comes with reference implementations for well-established baselines, as described in Sec. 4.4. These examples demonstrate both how to integrate popular Reinforcement Learning algorithms from Stable Baselines 3 [65] and how to incorporate specialized or state-of-the-art approaches. Additional documentation and usage examples, including configuration files and training scripts, are provided in the project repository at <https://github.com/IvanIZ/BenchNPIN>.

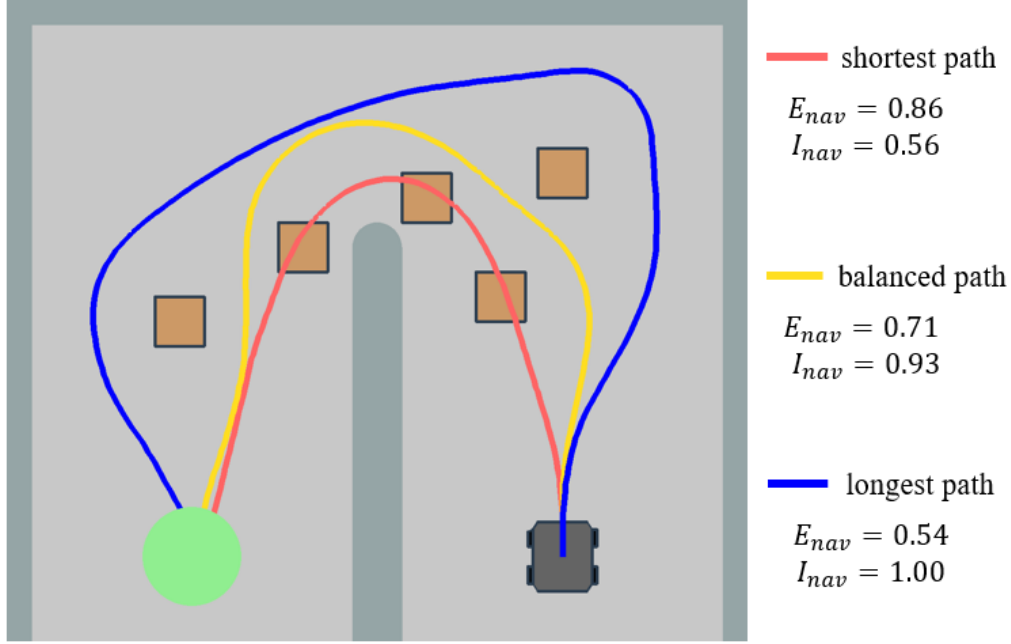


Figure 4.4: An illustration of three teleoperated paths and the corresponding performance. While the shortest path (red) has the highest efficiency score $E_{nav} = 0.86$, the excessive collisions with the movable obstacles significantly degrade the effort score $I_{nav} = 0.56$. In contrast, while the longest path (blue) is collision-free $I_{nav} = 1.00$, efficiency is largely compromised $E_{nav} = 0.54$. A balanced path (yellow) potentially offers the best trade-off.

4.3.3 Metrics

We now provide a set of novel metrics to evaluate policies for NPIN tasks. These metrics are inspired by [86] and aim to capture the efficiency and interaction efforts of the robot. Let K be the number of movable objects in the environment. The mass and motion distance for each object are denoted as m_i and l_i for $i \in \{1, \dots, K\}$, respectively. We denote the mass and motion distance of the robot as m_0 and l_0 , respectively. Since navigation-centric and manipulation-centric NPIN tasks are intrinsically different, Bench-NPIN provides two sets of metrics to address each class separately.

Navigation-centric Metrics

Efficiency in navigation-centric NPIN is computed with respect to the length of the robot’s path. Let $\mathbb{1}_{\text{success}}$ be an indicator function on whether the robot successfully reaches the goal, and l_0^* be the shortest path distance from the robot’s start position to the goal while ignoring interactions with movable objects. The efficiency score $E_{\text{nav}} \in [0, 1]$ is then computed as

$$E_{\text{nav}} = \mathbb{1}_{\text{success}} \frac{l_0^*}{l_0}. \quad (4.1)$$

Eq. 4.1 implies that the efficiency score $E_{\text{nav}} = 1$ if the robot reaches the goal with the shortest travel distance.

The interaction effort score measures the kinematic effort exerted by the robot to move objects along the path that it had traversed. This score is given by the ratio between the approximated minimum work and the approximated actual work done by the robot from overcoming kinetic frictional forces due to interactions in its path. To approximate this work, we assume that the entire environment has a kinetic friction coefficient μ . Following this, the kinetic frictional force for moving the i th object is $\mu m_i g$ and the work done by the robot to overcome this force (i.e., move this object) is $\mu m_i g l_i$. We further approximate the minimal work required by considering the robot as the 0th object, giving $\mu m_0 g l_0$. Thus, the interaction effort score $I_{\text{nav}} \in [0, 1]$ is computed as

$$I_{\text{nav}} = \frac{\mu m_0 g l_0}{\sum_{i=0}^K \mu m_i g l_i} = \frac{m_0 l_0}{\sum_{i=0}^K m_i l_i}. \quad (4.2)$$

Eq. 4.2 implies that the maximum interaction effort score is reached when the robot manages to reach the goal without moving any objects. Further, I_{nav} reduces when interacting with heavier objects at greater distances, an undesirable behavior in navigation-centric tasks. Fig. 4.4 illustrates how our metrics capture both efficiency and interaction effort. This visualization also highlights the inherent and unique trade-off in interactive navigation, where policies need to consider balancing both minimizing travel distance and avoiding unnecessary interactions.

Manipulation-centric Metrics

The metrics for manipulation-centric NPIN, unlike the navigation-centric NPIN metrics, must incorporate the minimum path length and effort required by the robot to manipulate objects towards completing its task. We also look to compare robot paths that may achieve

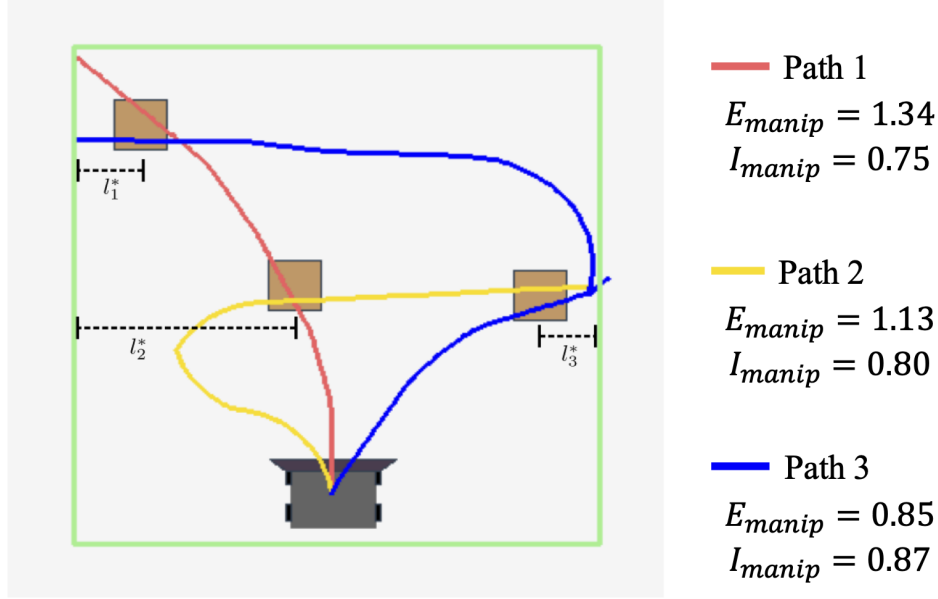


Figure 4.5: Three teleoperated paths for area clearing, with each path having a task success score $S_{manip} = 2/3$. While path 1 (red) has the highest efficiency score, its effort score is low as it moves a box longer than its shortest distance to the goal (dotted lines of length l_i^*). In contrast, path 3 (blue) achieves the best effort score by pushing as little as possible, but results in a long path and a low efficiency score. Path 2 (yellow) potentially offers the best trade-off.

partial completion of tasks, e.g., push two out of four boxes into the receptacle. To capture this effectively, we propose three metrics: (i) task success score S_{manip} , (ii) efficiency score E_{manip} , and (iii) interaction effort score I_{manip} .

To simplify our explanation, we consider that each of the K movable objects must be manipulated to a desired configuration (delivered or cleared). As such, the robot must complete a series of smaller manipulation *sub-tasks* corresponding to each box to complete the overall task. Let $\mathbb{1}_{success}^i$ be an indicator function that denotes whether the sub-task corresponding to the i^{th} movable object has been completed. Let $K' \leq K$ be the number of sub-tasks that have been completed successfully, i.e., $K' = \sum_{i=1}^K \mathbb{1}_{success}^i$. The task success score S_{manip} is given as

$$S_{manip} = \frac{K'}{K}. \quad (4.3)$$

Now, given that K' sub-tasks were completed, let $L^*(K')$ be the minimum path length required to complete the successful sub-tasks. For example, if the robot delivers 2 out of 4 boxes, $L^*(K')$ is the minimum path length the robot must take to move the two boxes to the receptacle. The efficiency score E_{manip} is given by

$$E_{\text{manip}} = \frac{L^*(K')}{l_0}. \quad (4.4)$$

In Bench-NPIN, we compute $L^*(K')$ using a minimum spanning tree (MST) of a graph G that we construct as follows. The vertex set includes K' vertices for the initial positions of the movable objects, another K' vertices for the closest point to the goal (receptacle or clearance area boundary) from each object, and a vertex for the robot. Edges are added between all pairs of objects, from the robot to each object, and from each object to its nearest goal, with the edge weights being the shortest distance between the two points. The MST of this graph is not a strict lower bound on the total path length, and as such, it is not guaranteed that $E_{\text{manip}} \in [0, 1]$. However, we find that this is a practical bound, as doubling the edges of the MST provides a path that can complete the sub-tasks (ignoring object-to-object interactions). Using both S_{manip} and E_{manip} , we can determine how many sub-tasks the robot has completed, and how efficient its path was to complete the sub-tasks.

For the interaction effort score, we look to approximate the minimum work the robot must do to complete K' sub-tasks while traversing its path. Using the friction coefficient μ of the environment, this work is approximated as $\mu g(m_0 l_0 + \sum_{i=1}^K \mathbb{1}_{\text{success}}^i m_i l_i^*)$, where l_i^* is the shortest distance from the i^{th} object to the goal (see Fig. 4.5). The actual work done by the robot is approximated as $\sum_{i=0}^K \mu m_i g l_i$, i.e., similar to the one in I_{nav} . Using this, the interaction effort score $I_{\text{manip}} \in [0, 1]$ is computed as

$$I_{\text{manip}} = \frac{m_0 l_0 + \sum_{i=1}^K \mathbb{1}_{\text{success}}^i m_i l_i^*}{\sum_{i=0}^K m_i l_i}. \quad (4.5)$$

Fig. 4.5 illustrates the manipulation-centric NPIN metrics capturing the efficiency and interaction effort for three trajectories for the area clearing task. All trajectories achieve a task success score of 2/3, but with different trade-offs for path length and interaction effort.

4.4 Evaluating Baselines using Bench-NPIN

In this section, we present experimental results for all interactive navigation tasks. These results are intended to confirm the extensibility of Bench-NPIN and provide benchmarking

references for future studies.

4.4.1 Implemented Baselines

To support both general NPIN research and task-specific studies, Bench-NPIN includes a diverse set of baselines. These baselines fall into two categories: (i) reinforcement learning baselines applicable to all tasks outlined in Section 4.3.1 and (ii) task-specific baselines for each task. The baselines for each task are summarized in Table 4.1. For each task, we test the corresponding baselines using a single configuration of the environment and evaluate their performance using the metrics from Section 4.3.3.

The reinforcement learning baselines include Soft Actor-Critic (SAC) [33] and Proximal Policy Optimization (PPO) [70]. These baselines were implemented using Stable Baselines 3 [65] and trained across all tasks. To handle image observations, both baselines employ a customized ResNet18 [35] backbone with the last linear layer removed as the high-dimensional observation encoder, followed by a 3-layer MLP to output actions or value estimates.

In addition to general RL baselines, Bench-NPIN provides specialized policies tailored for individual tasks. For *Ship-Ice*, we provide lattice-planning [14] and predictive-planning [95] policies designed for ASV navigation. For *Box-Delivery* and *Area-Clearing*, we adopt the Spatial Action Map (SAM) policy [83], which is a method designed for multi-object pushing [84]. The SAM policy learns to output a series of waypoints in the environment that enable the robot to complete a manipulation-centric task. To implement SAM, we also configured the *Box-Delivery* and *Area-Clearing* environments to receive waypoints in the environment as action inputs.

