AN ABSTRACT OF THE THESIS OF

Seth McCammon for the degree of Doctor of Philosophy in Robotics presented on December 15, 2020.

Title: Topologically-Guided Robotic Information Gathering

Abstract approved: _____

Geoffrey A. Hollinger

Information gathering tasks, such as terrestrial search and rescue, aerial inspection, and marine monitoring, require robotic unmanned systems to make decisions on how to travel within an environment to maximize or minimize a path-dependent information objective function. The distribution of information throughout the environment is the result of various processes, either natural or human-caused, and so this distribution exhibits an underlying structure. Existing information gathering algorithms seek to implicitly exploit this structure by selecting paths which maximize the robot's time in high-value regions. We see an opportunity to improve the performance of robots in these information gathering tasks by explicitly reasoning over the structure of information, allowing robots to plan their information gathering missions more efficiently and effectively. Topological representations provide an elegant way to describe the structure of an environment using descriptors that are defined relative to a set of features in the environment. Since these descriptors are inherently global, they provide a way

for robots to reason directly about their paths within the global context of their operational environments. This additional context enables robotic systems to efficiently plan non-myopically.

To accomplish this goal, this thesis develops four contributions that allow robotic systems to reason about topological structure in field robotics tasks. The first contribution is a method for formalizing topological path constraints using a Mixed Integer Programming formulation to plan. Our second contribution is a system for exploiting expert-provided domain knowledge to track a topological feature using a team of heterogeneous robots. Both of these contributions provide ways to exploit the existence of topological features in the environment to motivate and constrain information gathering tasks. However, these methods require the features to be defined before planning. While methods to identify features exist for well-constructed indoor environments, they do not extend to the less-structured outdoor environments more common in field robotics applications. Our third and fourth contributions address this problem. The third contribution of this thesis is a hierarchical planning algorithm which identifies hotspot regions in an information function and uses them to construct a high-level planning graph, while the fourth is an algorithm for fitting a Topology-Aware Self-Organizing Map to an information function. The benefits of reasoning about the topology of the information field is demonstrated in simulation and field experiments. By incorporating global context about the information gathering task via topology, our methods are able to plan paths that collect more information than a naïve myopic planner. Furthermore, we are able to produce comparable or superior paths more quickly than state-of-the-art planners that do consider the entire path, such as combinatorial branch and bound algorithms.

Topologically-Guided Robotic Information Gathering

by
Seth McCammon

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented December 15, 2020
Commencement June 2021

Doctor of Philosophy thesis of Seth McCammon presented on December 15, 2020.

APPROVED:

Major Professor, representing Robotics

Associate Dean of Graduate Studies of the College of Engineering

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Seth McCammon, Author

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF ALGORITHMS

Chapter 1: Introduction

Increasingly, robots are moving from highly controlled environments, such as warehouses and factory floors, to less structured outdoor environments, such as those encountered by autonomous marine, ground, and aerial vehicles. At the same time, the tasks in which robots are being employed are becoming more complex and open-ended, moving from simple point-to-point navigation around well-defined obstacles to more complex monitoring and inspection tasks. To succeed in these new environments and tasks, robots require more informative representations and new planning algorithms capable of leveraging these representations. Topological techniques provide an alternative to more traditional metric-based planning algorithms by creating a framework for decision making that scales with the complexity of the decision making problem itself, not the size of the robot's operating environment.

For a robotic arm on an assembly line or a mobile robot traversing a factory floor, the robot's workspace can be defined as a metric space: either in terms of Cartesian coordinates or in terms of the joint angles of the robot. In both cases, the robot's task is defined as moving itself from one state to another, minimizing a travel cost function. The solution to this problem is a sequence of states defined in terms of the metric environment that together form a path for the robot to take. There are well-established algorithms including A*, Rapidly Exploring Random Trees, and their countless variants that are capable of finding a solution to the motion planning problem [4]. However,

if we were to instead construct the motion planning problem as a high-level decision making problem, we can simplify the planning problem. Instead of a search over all possible sequences of states in the world, the solution to the motion planning problem is a sequence of key decisions (e.g. "should the robot move to the left or right of a pillar?"). Once these decisions are made, they provide additional constraints that reduce the search space within the original metric planning problem, allowing it to be solved more efficiently without sacrificing quality.

Topological representations enable this high-level decision making by providing a representation of the environment that naturally encodes the major elements of the decision making problem. These representations define the *connectivity* of sectors of the environment while abstracting away metric information, such as the distance between components [5]. In robotics applications, the most common form of a topological map is a graph that is comprised of a set of vertices connected by a set of edges. In the context of a motion planning task, the vertices of the graph represent different locations a robot needs to visit, such as cities, while the edges are categorically different routes the robot might take, such as interstate highways or backcountry trails. Another topological construct that differentiates between different types of trajectories is homotopy classes. These classes group trajectories based on the way that they move through the environment relative to a set of features. Similar to searching over the edges of a graph that connect two vertices, searching over the homotopy classes of trajectories is a search over meaningfully different trajectories.