Additionally for *Area-Clearing*, we introduce a new baseline policy that plans a global path to clear all boxes by solving a generalized traveling salesman problem (GTSP). This GTSP policy is inspired by an approach for robot coverage path planning [66]. For each box in the environment, we consider the path to clear the box across each edge of the clearance area boundary. For example, in the case of a rectangular clearance boundary (Fig. 4.3), we consider 4 paths to clear each box across each of the 4 edges. We will refer to these as the box’s *clearance paths*. We construct an auxiliary graph $G = (V, E)$ where each box constitutes a set S and contains vertices $v \in V$ that each correspond to a clearance path considered for the specific box. The robot’s start position is encoded as a separate vertex. The edge cost between two vertices is the shortest path length to reach the start of one clearance path from the end of another (or the robot’s start position). Solving the GTSP on this graph gives us a robot path that aims to clear each box individually along

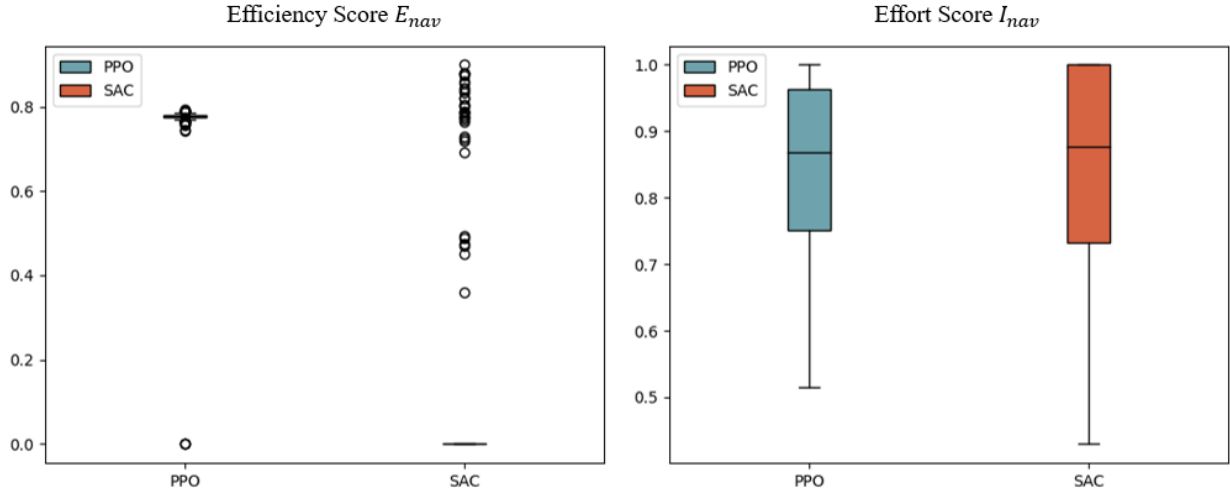


Figure 4.6: Evaluation results for PPO and SAC in *Maze*.

a series of clearance paths. The path is computed once and executed by the robot until it either (i) clears all boxes in the environment or (ii) finishes the full path. Note that this method does not necessarily clear all boxes and does not minimize the robot’s effort as it ignores object-to-object interactions.

Both SAC and PPO policies for all tasks are trained on a single 4GB NVIDIA GeForce RTX 3050 GPU for approximately 30 hours. For the manipulation-centric tasks, SAM was trained on NVIDIA RTX 6000 for approximately 7 hours. In each environment, we evaluated all baseline policies for 200 episodes. We fixed a constant random seed to evaluate each environment to ensure that all policies were tested against identical configurations.

4.4.2 *Maze* Evaluations

We evaluated the performance of SAC and PPO in a U-shaped maze with 5 movable obstacles, as shown in Fig. 4.4. The robot starts at the lower-right end of the maze and must reach the goal region located in the lower-left end. The obstacle placements are randomized at the start of each episode.

Fig. 4.6 shows the efficiency and interaction effort scores after evaluating PPO and SAC in *Maze*. Overall, PPO policy gives the best performance. The PPO policy achieves a mean efficiency of 0.76 and a mean effort score of 0.85. The SAC policy gives a comparable effort score (mean 0.84) to PPO (mean 0.85). However, it is important to note that SAC struggles

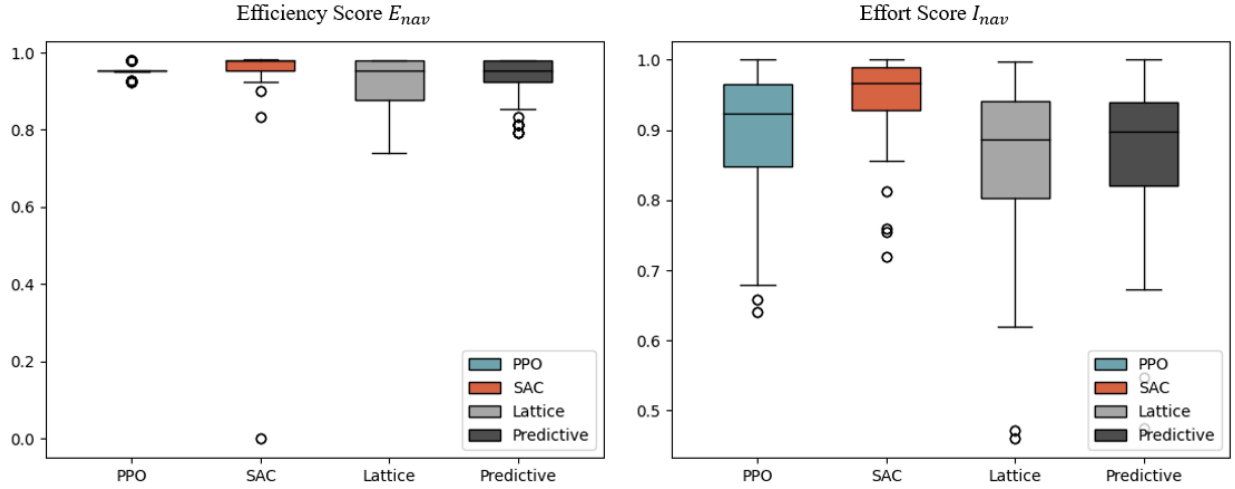


Figure 4.7: Evaluation results for PPO, SAC, lattice-planning policy (Lattice), and predictive-planning policy (Predictive) in *Ship-Ice*.

to navigate the robot toward the goal in most episodes. This challenge is reflected in the efficiency score, where the majority of episodes display efficiency values concentrated near zero, as indicated by a narrow interquartile range and a median close to zero. This explains the high effort score for SAC, as the robot did not have many opportunities to interact with the movable objects, as was the case with PPO.

4.4.3 *Ship-Ice* Evaluations

In *Ship-Ice*, the performance of SAC, PPO, lattice-planning policy, and predictive-planning policy are evaluated in 10% ice concentration with a 10-meter goal horizon. In each episode, the ice floes are randomly generated up to 10% concentration, and the ship is randomly initialized along a start line 10 meters away from the goal line.

Fig. 4.7 shows the evaluation results for PPO, SAC, lattice-planning policy, and predictive-planning policy. All baselines perform comparably in terms of efficiency with scores close to $E_{nav} = 1.0$. This is potentially due to the fact that evaluations are performed under a low ice concentration, so scenarios where long detours are required for avoiding large clusters of ice take place less frequently. In terms of collision avoidance, SAC gives the highest interaction effort scores, making it overall the best-performing model under 10% concentration setup.

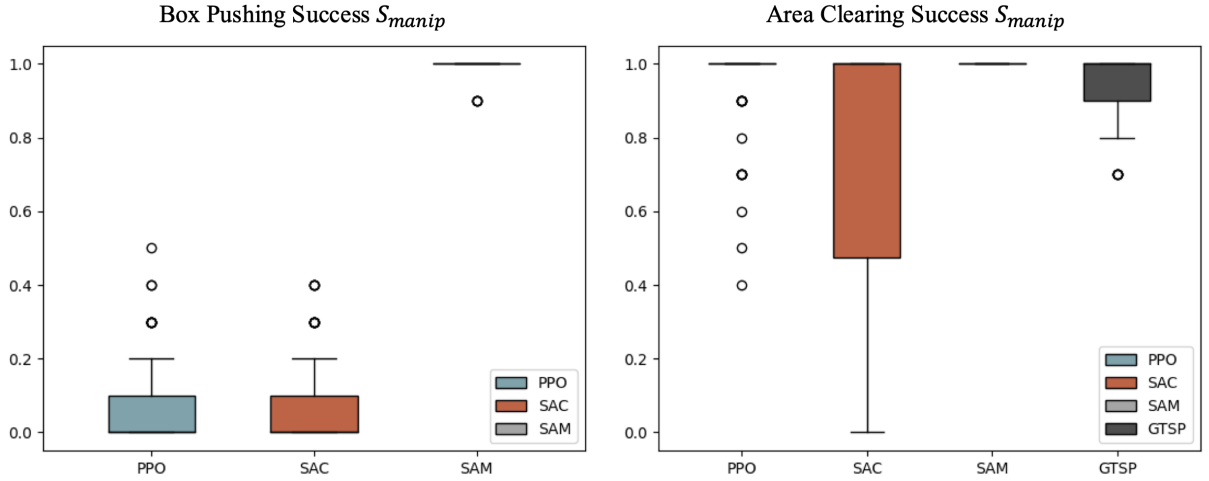


Figure 4.8: Task success scores S_{manip} for the manipulation-centric NPIN baselines included in the benchmark.

4.4.4 *Box-Delivery* Evaluations

For *Box-Delivery*, we evaluated the performance of PPO, SAC, and SAM under the setting shown in the left-most *Box-Delivery* snapshot in Fig. 4.1, where the robot must deliver 10 boxes to the receptacle with no static obstacles impeding its path. The positions of the robot and boxes are randomly initialized at the start of an episode. The episode terminates after all boxes have been delivered, or if the robot does not deliver any boxes to the receptacle for 200 steps.

The results of evaluating the baseline policies are shown in Figs. 4.8 and 4.9. We observe that SAM drastically outperforms the other baselines in terms of success and efficiency. This is mainly because SAM is a specialized algorithm for manipulation-centric tasks, whereas PPO and SAC are general RL algorithms. Another difference is that the SAM policy learns to output waypoints for the robot to drive to, whereas PPO and SAC learn to output a heading. As such, SAM is able to learn a more effective policy than PPO and SAC within the provided training time as the mapping from the high-dimensional observation space to a series of waypoints is less complicated to learn than mapping to a series of headings.

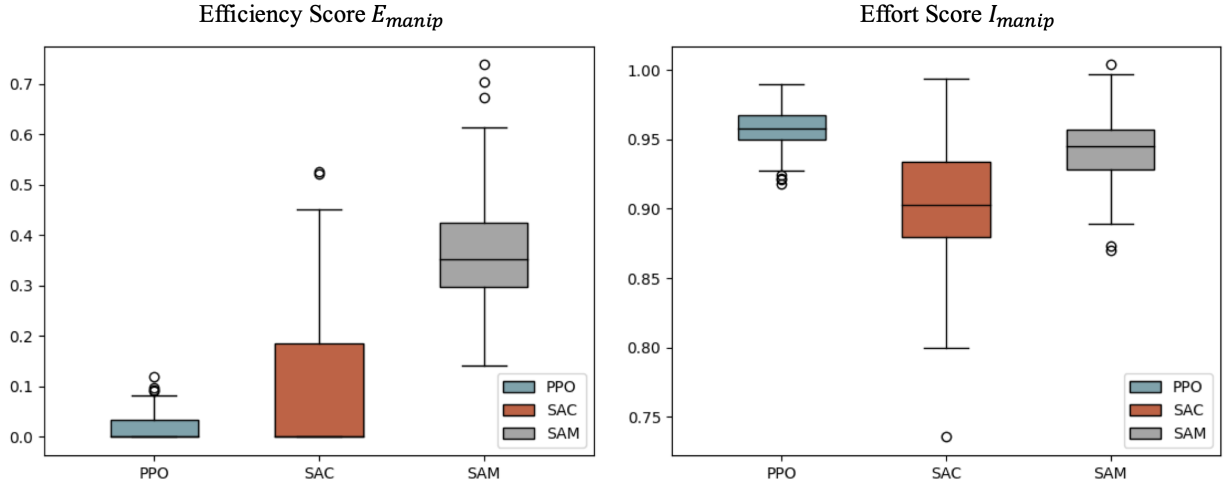


Figure 4.9: Evaluation results for PPO, SAC, and SAM in *Box-Delivery*.