Information gathering tasks, such as inspection or monitoring, require an additional level of reasoning above and beyond that required in the simple motion planning prob-

lem. In these tasks, a robot needs to move through an environment, gathering useful sensor measurements. These tasks are made even more difficult since they require robots operate in large and complex environments, with features distributed over areas tens of kilometers in size. Because of these challenges, we cannot assume that the robot has access to a highly detailed models of their environment, as a robot arm on an assembly line does. The problem of finding the path that enables the robot to collect the most useful sensor measurements is called the Informative Path Planning Problem (IPPP) [6]. In this thesis, we will explore the potential of topological path planning methods when applied to information gathering problems.

Prior work in topological path planning has concentrated on well-defined topological features created by obstacles in the environment. These techniques have been successfully used to build topological representations of domestic environments [7], used for planning paths for tethered robots [8], and for identifying different classes of trajectories in a robot's workspace [9]. However, for information gathering tasks, particularly in the marine and aerial domains where often there are no obstacles to impose a topology in physical space, the underlying information field will have structure determined by the distribution of information in the environment. Through exploiting this structure by using it to create a topological space, we can improve the efficiency of robots in information gathering tasks.

This thesis develops a framework for incorporating topological information into robotic path planning through task descriptions. These descriptions can be implemented either as constraints, which enforce limits preventing undesirable robot behaviors, or as objective functions, which map a topological objective onto paths being optimized

within a metric space. In doing so, we have developed a framework by which topological information can be embedded within robotic path planning algorithms, enabling it to be utilized in a planning task. However, this ability is predicated on the existence of a set of topological features within the operating zone of a robot or team of robots. Some features, such as obstacles, are readily apparent and provide a partitioning of the robot workspace into distinct topological regions. However for some tasks, such as marine monitoring, the physical environment can be topologically homogeneous. However, physical features (e.g. temperature or salinity) being monitored have structure created by the interaction of physical processes such as ocean currents. By identifying the topology of the underlying system, we can leverage it to improve the quality of paths in an information gathering task.

There are several major advantages to planning in a topological space. First, there can be significant computational benefits. Compared to metric representations, which typically model every point in the environment, topological representations distill a relatively sparse set of key features such as regions in the ocean, rooms in a building, or classes of trajectories in an environment. Thus, the complexity of solving a decision making problem using a topological representation scales with size of the topological representation, rather than with the size of the physical environment. Reasoning over this relatively sparse set of features is more computationally efficient than reasoning over the entire metric space, and by leveraging the topological structure of an environment, a robot can more quickly identify high-quality paths [10].

The second major advantage of topological path planning is that the topology of an environment provides a global descriptor. Since topological paths are defined in relation

to features in the environment without considering the distance to those features, they necessarily are defined relative to all features within the environment. As a result, topological path planners provide an elegant way to use global context about the structure of the environment in a path planning problem [11]. When applied to the IPPP, this global information helps prevent robots from behaving myopically and having their paths stuck in local minima of the information reward function.

The third major advantage of topological path planning is that plans and reasoning developed in a topological framework can be easier to communicate and more understandable by humans. Humans' mental models of their environments appear to be inherently topological, meaning we often think about space as a set of distinct regions and the connectivity of those regions [12]. By utilizing a shared representation, human-to-robot and robot-to-human communication can be improved, increasing trust and estimated capability of autonomous robotic systems [7]. In human-to-robot communication, the human operator of an autonomous system may wish to specify a behavior for the system in relation to topological features in the environment. Such a specification can come in the form of a constraint such as "Do not choose paths that loop around obstacles" or an objective such as "Repeatedly cross the salinity front". In both cases, a strictly metric representation of the environment is insufficient to transform the specification into an actionable path. For robot-to-human communication, studies suggest that overall user satisfaction is increased when the human operator feels that they understand the reasoning behind an autonomy system's behavior [13].

To enable robots to plan using topological specifications and to allow the benefits of topological planning to be realized in marine and aerial domains, the four primary

contributions of this thesis are as follows:

1. A path planning algorithm that utilizes topological constraints in a Mixed-Integer Programming formulation to plan inspection tours for a tethered ROV that are guaranteed to prevent its tether from becoming entangled around obstacles in its environment.

2. A heterogeneous multi-robot planning algorithm that transforms a topological objective into one that can be optimized in a metric space, enabling a team of robots to sample along a topological feature in their environment, such as a salinity front.

3. A method that builds a topological representation of an arbitrary information function by identifying hotspot features within it using the Fast Marching Method [14] and uses it within a hierarchical informative path planner.

4. A method that uses Stochastic Gradient Ascent [15] to exploit topologically diverse trajectories found by partitioning the path space of a robot based on features identified in the information function using a novel Topology-Aware Self Organizing Map.