4.4.5 Area-Clearing Evaluations

We now evaluate the performance of PPO, SAC, SAM, and GTSP (Sec. 4.4.1) policies for the *Area-Clearing* environment. For this evaluation, we used the environment configuration where the robot must clear 10 boxes from within a rectangular clearance boundary without any static walls or obstacles blocking its path (Fig. 4.3). Each policy was run for 200 episodes, where the initial positions of the boxes are randomized after every episode. Also, the robot is initialized facing upwards at a random point close to the bottom of the environment. The episode terminates when all boxes are cleared out, or if the robot does not clear a box for 200 steps.

Figs. 4.8 and 4.10 show the results of evaluating the baseline policies from Table 4.1. All baselines achieve high task success scores, with SAM outperforming all baselines as it successfully clears all boxes in every episode. We also observe that SAM outperforms all baselines in efficiency, while SAC achieves the best effort score. This is because the SAM policy prefers stacking multiple boxes and clearing them simultaneously, while the SAC policy results in long paths with relatively minimal interaction with the boxes. However, in comparison with the GTSP planner, SAM achieves better effort scores as it considers object-to-object interactions while stacking boxes and minimizes the distance traveled by the boxes (lower effort). In comparison, the GTSP policy does not consider this and results in accidental stacking of boxes, which increases the robot’s effort (see Sec. 4.4.1).

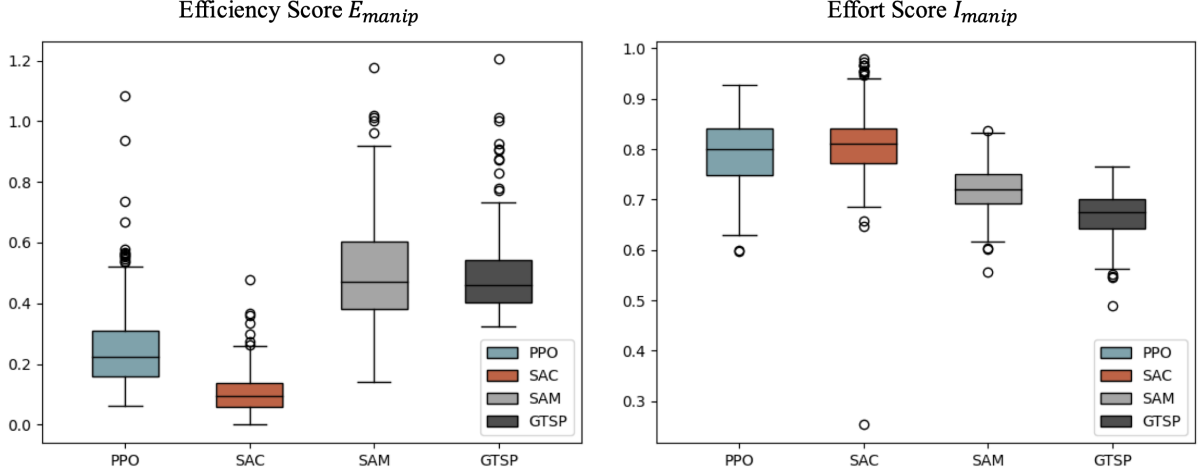


Figure 4.10: Evaluation results for PPO, SAC, SAM, and GTSP in *Area-Clearing*.

4.5 Conclusion

This chapter introduces Bench-NPIN, the first comprehensive benchmark for non-prehensile interactive navigation. Bench-NPIN features various tasks in simulated environments divided into two categories: navigation-centric tasks and manipulation-centric tasks. Users can customize tasks with different levels of complexity. Additionally, the benchmark includes implementations of both established learning policies and task-specific policies for each task, demonstrating its extensibility. It also provides a set of metrics specifically designed to measure efficiency and interaction effort within interactive navigation tasks.

Future work includes extending Bench-NPIN to allow collecting human demonstration data from the provided environments. This may enable the development of learning from demonstration (LfD) policies to tackle NPIN tasks. Bench-NPIN will also be extended to 3D environments with realistic sensor models and open-world dynamics. Finally, we plan to apply Bench-NPIN on real mobile robots to address sim-to-real gaps for interactive navigation research.

We acknowledge that it is challenging, or even impossible, to build a single benchmark that will suit the needs of all researchers in non-prehensile interactive navigation. However, we have been motivated by the importance of the area and the lack of existing benchmarks. There is significant potential for progress in NPIN, and learning-based algorithms show promise. As such, we hope that Bench-NPIN will provide a useful tool for researchers working in non-prehensile interactive navigation, and will facilitate further research and

breakthroughs in the area.

Chapter 5

Generative Skill Chaining with Out-of-distribution Predictions

In this chapter, we advance the core discussion on non-prehensile interactive navigation from specific problem grounding and benchmarking to long-horizon planning. In particular, this chapter, explores generative skill chaining with diffusion models, enabling robots to synthesize long-horizon interaction sequences for long-horizon navigation problems, while reasoning about plan execution feasibility. The resulting method could potentially be instantiated on Bench-NPIN tasks such as *Area-Clearing* and *Box-Delivery*.

5.1 Motivation

Non-prehensile interactive navigation (NPIN) tasks often involve complex, long-horizon planning with intricate inter-step dependencies. In such scenarios, the robot must execute a carefully coordinated sequence of interactions with the environment to complete tasks, such as those seen in *Box-Delivery* (Chapter 4). These interactions can cause cascading changes to the environment, compounding the difficulty of planning. Additional constraints, such as limited mobility or task-specific motion restrictions, further increase the complexity.

An illustrative example is shown in Fig. 5.1, where a robot must transport a fragile blue box to a designated goal area (green) without causing it to collide with other obstacles. A naive strategy of pushing the blue box directly would result in collisions with boxes *A*, *B*, and *C*. Instead, the robot must reason several steps ahead—planning to first move the obstructing boxes before interacting with the blue one. The planning space for such

a task quickly becomes large, growing exponentially with both the number of objects and the length of the required action sequence.

One promising way to mitigate this complexity is through motion skill planning. Today, many low-level motion controllers are well-established: for example, mobile robots can reliably perform goal-directed movement using planners like A* or RRT, or execute simple pushing behaviors via PID controllers or reinforcement learning. By assuming access to a library of such low-level motion skills, the overall problem can be structured hierarchically, where the planner first selects a high-level sequence of motion skills, and then composes them to form a complete solution trajectory.

Task and Motion Planning (TAMP) frameworks [13, 26] address this hierarchical formulation by jointly reasoning over discrete symbolic decisions and continuous motion trajectories. However, classical TAMP approaches typically assume full knowledge of system dynamics and object state representations, which limits their practicality in real-world deployments. Moreover, most hierarchical planners—including TAMP—follow an auto-regressive strategy: they generate plans sequentially, one step at a time. This leads to compounding errors and large exploration spaces as the planning horizon grows or the scene becomes increasingly cluttered.

Another fundamental challenge in long-horizon NPIN is uncertainty. As the robot interacts with multiple objects over time, the future state of the environment becomes more difficult to predict. Small modeling errors or execution disturbances can propagate across the plan, leading to compounded deviations and even failures. This is particularly problematic in densely cluttered environments, where interactions are highly coupled and nonlinear. Consequently, effective NPIN planning requires not just long-horizon reasoning, but also a mechanism to assess and anticipate uncertainty or potential failure points.

In this work, we address these challenges using the Generative Skill Chaining (GSC) framework [58], which leverages diffusion models to generate entire sequences of motion skills in a single shot. This one-shot generation approach improves planning scalability by avoiding the step-by-step expansion from traditional auto-regressive approaches. To enhance robustness, we further introduce a novel infeasibility prediction mechanism that evaluates each intermediate step in a skill sequence based on out-of-distribution (OOD) predictions. This OOD-based evaluation allows the planner to identify weak links in a proposed plan, enabling it to reject infeasible trajectories or trigger early replanning when necessary.

In summary, the key contribution of this work is an **out-of-distribution-based infeasibility prediction approach** that provides fine-grained feedback on the reliability of each step within a generative skill plan. This allows the system to reason more effectively

about uncertainty and feasibility in long-horizon, interaction-rich navigation tasks.

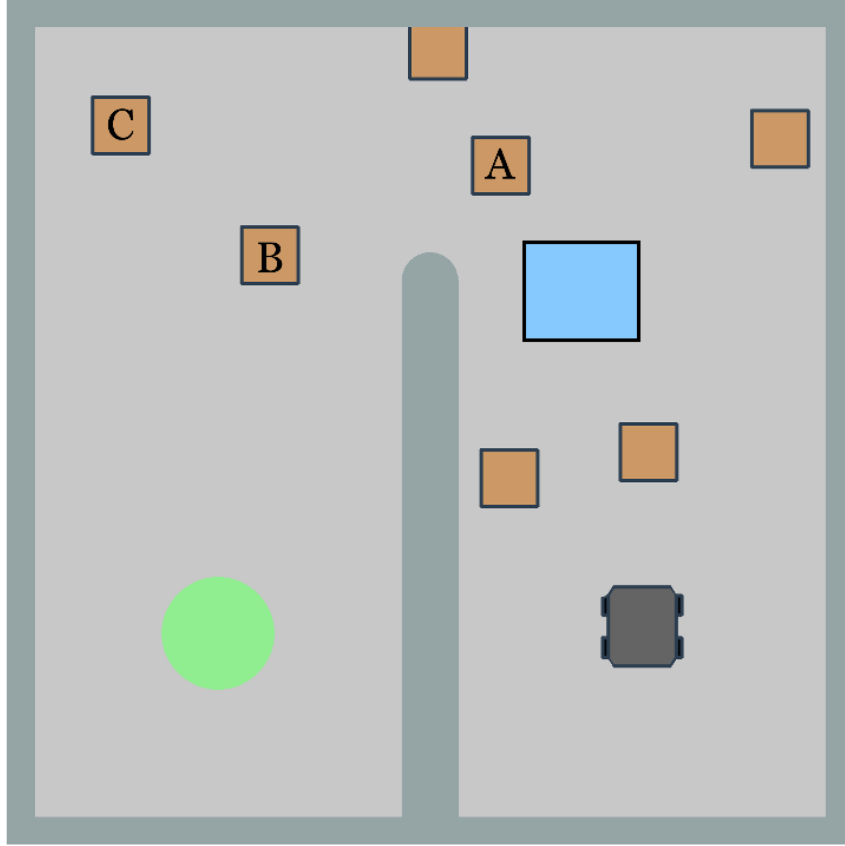


Figure 5.1: An illustration of long-horizon non-prehensile interactive navigation task - *Delicate Transport*. The robot is required to push the blue box into the green goal area. However, the blue box is a delicate object and cannot collide with any other objects. We assume that the robot has a pushing bumper with protective design. The blue box is safe to be pushed by the robot, but will be damaged by collisions with other objects in the environment.

5.2 Related Work

Task and Motion Planning. The problem of composing sequences of actions to achieve long-horizon goals has traditionally been addressed through Task and Motion Planning

(TAMP) frameworks [60, 13, 26]. TAMP typically adopts a hierarchical structure, decomposing high-level tasks into symbolically meaningful subtasks that are then grounded into continuous low-level motions. These frameworks integrate a symbolic task planner, which searches over discrete, abstract states, and a set of low-level motion planners, which solve for feasible trajectories in continuous state spaces.

When a candidate task sequence is found to be infeasible in the continuous domain (e.g., due to motion constraints or collisions), this information is often propagated back to the symbolic layer as additional constraints [13], prompting replanning. While effective in theory, TAMP approaches rely heavily on accurate, fully-observable models of the environment and robot dynamics, which are difficult to acquire or maintain in real-world and partially observable settings. As a result, their scalability and applicability to complex, dynamic environments are limited [25]. These limitations have motivated the development of learning-based alternatives that can better cope with uncertainty and partial observability.

Learning-based Models for Long-Horizon Planning. Learning-based approaches have recently gained traction for skill composition and long-horizon planning [1, 18]. Many of these methods adopt auto-regressive strategies, generating a sequence of skills or actions one step at a time. However, auto-regressive models are prone to compounding errors and suffer from large exploration spaces as the planning horizon increases or the task complexity grows.

Recent work in diffusion models for planning offers a promising alternative [48, 9, 4]. These models generate full trajectories in a single shot by denoising from an initial noise sample, effectively blurring the line between state prediction and planning. By learning the distribution of successful trajectories, they provide a scalable and expressive mechanism for long-horizon reasoning. However, since these models are typically trained via imitation learning, their generalization is limited to the data distribution observed during training. As such, they may struggle to handle novel or out-of-distribution scenarios.

The work most closely related to ours is Generative Skill Chaining (GSC) [58], which learns skill-level diffusion models and composes them to generate long-horizon trajectories. GSC enables flexible skill composition and has demonstrated strong performance in previously unseen task setups. However, a key limitation of GSC is its reliance on an external high-level task planner to provide feasible skill sequences. This assumption restricts its applicability in open-ended or unstructured domains where such a planner may not exist or be reliable.

5.3 Preliminaries

5.3.1 Diffusion Models

Diffusion models are a class of latent variable models that have demonstrated remarkable success in generating high-quality samples from complex data distributions. A diffusion model is a parametrized model $p_\theta(x_0)$ that estimates an unknown distribution $q(x_0)$ using samples $x_0 \sim q(x_0)$. There are two fundamental processes:

1. **Forward Process:** Data from $q(x_0)$ is gradually perturbed by adding Gaussian noise over a sequence of timesteps. This transforms the complex data distribution into a simple, noised distribution $q_t(x_t)$.
2. **Reverse Process:** A neural network is trained to iteratively remove the added noise to recover the clean data. At each timestep, the network predicts the noise that was added.

The reverse denoising process can be represented by a stochastic differential equation (SDE) [37] as:

$$dx = -2\dot{\sigma}_t\sigma_t\nabla_x \log q_t(x_t)dt + \sqrt{2\dot{\sigma}_t\sigma_t}d\mathbf{w}_t$$

where $\nabla_x \log q_t(x_t)$ is referred to as the **score function** of the noised distribution q_t , and $\mathbf{w}_t$ is a standard Wiener process. The score function allows the recovery of Minimum Mean Square Error Estimator (MMSE) of the clean data x_0 given x_t [58]:

$$\tilde{x}_0 = x_t + \sigma_t^2 \nabla_x \log q_t(x_t)$$

The score function is estimated using a neural network $\epsilon_\psi(x_t, t)$ with parameters ψ , which is the central quantity learned by the denoising model. In practice, the neural network $\epsilon_\psi(x_t, t)$ is trained to predict the noise ϵ that was added to a clean sample x_0 to produce the noisy sample x_t at timestep t . This is because the noise at step t is proportional to the score of the noisy data distribution $\nabla_{x_t} \log q_t(x_t)$. As such, the network is trained using the MSE loss [9]

$$\mathcal{L} = \text{MSE}(\epsilon_t, \epsilon_\psi(x_0 + \epsilon_t, t))$$

where ϵ_t is the noise added at step t . Once the model is trained, we iteratively perform the reverse denoising process. A noised data x_t is first sampled from pure noise q_t . We iteratively repeat T steps, where in each step the neural network $\epsilon_\psi(x_t, t)$ predicts the added noise and we denoise the data accordingly. The clean sample x_0 is obtained after T iterative steps of denoising.

5.3.2 Motion Skill Formulation

Here we formally define the problem of skill planning following the Generative Skill Chaining framework [58]. Let $\mathcal{S}$ denote the state space. We assume access to a library of motion skills Π , where each skill $\pi \in \Pi$ is treated as a black-box primitive to ensure generality and modularity.

Each skill π is characterized by the three following components:

1. **Precondition states** $\text{Pre}(\pi) \subseteq \mathcal{S}$ is a set of possible starting states. The skill can only be invoked when the robot is in a precondition state $s \in \text{Pre}(\pi)$. For instance, a *push-object* skill can only be invoked when the robot is within a close range to the target object.
2. **Skill parameters** Θ_π is the parameter space that dictates the behavior of the motion skill. For instance, for an *approach* skill, a skill parameter $\theta_\pi \in \Theta_\pi \subseteq \mathbb{R}^2$ could be the location of an object of interest to be approached.
3. **Postcondition states** $\text{Post}(\pi) \subseteq \mathcal{S}$ is the set of states the robot reach after a successful execution of π .

By treating each motion skill as a black box, a motion skill π can be abstractly represented as a mapping

$$\pi : \text{Pre}(\pi) \times \Theta_\pi \rightarrow \text{Post}(\pi),$$

which encodes the transition dynamics of the skill as a *state-parameter-state* relation.

An illustrative example of defining a skill’s precondition, parameters, and postcondition is presented in Fig. 5.2. Under such a setup, one can define a simple *push-box* skill as follows.

1. *Preconditions*: the robot must be within a close radius r around the target box

$$\text{Pre}(\text{push-box}) = \{(x, y) | \sqrt{(x - x_{\text{box}})^2 + (y - y_{\text{box}})^2} \leq r\}$$

2. *Skill parameters*: the parameter space could be

$$\Theta_{push-box} = \{(x_{box}, y_{box}, x_{goal}, y_{goal}) | (x_{box}, y_{box}) \in \mathbb{R}^2 \text{ and } (x_{goal}, y_{goal}) \in \mathbb{R}^2\}$$

where a valid parameter $(x_{box}, y_{box}, x_{goal}, y_{goal}) \in \Theta_{push-box}$ could be the location of the target box and the goal location.

3. *Postconditions*: the robot is within the goal region $\mathbf{G}$, assuming the robot stays in contact with the box

$$\text{Post}(push-box) = \{(x, y) | (x, y) \in \mathbf{G}\}$$

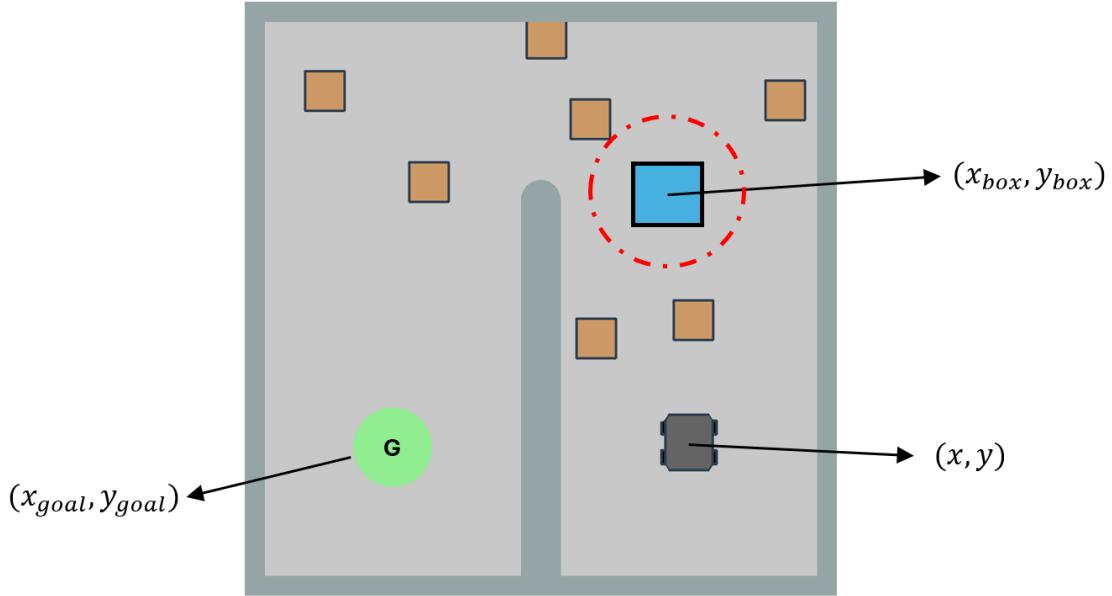


Figure 5.2: An illustration of defining a simple *push-box* skill. The blue box is the assumed target box to be pushed, and the red dotted circle denotes a close vicinity of the target box that the robot must reach before pushing. The green area is the goal.

5.3.3 Generative Skill Chaining

Building on the above formulation (Sec. 5.3.2), we define the **skill chaining** problem as finding a feasible sequence of skill transitions that take the robot from an initial state to

a goal state using a given skill sequence $\Phi_K = \{\pi_1, \pi_2, \dots, \pi_K\}$. The goal is to find a valid trajectory of interleaved states and parameters:

$$\{s_0, \theta_{\pi_1}, s_1, \theta_{\pi_2}, \dots, s_{K-1}, \theta_{\pi_K}, s_K\}, \quad (5.1)$$

where s_0 is the initial robot state, s_K is a goal state. Further, for $i \in \{1, \dots, K-1\}$, each intermediate state s_i must satisfy the consistency condition:

$$s_i \in \text{Post}(\pi_i) \cap \text{Pre}(\pi_{i+1}).$$

In other words, all intermediate states must lie in the postcondition of the previous skill, and the precondition of the subsequent skill.

Traditional methods typically follow an auto-regressive approach – sequentially selecting one skill transition at a time. Under such methods, finding a valid solution often requires searching a large planning space that expands exponentially and may lead to compounding errors [58]. To address this problem, the Generative Skill Chaining (GSC) method [58] aims to use chained diffusion models to directly sample the entire sequence, improving scalability.

Recall that each skill π is characterized by its precondition $s \in \text{Pre}(\pi)$, skill parameter $\theta_\pi \in \Theta_\pi$, and postcondition $s' \in \text{Post}(\pi)$ (Sec. 5.3.2), which is its *state-parameter-state* skill transition. For each skill $\pi \in \Pi$, a diffusion model learns the joint distribution of the skill transition

$$q_\pi(s, \theta_\pi, s').$$

To learn such a joint distribution, we train a diffusion model score function

$$\epsilon_\pi(s^t, \theta_\pi^t, s'^t, t),$$

where t is the diffusion timestep, and $(s^t, \theta_\pi^t, s'^t)$ is a noisy transition sampled during diffusion process. Such a score function is trained using the observed skill transitions $s, \theta_\pi, s' \sim q_\pi(s, \theta_\pi, s')$ from executing π during some data collection process.

Given a skill sequence Φ_K , to generate a sample of the entire skill transition sequence in Eq. 5.1, we model the full joint distribution:

$$q_{\Phi_K}(s_0, \theta_{\pi_1}, s_1, \theta_{\pi_2}, \dots, s_{K-1}, \theta_{\pi_K}, s_K)$$

using a chained score function

$$\epsilon_{\Phi_K}(s_0^t, \theta_{\pi_1}^t, s_1^t, \theta_{\pi_2}^t, \dots, s_{K-1}^t, \theta_{\pi_K}^t, s_K^t, t).$$

The GSC [58] showed that this chained score function can be obtained by composing the individual skill diffusion models. In particular, given a skill sequence $\Phi_K = \{\pi_1, \pi_2, \dots, \pi_K\}$ and the corresponding diffusion models $\{\epsilon_{\pi_1}, \epsilon_{\pi_2}, \dots, \epsilon_{\pi_K}\}$, the chained score function is

$$\begin{aligned} \epsilon_{\Phi_K}(s_0^t, \theta_{\pi_1}^t, s_1^t, \theta_{\pi_2}^t, \dots, s_{K-1}^t, \theta_{\pi_K}^t, s_K^t, t) = \\ \epsilon_{\pi_1}(s_0^t, \theta_{\pi_1}^t, s_1^t, t) + \epsilon_{\pi_2}(s_1^t, \theta_{\pi_2}^t, s_2^t, t) + \dots + \epsilon_{\pi_K}(s_{K-1}^t, \theta_{\pi_K}^t, s_K^t, t) \\ - [\gamma_{\pi_1} \epsilon_{\pi_1}(s_1^t, t) + (1 - \gamma_{\pi_1}) \epsilon_{\pi_2}(s_1^t, t)] - [\gamma_{\pi_2} \epsilon_{\pi_2}(s_2^t, t) + (1 - \gamma_{\pi_2}) \epsilon_{\pi_3}(s_2^t, t)] - \dots \\ - [\gamma_{\pi_{K-1}} \epsilon_{\pi_{K-1}}(s_{K-1}^t, t) + (1 - \gamma_{\pi_{K-1}}) \epsilon_{\pi_K}(s_{K-1}^t, t)], \end{aligned} \quad (5.2)$$

where each $\gamma_{\pi_i} \in [0, 1]$ is a tunable parameter that balances the influence of π_i and π_{i+1} on the intermediate state s_i . Eq. 5.2 exhibits several nice properties:

1. **No addition training required.** The chained score function is simply a chained sequence of calls to the trained neural networks. Once the individual skill diffusion models are trained, any skill sequence can be composed at inference time without additional training.
2. **Parallelizability.** Since the individual skill models are independent, inference can be parallelized across the sequence, improving scalability.

Limitations and Motivation for Extension Despite its efficiency, GSC assumes that a feasible skill sequence is known and valid, an assumption that limits its applicability in many real-world settings. In practice, higher-level planners may suggest skill sequences that are kinematically infeasible due to unforeseen motion constraints or environment dynamics [13]. GSC, as originally proposed, lacks a mechanism to detect or handle such infeasibility.

Furthermore, when deploying GSC in long-horizon non-prehensile interactive navigation tasks, the robot must reason not only about generating skill transitions efficiently, but also about uncertainty and potential failures during execution.

To bridge this gap, we propose an extension to the GSC framework that integrates out-of-distribution (OOD) based infeasibility prediction. Given a generated skill chain, our approach estimates the likelihood of failure at each intermediate transition by detecting OOD transitions. This potentially enables:

1. Identification of weak or unreliable portions of a plan.

2. Replanning from potentially infeasible segments, rather than from scratch.
3. Reasoning about partial feasibility, which allows more flexible and robust execution.

This extension could potentially enhance the practicality of GSC in complex and cluttered environments where skill-level failures are common and unavoidable.

5.4 OOD Based Infeasibility Predictions

Recall that we consider each motion skill π as a black-box transition model, characterized by a joint distribution over its precondition, parameters, and postcondition as $q_\pi(s, \theta_\pi, s')$. Such a joint distribution captures the feasible transitions of π . An out-of-distribution (OOD) data here refers to a transition that significantly differs from the distribution of feasible transitions, and thus signals infeasibility. Specifically, if the sample (s, θ_π, s') is out-of-distribution from q_π , then (s, θ_π, s') is unlikely a feasible transition for π . In other words, executing π from state s with parameter θ_π will unlikely to result in a postcondition state s' and may lead to failure or undefined behaviors. Our approach builds on this intuition by computing an OOD score for each transition along the generated sequence $\{s_0, \theta_{\pi_1}, s_1, \theta_{\pi_2}, \dots, s_{K-1}, \theta_{\pi_K}, s_K\}$.

Instead of training new models, we leverage the diffusion models ϵ_π that were already trained to model the distribution q_π . Inspired by methods for OOD detection in generative modeling [30], we measure reconstruction error as a proxy for infeasibility.

The overall approach is that given a generated transition sequence using GSC, we re-inject T steps of i.i.d. noise into the sequence to perturb the full sequence. We then reconstruct each transition using individual skill diffusion, and compute the OOD score (i.e. infeasibility measures) as the reconstruction error for each skill transition (s, θ_π, s') . The overall intuition of this approach is that the diffusion models have already learned to denoise in-distribution data accurately, so it's expected to have a low reconstruction error for noised inlier transitions (i.e. feasible transitions). Conversely, a noised skill transition that exhibits a high reconstruction error is likely an OOD transition and infeasible.

Alg. 2 presents our algorithm that performs generative skill chaining with OOD predictions. Specifically, our algorithm takes as inputs a given skill sequence $\Phi_K = \{\pi_1, \pi_2, \dots, \pi_K\}$ from some higher-level skill planner, the corresponding set of diffusion models $\mathcal{E}_K = \{\epsilon_{\pi_1}, \epsilon_{\pi_2}, \dots, \epsilon_{\pi_K}\}$, and a reconstruction step T .

A sequence of potentially feasible transitions $\{s_0, \theta_{\pi_1}, s_1, \theta_{\pi_2}, \dots, s_{K-1}, \theta_{\pi_K}, s_K\}$ is first computed using the Generative Skill Chaining technique [58], as outlined in Sec. 5.3.3 (Line

1). The generated transition sequence is then iteratively perturbed by T steps of i.i.d. noise (Line 2). An empty list is created to store the reconstruction error for each motion skill transition (Line 3). The algorithm then loops through all perturbed skill transitions, and for each perturbed transition, T steps of reverse denoising process is performed using the corresponding skill diffusion model to reconstruct the skill transition (Lines 5 - 7). For each skill transition, the reconstruction error is then individually computed for its precondition, skill parameter, and postcondition (Line 8) and recorded (Line 9). In particular, $\delta_{s_{i-1}}^{\pi_i}$ and $\delta_{s_i}^{\pi_i}$ denotes the reconstruction error for the precondition state and postcondition state for skill π_i . Finally, the generated skill transitions along with its reconstruction error as OOD scores are returned.

Algorithm Discussion: Note that for the sequence $\{s_0, \theta_{\pi_1}, s_1, \theta_{\pi_2}, \dots, s_{K-1}, \theta_{\pi_K}, s_K\}$, Alg. 2 does not compute OOD scores for each element along the sequence, but rather for each individual skill transition. In other words, for $i \in \{1, \dots, K-1\}$, each intermediate state s_i has two OOD scores: $\delta_{s_i}^{\pi_i}$ measures the feasibility of s_i as a **postcondition** for skill π_i , and $\delta_{s_i}^{\pi_{i+1}}$ measures the feasibility of s_i as a **precondition** for skill π_{i+1} . This allows for more in-depth reasoning about the feasibility of the generated transitions, as illustrated in Sec. 5.5.

For instance, it is possible that an intermediate states s_i exhibits a low OOD score as a postcondition for skill π_i , but a high OOD score as a precondition for subsequent skill π_{i+1} . This implies that the execution of π_i and reaching state s_i is likely to succeed, but the next skill π_{i+1} is unlikely to be successfully executed from s_i . Such predictions on partial infeasibility enable selective replanning from π_{i+1} onward, rather than restarting the plan from scratch.

Algorithm 2: Generative Skill Chaining with OOD Predictions

Input: $\Phi_K = \{\pi_1, \pi_2, \dots, \pi_K\}$, $\mathcal{E}_K = \{\epsilon_{\pi_1}, \epsilon_{\pi_1}, \dots, \epsilon_{\pi_K}\}$, T

- 1 $\{s_0, \theta_{\pi_1}, s_1, \theta_{\pi_2}, \dots, s_{K-1}, \theta_{\pi_K}, s_K\} \leftarrow \text{GSC}(\Phi_K, \mathcal{E}_K)$ // generate transitions [58]
- 2 $\{s_0^T, \theta_{\pi_1}^T, s_1^T, \theta_{\pi_2}^T, \dots, s_{K-1}^T, \theta_{\pi_K}^T, s_K^T\} \leftarrow \text{Inject } T \text{ steps of i.i.d. noise}$
- 3 $\text{reconst_err}[] \leftarrow \text{reconstruction error list}$
- 4 **for** $i = 1, \dots, K$ **do**
- 5 **for** $t = T, \dots, 1$ **do**
- 6 $\epsilon \leftarrow \epsilon_{\pi_i}(s_{i-1}^t, \theta_{\pi_i}^t, s_i^t, t)$ // noise prediction
- 7 $\{s_{i-1}^{t-1}, \theta_{\pi_i}^{t-1}, s_i^{t-1}\} \leftarrow \text{Denoise}(\{s_{i-1}^t, \theta_{\pi_i}^t, s_i^t\}, \epsilon)$ // reconstruct skill transition
- 8 $\delta_{s_{i-1}}^{\pi_i}, \delta_{\theta_{\pi_i}}^{\pi_i}, \delta_{s_i}^{\pi_i} \leftarrow \|(s_{i-1}, \theta_{\pi_i}, s_i) - (s_{i-1}^0, \theta_{\pi_i}^0, s_i^0)\|$ // compute reconstruction error
- 9 $\text{reconst_err.add}(\delta_{s_{i-1}}^{\pi_i}, \delta_{\theta_{\pi_i}}^{\pi_i}, \delta_{s_i}^{\pi_i})$
- 10 **return** $\{s_0, \theta_{\pi_1}, s_1, \theta_{\pi_2}, \dots, s_{K-1}, \theta_{\pi_K}, s_K\}, \text{reconst_err}$

5.5 Preliminary Studies

This section presents a preliminary studies on a toy example to demonstrate the effectiveness of our proposed algorithm. Recall that we treat motion skills as black box functions that map a precondition state and skill parameter pair to a postcondition state. To construct a toy example, we define three motion skills, π_a, π_b, π_c , as simple addition functions as follows.

1. π_a : addition function $\pi_a((x, y), (dx, dy)) = (x + dx, y + dy)$
 - (a) Precondition states: $\{(x, y) | x, y \in [0, 5]\}$
 - (b) Skill parameter $\{(dx, dy) | dx, dy \in [10, 15]\}$
 - (c) Postcondition states $\{(x', y') | x', y' \in [10, 20]\}$
2. π_b : addition function $\pi_b((x, y), (dx, dy)) = (x + dx, y + dy)$
 - (a) Precondition states: $\{(x, y) | x, y \in [15, 20]\}$
 - (b) Skill parameter $\{(dx, dy) | dx, dy \in [10, 15]\}$
 - (c) Postcondition states $\{(x', y') | x', y' \in [25, 35]\}$
3. π_c : subtraction function $\pi_c((x, y), (dx, dy)) = (x - dx, y - dy)$
 - (a) Precondition states: $\{(x, y) | x \in [30, 35], y \in [20, 30]\}$
 - (b) Skill parameter $\{(dx, dy) | dx \in [0, 5], dy \in [15, 20]\}$
 - (c) Postcondition states $\{(x', y') | x' \in [25, 35], y' \in [0, 15]\}$

A visual illustration of the set of precondition states and the set of postcondition states for the motion skills is presented in Fig. 5.3.

For each motion skill $\pi \in \{\pi_a, \pi_b, \pi_c\}$, a diffusion model is trained using a synthetic dataset $s, \theta_\pi, s' \sim q_\pi(s, \theta_\pi, s')$. Each data point (s, θ_π, s') is generated by first randomly sampling a precondition state s and a skill parameter θ_π from their corresponding ranges. Then postcondition state s' is computed by either adding or subtract s with θ_π , depending on the motion skill defined. 500 entries are used to train each diffusion model denoiser. The sampling results for each skill's diffusion model can be visualized in Fig. 5.4.

As shown in Fig. 5.4, the sampled precondition states are generally mapped to the sampled postcondition states by the sampled arrows. This indicates that the diffusion models learned the underlying skill transition distribution reasonably well.

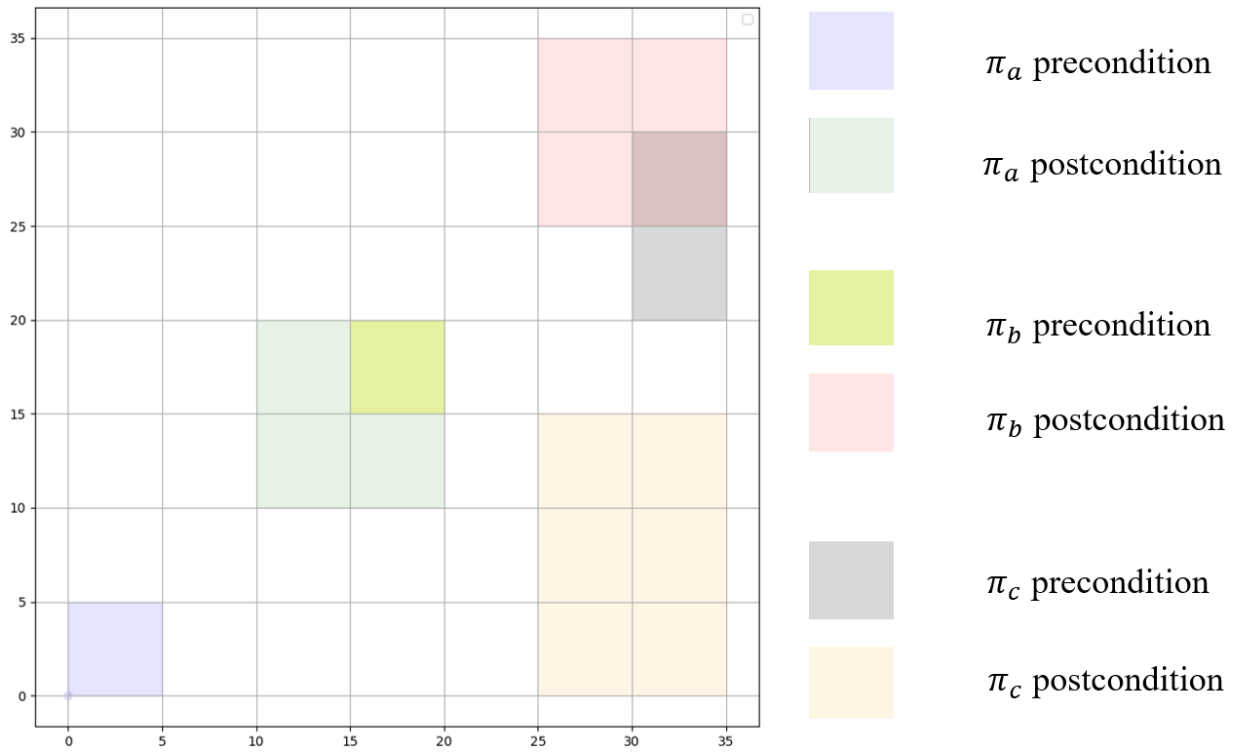


Figure 5.3: A visual illustration of the precondition and postcondition states for motion skills π_a, π_b, π_c .

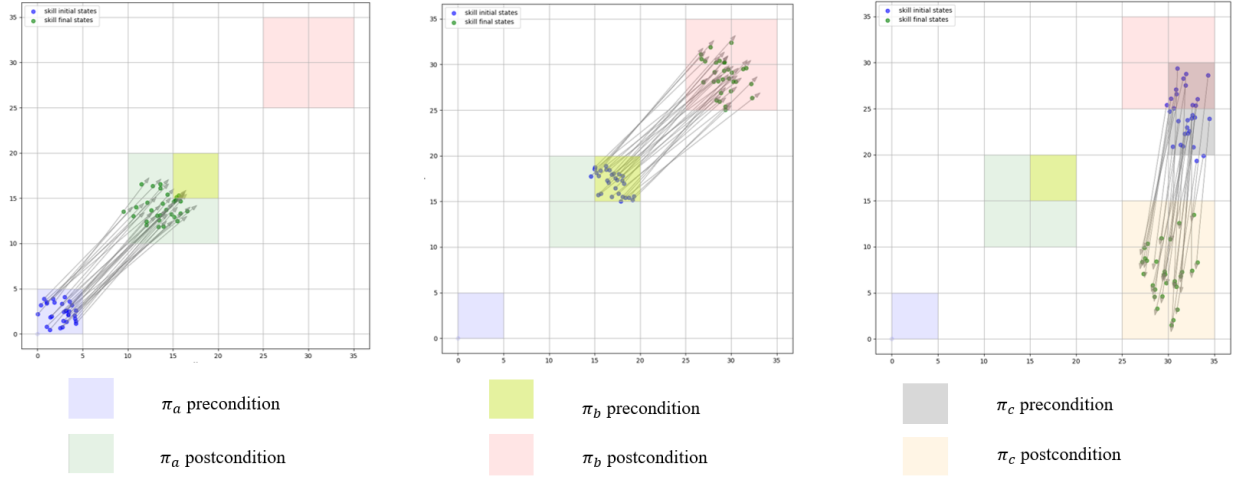


Figure 5.4: A visualization of sampling results for diffusion models for each skill. The sampled states (both precondition and postcondition states) are represented by dots, while skill parameters are represented as arrows. 30 transitions are sampled for each diffusion model.

We now apply Generative Skill Chaining [58] on the defined skills. From Fig. 5.4, note that π_a 's postcondition set overlaps with π_b 's precondition set, and π_b 's postcondition set overlaps with π_c 's precondition set. It follows that the skill sequence $\Phi_{\text{feasible}} = \{\pi_a, \pi_b, \pi_c\}$ is clearly feasible. The skill composition results for sequence Φ_{feasible} are visualized in Fig. 5.5. Note that Fig. 5.5 and Fig. 5.4 together show that the GSC effectively steer the intermediate states (green and purple dots) toward the overlap regions between two subsequent skills. This reflects the constraint from Sec. 5.3.3 where all intermediate states must lie in the postcondition of the previous skill and the precondition of the subsequent skill.

We now construct an infeasible skill sequence as $\Phi_{\text{infeasible}} = \{\pi_a, \pi_b, \pi_a\}$. More specifically, one can see that $\Phi_{\text{infeasible}}$ is *partially feasible* that, it is feasible for executing π_a and π_b (i.e. first two skills), but the execution of the third skill π_a after the second skill π_b is infeasible. The generated GSC results of this infeasible skill sequence are visualized in Fig. 5.6. The arrows from π_b that do not point at any dots indicate that the generated transition sequence is somewhat meaningless. This also reflects a limitation of the current GSC framework [58] that the method will also always generate a chained sequence regardless of whether the generation is meaningful or feasible. This potentially raises reliability issue for deploying the framework in real world without any additional method to check for infeasibility.

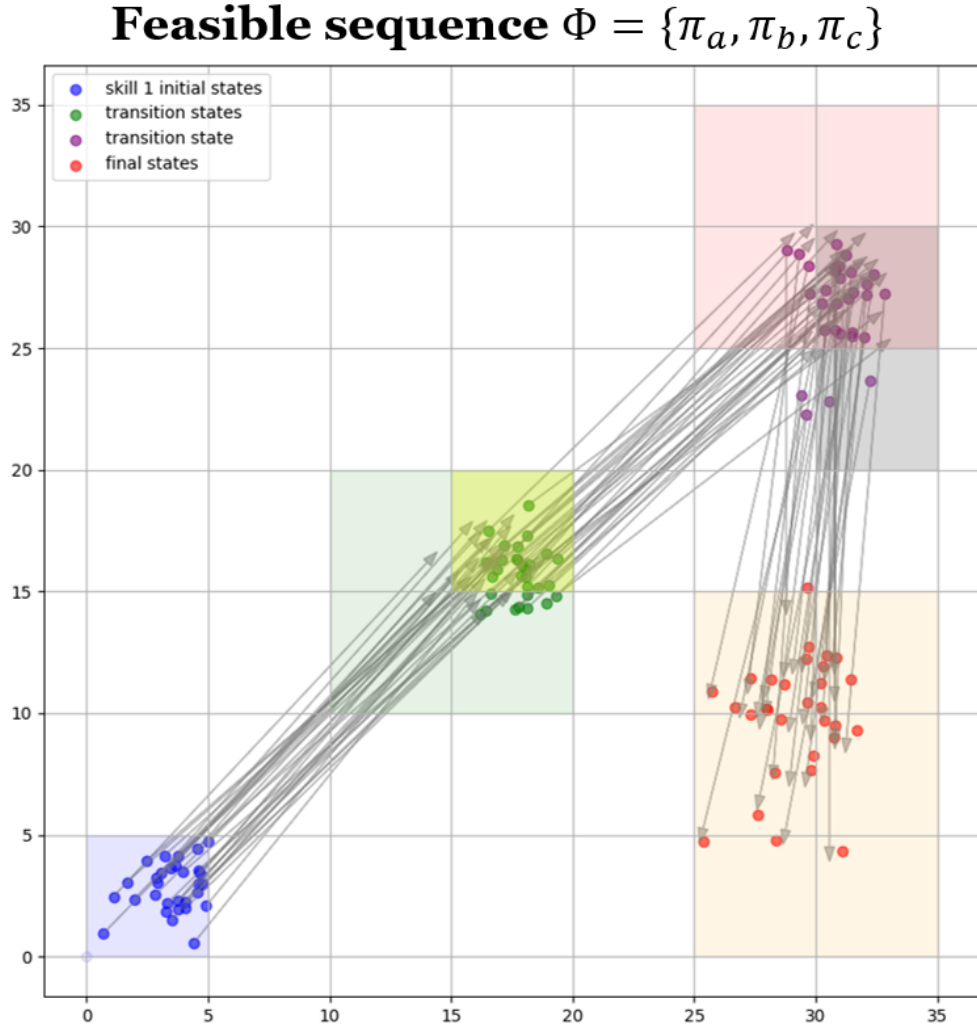


Figure 5.5: A visualization of 30 sampling results for applying Generative Skill Chaining [58] on the feasible skill sequence $\Phi_{\text{feasible}} = \{\pi_a, \pi_b, \pi_c\}$

Infeasible sequence $\Phi = \{\pi_a, \pi_b, \pi_a\}$

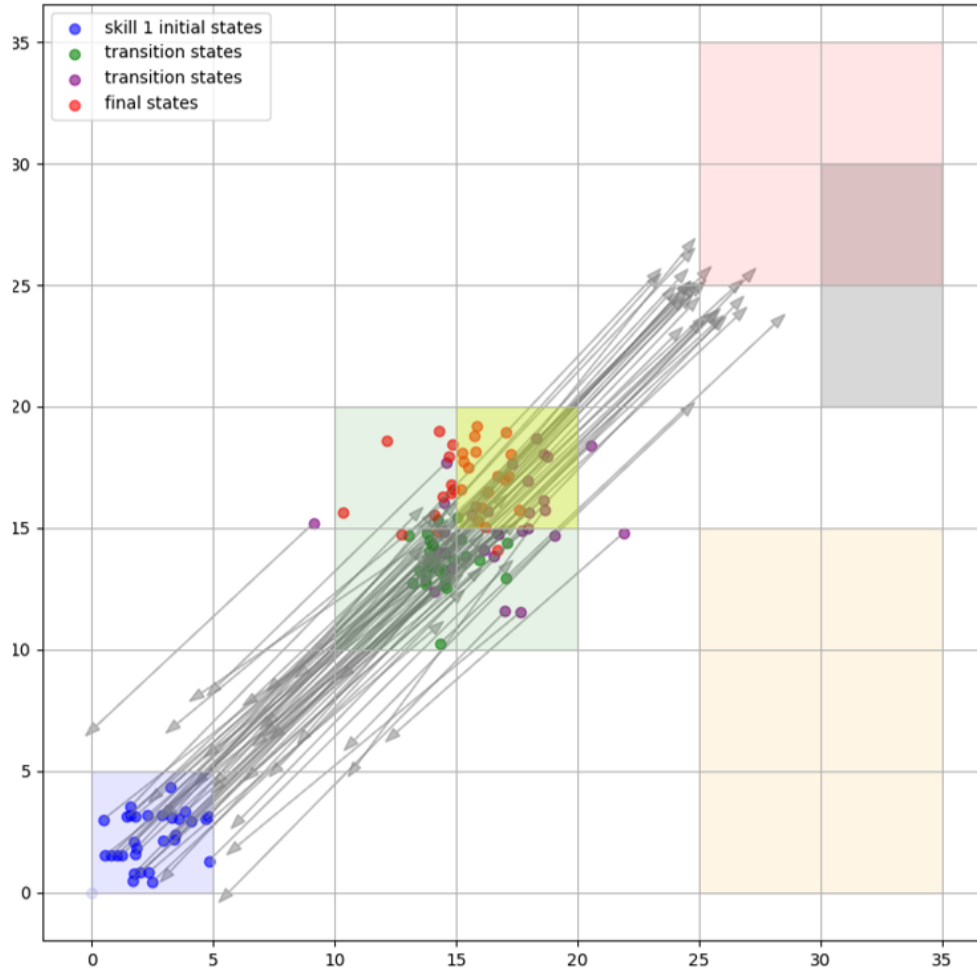


Figure 5.6: A visualization of 30 sampling results for applying Generative Skill Chaining [58] on the infeasible skill sequence $\Phi_{\text{infeasible}} = \{\pi_a, \pi_b, \pi_a\}$

We now apply our OOD predictions outlined in Alg. 2 on both $\Phi_{\text{feasible}} = \{\pi_a, \pi_b, \pi_c\}$ and $\Phi_{\text{infeasible}} = \{\pi_a, \pi_b, \pi_a\}$ to demonstrate its effectiveness in quantifying infeasibility. By running the algorithm on both sequences, the outputs are visualized in Fig. 5.7.

Note that the OOD scores from our algorithm provides detailed information regarding the skill transitions feasibility. For the feasible skill sequence $\Phi_{\text{feasible}} = \{\pi_a, \pi_b, \pi_c\}$, our algorithm consistently outputs low OOD scores for all skill transitions, indicating a high likelihood of feasible transitions. For the infeasible skill sequence $\Phi_{\text{infeasible}} = \{\pi_a, \pi_b, \pi_a\}$ (partially feasible), our algorithm first outputs low OOD scores up to θ_{π_2} . This indicates that the execution for skill $\pi_1 = \pi_a$ is likely to be feasible. Moreover, it is feasible to invoke $\pi_2 = \pi_b$ after $\pi_1 = \pi_a$. However, the OOD scores increased drastically from s_2 onward. This suggests that while it is feasible to invoke $\pi_2 = \pi_b$, π_2 is unable to steer its postcondition state to make the subsequent skill $\pi_3 = \pi_a$ feasible to execute. In other words, our OOD scores indicate that $\Phi_{\text{infeasible}} = \{\pi_a, \pi_b, \pi_a\}$ is partially feasible for the first two skills, but becomes infeasible for executing the third skill, which conform with out setup shown in Fig. 5.4.

The OOD scores from Fig. 5.7 indicate that our method not only capture infeasibility, but also provides more detailed information regarding partial feasibility and specific regions of infeasibility. This could potentially be useful for rejecting infeasible skill sequence, or developing efficient replan strategies that leverage partial feasibility.

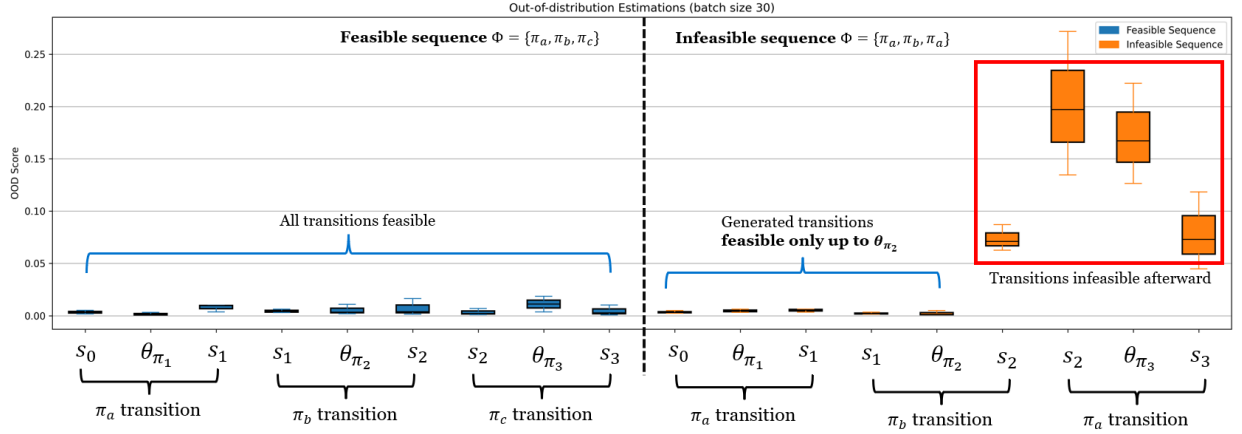


Figure 5.7: OOD scores computed using our Alg. 2 on the generated feasible sequences from Fig. 5.5 for $\Phi_{\text{feasible}} = \{\pi_a, \pi_b, \pi_c\}$ and on the generated infeasible sequences from Fig. 5.6 for $\Phi_{\text{infeasible}} = \{\pi_a, \pi_b, \pi_a\}$

5.6 Conclusion

This chapter introduced a preliminary framework for applying generative skill chaining to long-horizon interactive navigation tasks. Building on recent advances in diffusion-based trajectory generation, we modeled motion skills as learned transitions between preconditions, skill parameters, and postconditions. We then formulated skill chaining as a generative sampling problem and introduced an out-of-distribution (OOD) estimation technique for identifying infeasible skill transitions. Our preliminary studies demonstrated the effectiveness of using OOD detection to anticipate failure points in skill compositions and highlighted the potential of this method for improving robustness in long-horizon planning under uncertainty. As a forward-looking component of this thesis, this chapter serves to bridge the foundational benchmarks and applied case studies with emerging generative methods for complex planning problems.

Looking ahead, we aim to continue to improve and evaluate our method. This includes collecting robotic-scale NPIN dataset to test whether the OOD prediction framework generalizes beyond the toy environments used in our initial study. We also aim to build a full-stack skill planner that integrates high-level task planning with low-level skill composition. In particular, we seek to relax the assumption from [58] that a feasible skill sequence is given in advance. Instead, our long-term goal is to develop a high-level task planner that interfaces with the low-level skill generator and uses OOD feedback to replan when infeasibility is detected.

Chapter 6

Conclusion

This thesis investigates non-prehensile interactive navigation (NPIN), a class of robot navigation problems where physical interaction with the environment is essential. While traditional navigation systems prioritize collision avoidance, NPIN accepts and leverages contact to enable navigation in cluttered or constrained environments. Our work spans three key contributions that reflect both foundational and forward-looking efforts in this domain.

First, we studied a real-world application of NPIN in autonomous surface vehicle (ASV) navigation through ice-covered waters. We proposed a learning-based planning approach that incorporates predicted obstacle motion into the planning loop. By combining deep neural networks for occupancy prediction with a graph search planner, we demonstrated improved navigation performance both in simulation and in physical testbeds, even under high-density ice conditions.

Second, we broadened our scope and introduced Bench-NPIN, the first comprehensive benchmarking suite for NPIN. This tool includes a variety of environments, ranging from maze navigation to object clearing tasks, and offers evaluation metrics that explicitly account for trade-offs between efficiency and interaction. Bench-NPIN aims to provide a shared platform for researchers to compare algorithms, study generalization, and accelerate progress in this emerging field.

Third, we presented a preliminary investigation into generative skill chaining for long-horizon planning under uncertainty. Leveraging diffusion models, we showed how entire sequences of low-level motion skill transitions can be sampled directly. Furthermore, we proposed an out-of-distribution (OOD) prediction framework to identify infeasible skill executions, enabling early failure detection and improving the reliability of planning.

Together, these contributions form a cohesive effort toward enabling interactive navigation in realistic, unstructured, and dynamic settings. While our benchmark and ASV navigation study provide concrete tools and applications, the generative planning work serves as an initial step toward more flexible and generalizable planning frameworks.

6.1 Discussions and Future Work

Non-prehensile interactive navigation remains a young field, and the three threads of this thesis—the ship-ice case study, Bench-NPIN, and generative skill chaining—highlight both its promise and its open problems. Below outlines the main challenges and practical directions that can potentially move the community forward.

First, closing the sim-to-real gap is essential. Reproducing contact-rich behavior realistically remains a challenge in today’s simulators. As seen in Chapter 3, while our trained ice motion prediction model enhanced the planner’s performance, our training data is limited in simulation and does not consider environmental factors such as fluid dynamics and winds, which play an important role in real-world ship planning. Progress will come from building higher-fidelity and efficient 3-D simulators that include better contact and hydrodynamic physics, or applying domain randomization techniques to expose learning algorithms to realistic disturbances.

Second, robots must cope with a wider spectrum of interaction modalities than those modeled so far. A household robot, for instance, may need to push box-shaped objects, roll wheeled furniture, and slide or compress deformable items such as cloth toys; each class of object obeys different contact dynamics, so extending Bench-NPIN to cover rigid, articulated, rolling, and deformable obstacles will encourage algorithms that handle heterogeneous interactions rather than a single object type.

Third, uncertainty and partial observability are natural challenges for NPIN in cluttered environments. The experiments in this thesis assume an overhead camera with complete visibility, but real robots must deal with occlusions, limited sensor range, and noisy measurements. Informative path planning, active perception strategies that strategically leverage sensors to reduce uncertainty, and robust policy-learning techniques that inject sensor noise during training are potentially promising directions for open-world NPIN.

Future work should address these challenges, and an upgraded benchmark infrastructure is already under construction: our ongoing efforts involve extending Bench-NPIN to a fully 3D MuJoCo simulator with modular robot libraries and multiple sensing modalities. We also plan to extend the benchmark to include physical robot experiments, particularly in

the *Box-Delivery*, *Maze*, and *Area-Clearing* tasks, to validate transferability and real-world performance.

Foundation models are another tool worth exploring in future work. Understanding a cluttered and unseen environment could be challenging for existing methods. Large vision-language-action (VLA) models can inject semantic priors that help robots generalize to novel layouts and propose high-level task decompositions for skill composition, a direction that can potentially accelerate the generative skill chaining concepts from Chapter 5.

Finally, the long-term goal is lifelong learning. Rather than freezing a set of reactive skills during deployment, future NPIN agents should refine contact models, expand their skill libraries, and update their planners online—without costly retraining. Achieving this ambition will require safe self-supervised exploration in contact-rich settings, efficient online adaptation of learned dynamics and reward models, and new metrics that quantify continual learning.

By pursuing these directions—better physics, richer objects, principled handling of uncertainty, stronger benchmarks, semantic priors from foundation models, and lifelong learning—the robotics community can transform NPIN from an intriguing research topic into practical solutions for robots operating in the messy, human-centered spaces where they are most needed.

References

- [1] Christopher Agia, Toki Migimatsu, Jiajun Wu, and Jeannette Bohg. Stap: Sequencing task-agnostic policies. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7951–7958. IEEE, 2023.
- [2] Ben Agro, Quinlan Sykora, Sergio Casas, and Raquel Urtasun. Implicit occupancy flow fields for perception and prediction in self-driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1379–1388, 2023.
- [3] Hyemin Ahn, Timothy Ha, Yunho Choi, Hwiyeon Yoo, and Songhwai Oh. Text2action: Generative adversarial synthesis from language to action. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5915–5920. IEEE, 2018.
- [4] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.
- [5] Kristoffer Bergman, Oskar Ljungqvist, Jonas Linder, and Daniel Axehill. An optimization-based motion planner for autonomous maneuvering of marine vessels in complex environments. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 5283–5290. IEEE, 2020.
- [6] Victor Blomqvist. Pymunk: A easy-to-use pythonic rigid body 2d physics library. (version 6.7.0), 2024.
- [7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- [8] Tingting Chen, Xueping Liu, Bizhong Xia, Wei Wang, and Yongzhi Lai. Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder. *Ieee Access*, 8:47072–47081, 2020.

- [9] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.
- [10] Hao-Tien Lewis Chiang and Lydia Tapia. Colreg-rrt: An rrt-based colregs-compliant motion planner for surface vehicle navigation. *IEEE Robotics and Automation Letters*, 3(3):2024–2031, 2018.
- [11] Minjoo Choi, Hyun Chung, Hajime Yamaguchi, and Keisuke Nagakawa. Arctic sea route path planning based on an uncertain ice prediction model. *Cold Regions Science and Technology*, 109:61–69, 2015.
- [12] Cunxi Dai, Xiaohan Liu, Koushil Sreenath, Zhongyu Li, and Ralph Hollis. Interactive navigation with adaptive non-prehensile mobile manipulation. *arXiv preprint arXiv:2410.13418*, 2024.
- [13] Neil T Dantam, Zachary K Kingston, Swarat Chaudhuri, and Lydia E Kavraki. An incremental constraint-based framework for task and motion planning. *The International Journal of Robotics Research*, 37(10):1134–1151, 2018.
- [14] Rodrigue de Schaetzen, Alexander Botros, Robert Gash, Kevin Murrant, and Stephen L Smith. Real-time navigation for autonomous surface vehicles in ice-covered waters. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1069–1075. IEEE, 2023.
- [15] Rodrigue de Schaetzen, Alexander Botros, Ninghan Zhong, Kevin Murrant, Robert Gash, and Stephen L Smith. AUTO-IceNav: A local navigation strategy for autonomous surface ships in broken ice fields. *arXiv preprint arXiv:2411.17155*, 2024.
- [16] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.
- [17] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.
- [18] Danny Driess, Jung-Su Ha, and Marc Toussaint. Learning to solve sequential physical reasoning problems from a scene image. *The International Journal of Robotics Research*, 40(12-14):1435–1466, 2021.

- [19] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.
- [20] Kirsty Ellis, Denis Hadjivelichkov, Valerio Modugno, Danail Stoyanov, and Dimitrios Kanoulas. Navigation among movable obstacles via multi-object pushing into storage zones. *IEEE Access*, 11:3174–3183, 2023.
- [21] Gyuho Eoh. Deep-reinforcement-learning-based object transportation using task space decomposition. *Sensors*, 23(10):4807, 2023.
- [22] Gyuho Eoh and Tae-Hyoung Park. Cooperative object transportation using curriculum-based deep reinforcement learning. *Sensors*, 21(14):4780, 2021.
- [23] Tom Erez, Yuval Tassa, and Emanuel Todorov. Infinite horizon model predictive control for nonlinear periodic tasks. *Manuscript under review*, 4, 2011.
- [24] Changxiang Fan, Shouhei Shirafuji, and Jun Ota. Modal planning for cooperative non-prehensile manipulation by mobile robots. *Applied Sciences*, 9(3):462, 2019.
- [25] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2786–2793. IEEE, 2017.
- [26] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4(1):265–293, 2021.
- [27] Robert M Gash, Kevin A Murrant, Jason W Mills, and David EL Millan. Machine vision techniques for situational awareness and path planning in model test basin ice-covered waters. In *Global Oceans 2020: Singapore–US Gulf Coast*, pages 1–8. IEEE, 2020.
- [28] Floris Goerlandt, Jakub Montewka, Weibin Zhang, and Pentti Kujala. An analysis of ship escort and convoy operations in ice conditions. *Safety science*, 95:198–209, 2017.
- [29] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

- [30] Mark S. Graham et al. Denoising diffusion models for out-of-distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [31] Nico Gürtler, Felix Widmaier, Cansu Sancaktar, Sebastian Blaes, Pavel Kolev, Stefan Bauer, Manuel Wüthrich, Markus Wulfmeier, Martin Riedmiller, Arthur Allshire, et al. Real robot challenge 2022: Learning dexterous manipulation from offline data in the real world. In *NeurIPS 2022 Competition Track*, pages 133–150. PMLR, 2023.
- [32] Jérôme Guzzi, R Omar Chavez-Garcia, Mirko Nava, Luca Maria Gambardella, and Alessandro Giusti. Path planning with local motion estimations. *IEEE Robotics and Automation Letters*, 5(2):2586–2593, 2020.
- [33] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.
- [34] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
- [35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [36] Agnieszka Herman. Wave-induced surge motion and collisions of sea ice floes: Finite-floe-size effects. *Journal of Geophysical Research: Oceans*, 123(10):7472–7494, 2018.
- [37] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [38] Lauren Hoffman, Matthew R Mazloff, Sarah T Gille, Donata Giglio, Cecilia M Bitz, Patrick Heimbach, and Kayli Matsuyoshi. Machine learning for daily forecasts of arctic sea ice motion: An attribution assessment of model predictive skill. *Artificial Intelligence for the Earth Systems*, 2(4):230004, 2023.
- [39] Jarrett Holtz and Joydeep Biswas. Socialgym: A framework for benchmarking social robot navigation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11246–11252. IEEE, 2022.

- [40] Tsung-Hsuan Hsieh, Shengzheng Wang, Huijia Gong, Wei Liu, and Ning Xu. Sea ice warning visualization and path planning for ice navigation based on radar image recognition. *Journal of Marine Science and Technology*, 29(3):280–290, 2021.
- [41] Tsung-Hsuan Hsieh, Shengzheng Wang, Huijia Gong, Wei Liu, and Ning Xu. Sea ice warning visualization and path planning for ice navigation based on radar image recognition. *Journal of Marine Science and Technology*, 29(3):280–290, 2021.
- [42] Luofeng Huang, Zhiyuan Li, Christopher Ryan, Jonas W Ringsberg, Blanca Pena, Minghao Li, Li Ding, and Giles Thomas. Ship resistance when operating in floating ice floes: Derivation, validation, and application of an empirical equation. *Marine Structures*, 79:103057, 2021.
- [43] Luofeng Huang and Giles Thomas. Simulation of wave interaction with a circular ice floe. *Journal of Offshore Mechanics and Arctic Engineering*, 141(4):041302, 2019.
- [44] Luofeng Huang, Jukka Tuhkuri, Bojan Igrec, Minghao Li, Dimitris Stagonas, Alessandro Toffoli, Philip Cardiff, and Giles Thomas. Ship resistance when operating in floating ice floes: A combined cfd&dem approach. *Marine Structures*, 74:102817, 2020.
- [45] Peter J Huber. Robust statistics. In *International encyclopedia of statistical science*, pages 1248–1251. Springer, 2011.
- [46] Boris Ivanovic, Karen Leung, Edward Schmerling, and Marco Pavone. Multimodal deep generative models for trajectory prediction: A conditional variational autoencoder approach. *IEEE Robotics and Automation Letters*, 6(2):295–302, 2020.
- [47] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.
- [48] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.
- [49] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.
- [50] Linh Kästner, Teham Bhuiyan, Tuan Anh Le, Elias Treis, Johannes Cox, Boris Meinardus, Jacek Kmiecik, Reyk Carstens, Duc Pichel, Bassel Fatloun, et al. Arena-bench: A benchmarking suite for obstacle avoidance approaches in highly dynamic environments. *IEEE Robotics and Automation Letters*, 7(4):9477–9484, 2022.

- [51] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.
- [52] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.
- [53] Yoshiaki Kuwata, Michael T Wolf, Dimitri Zarzhitsky, and Terrance L Huntsberger. Safe maritime autonomous navigation with colregs, using velocity obstacles. *IEEE Journal of Oceanic Engineering*, 39(1):110–119, 2013.
- [54] Jean-Claude Latombe. Potential field methods. In *Robot motion planning*, pages 295–355. Springer, 1991.
- [55] Steven LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.
- [56] Ville Lehtola, Jakub Montewka, Floris Goerlandt, Robert Guinness, and Mikko Lensu. Finding safe and efficient shipping routes in ice-covered waters: A framework and a model. *Cold regions science and technology*, 165:102795, 2019.
- [57] Martin Levihn, Jonathan Scholz, and Mike Stilman. Hierarchical decision theoretic planning for navigation among movable obstacles. In *Algorithmic Foundations of Robotics X: Proceedings of the Tenth Workshop on the Algorithmic Foundations of Robotics*, pages 19–35. Springer, 2013.
- [58] Utkarsh Aashu Mishra, Shangjie Xue, Yongxin Chen, and Danfei Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *Conference on Robot Learning*, pages 2905–2925. PMLR, 2023.
- [59] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.
- [60] Tianyang Pan, Rahul Shome, and Lydia E. Kavraki. Task and motion planning for execution in the real. *IEEE Transactions on Robotics*, 40:3356–3371, Jun 2024.
- [61] Xiao Peng, Chunhui Wang, Guihua Xia, Fenglei Han, Zhuoyan Liu, Wangyuan Zhao, Jianfeng Yang, and Qi Lin. An improved physical information network for forecasting the motion response of ice floes under waves. *Physics of Fluids*, 36(7), 2024.

- [62] Daniel Perille, Abigail Truong, Xuesu Xiao, and Peter Stone. Benchmarking metric ground navigation. In *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 116–121. IEEE, 2020.
- [63] Zisis I Petrou and Yingli Tian. Prediction of sea ice motion with convolutional long short-term memory networks. *IEEE Transactions on Geoscience and Remote Sensing*, 57(9):6865–6876, 2019.
- [64] Anna Petrovskaya and Andrew Y Ng. Probabilistic mobile manipulation in dynamic environments, with application to opening doors. In *IJCAI*, pages 2178–2184, 2007.
- [65] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.
- [66] Megnath Ramesh, Frank Imeson, Baris Fidan, and Stephen L. Smith. Anytime Re-planning of Robot Coverage Paths for Partially Unknown Environments. *IEEE Transactions on Robotics*, 40:4190–4206, 2024.
- [67] Hailin Ren and Pinhas Ben-Tzvi. Learning inverse kinematics and dynamics of a robotic manipulator using generative adversarial networks. *Robotics and Autonomous Systems*, 124:103386, 2020.
- [68] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.
- [69] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347, 2019.
- [70] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [71] Jonathan D Smith, Samuel Hall, George Coombs, James Byrne, Michael AS Thorne, J Alexander Brearley, Derek Long, Michael Meredith, and Maria Fox. Autonomous passage planning for a polar vessel. *arXiv preprint arXiv:2209.02389*, 2022.

- [72] Anthony Stentz. The d* algorithm for real-time planning of optimal traverses. Technical report, Carnegie Mellon University, the Robotics Institute, 1994.
- [73] Mike Stilman and James Kuffner. Planning among movable obstacles with artificial constraints. *The International Journal of Robotics Research*, 27(11-12):1295–1307, 2008.
- [74] Mike Stilman and James J Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. *International Journal of Humanoid Robotics*, 2(04):479–503, 2005.
- [75] Mike Stilman, Koichi Nishiwaki, Satoshi Kagami, and James J Kuffner. Planning and executing navigation among movable obstacles. *Advanced Robotics*, 21(14):1617–1634, 2007.
- [76] Biao Su, Karl Gunnar Aarsæther, and David Kristiansen. Numerical study of a moored structure in moving broken ice driven by current and wave. *Journal of Offshore Mechanics and Arctic Engineering*, 141(3):031501, 2019.
- [77] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. *Advances in neural information processing systems*, 29, 2016.
- [78] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.
- [79] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.
- [80] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.
- [81] Binyu Wang, Zhe Liu, Qingbiao Li, and Amanda Prorok. Mobile robot path planning in dynamic environments through globally guided reinforcement learning. *IEEE Robotics and Automation Letters*, 5(4):6932–6939, 2020.
- [82] Hsueh-Cheng Wang, Siao-Cing Huang, Po-Jui Huang, Kuo-Lun Wang, Yi-Chen Teng, Yu-Ting Ko, Dongsuk Jeon, and I-Chen Wu. Curriculum reinforcement learning from

avoiding collisions to navigating among movable obstacles in diverse environments. *IEEE Robotics and Automation Letters*, 8(5):2740–2747, 2023.

- [83] Jimmy Wu, Xingyuan Sun, Andy Zeng, Shuran Song, Johnny Lee, Szymon Rusinkiewicz, and Thomas Funkhouser. Spatial action maps for mobile manipulation. *arXiv preprint arXiv:2004.09141*, 2020.
- [84] Jimmy Wu, Xingyuan Sun, Andy Zeng, Shuran Song, Szymon Rusinkiewicz, and Thomas Funkhouser. Spatial intention maps for multi-agent mobile manipulation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8749–8756. IEEE, 2021.
- [85] Jimmy Wu, Xingyuan Sun, Andy Zeng, Shuran Song, Szymon Rusinkiewicz, and Thomas Funkhouser. Learning pneumatic non-prehensile manipulation with a mobile blower. *IEEE Robotics and Automation Letters*, 7(3):8471–8478, 2022.
- [86] Fei Xia, William B Shen, Chengshu Li, Priya Kasimbeg, Micael Edmond Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. Interactive gibbon benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 5(2):713–720, 2020.
- [87] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9068–9079, 2018.
- [88] Zifan Xu, Bo Liu, Xuesu Xiao, Anirudh Nair, and Peter Stone. Benchmarking reinforcement learning techniques for autonomous navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9224–9230. IEEE, 2023.
- [89] Taegeun Yang, Minsung Yoon, Jeil Jeong, and Sung-eui Yoon. Navigation among movable obstacles with mobile manipulator using learned robot-obstacle interaction model. In *2nd Workshop on Mobile Manipulation and Embodied Intelligence at ICRA 2024*, 2024.
- [90] Zhiyong Yang, Jun Zong, Yuelong Zhu, Xiuheng Liu, Ran Tao, and Yufeng Yu. River ice regime recognition based on deep learning: Ice concentration, area, and velocity. *Water*, 16(1):58, 2024.
- [91] Linghong Yao, Valerio Modugno, Andromachi Maria Delfaki, Yuanchang Liu, Danail Stoyanov, and Dimitrios Kanoulas. Local path planning among pushable objects based

- on reinforcement learning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3062–3068. IEEE, 2024.
- [92] Kai Zhang, Eric Lucet, Julien Alexandre dit Sandretto, and David Filliat. Navigation among movable obstacles using machine learning based total time cost optimization. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11321–11327. IEEE, 2023.
 - [93] Dengyan Zhong, Na Liu, Lei Yang, Lina Lin, and Hongxia Chen. Self-attention convolutional long short-term memory for short-term arctic sea ice motion prediction using advanced microwave scanning radiometer earth observing system 36.5 ghz data. *Remote Sensing*, 15(23):5437, 2023.
 - [94] Ninghan Zhong, Steven Caro, Avraiem Iskandar, Megnath Ramesh, and Stephen L Smith. Bench-NPIN: Benchmarking non-prehensile interactive navigation. *arXiv preprint arXiv:2505.12084*, 2025.
 - [95] Ninghan Zhong, Alessandro Potenza, and Stephen L Smith. Autonomous navigation in ice-covered waters with learned predictions on ship-ice interactions. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025.
 - [96] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.