Taken together, these contributions provide a framework that allows robots to realize the benefits of topological path planning under increasing levels of uncertainty regarding the position, shape, and size of the topological features in the environment, whether they are derived from physical features or features of the information space.

Thesis Roadmap

In Chapter 2, we provide context for the contributions of this thesis through an overview of supporting background literature on which discusses how topology can be incorporated into robot path planning path planning as well as other related works in the areas of robot information gathering.

In Chapter 3, we discuss how topological considerations can be formulated as constraints on a robot's path. These constraints demonstrated through the Non-Entangling Travelling Salesperson Problem (NE-TSP), wherein a tethered vehicle is required to complete an inspection tour of a set of points in an environment with obstacles without causing its tether to become entangled in the obstacles, rendering the vehicle unrecoverable. We show how a topological constraint can be encoded as a component of a Mixed Integer Programming formulation, resulting non-entangling paths.

In Chapter 4, rather than assume we are given a map of the topological features in the environment, we instead exploit human domain experts' knowledge about idealized robot behavior relative to topological features in the environment. We use this knowledge to build information objectives and environmental models that incentivize the desired behavior in a team of heterogeneous robots. We show that planning using a topological environment model improves the performance of the robot team in a front-mapping task. Additionally, we show that planning using these models enables autonomous control of the robot team to match or exceed the performance of human experts on a front-sampling task.

In Chapter 5, we further relax the assumptions on the prior knowledge that the robot

is provided about the topological features in the environment. Instead of relying on expert knowledge about the specific problem domain, we use characteristics of the information gathering problem itself. We develop our Hierarchical Hotspot Information Gathering (HHIG) planner, which constructs a topological graph from local maxima within an information function using the Fast Marching method. This graph can then provides global information about the robot's path that allows us to approach the information gathering problem using a hierarchical strategy: first planning over the graph, then planning within each hotspot.

In Chapter 6, we adopt a different topological representation to better capture the relationship between the robot's paths in an environment and the information function, while addressing some of the limitations of our HHIG planner. We use the ability of Self Organizing Maps to fit a simplicial mesh to an underlying field while leveraging Persistent Homology to adapt the topology of that mesh to match that of the field. Our Topology-Aware Self Organizing Map (TA-SOM) can create a environment representation that partitions the path space into topological trajectory classes that correspond with local maxima in the information function. We then exploit these classes using a Stochastic Gradient Ascent optimizer to more quickly find the globally optimal path. In a set of simulated trials and field experiments, we show that the added global context provided by topological representations allows robots to plan paths that collect more information than other state-of-the-art methods that do not leverage topology.

In Chapter 7, we provide some closing remarks on the contributions presented here, as well as discuss some possible directions for future research to build on and expand the work presented here.

Chapter 2: Background and Related Work

In this chapter, we provide an overview of important concepts related to topological path planning and robot information gathering. Topology is a broad mathematical field, which covers a variety of concepts relating to the study of shape. Of particular interest and relevance to this thesis is the use of topological environmental representations and the use of algebraic topology to specify homotopy classes and homology groups of trajectories that pass through an environment. Taken together, these ideas allow for the construction of sparse yet informative representations that not only improve the performance of robots in many tasks, but also improve the ability of robots to communicate their reasoning in a way that is understandable to humans. In Chapter 2.1, we provide an introduction to the field of topology and how it has been used previously in robotics applications.

We seek to leverage these principles of topology in the context of robotic information gathering tasks. Many field robotics applications, such as inspections, environmental monitoring, or adaptive sampling tasks can be formulated as a general information gathering task. In Chapter 2.3 we summarize the existing state-of-the-art in information gathering planning algorithms and highlight many of the important considerations that we will take into account in our contributions.

2.1 Topology

Topology has been defined as ”the study of shape without reference to distances”, ”the study of rubber sheet geometry”, or ”the study of continuous functions” [16]. More intuitively, topology examines non-metric relationships between discrete features of geometric objects. A classic example of this definition is presented in Fig. 2.1, which shows a coffee mug deforming into a donut (torus). From a topological perspective, the mug and the donut are the same since one may be continuously deformed into another (without requiring gluing or tearing) [17]. This type of equivalence is called a homeomorphism, and it, along with the closely-related concepts of homology and homotopy have many applications in robotics, particularly in path planning. However, these three concepts are not the only useful aspects of topology that can be leveraged for robotic planning.

Abstract graphs, such as those often used in computer science as data structures, are topological constructs. Since a graph is wholly defined by its vertices and their connectivity via a set of edges, it is plain to see how a graph fits beneath the umbrella of topology. Topological representation of graphs has used to describe abstract spaces, such as the belief space of a factor graph in a simultaneous localization and mapping (SLAM) task [18]. In other robotics applications, topological graphs have been used to provide high-level representations of environments for use in planning [10]. These are useful for a variety of reasons, among which are their relative sparsity and the ease with which they can be augmented with semantic information [19]. However, typically these graphs are constructed in domestic environments where walls and doorways easily

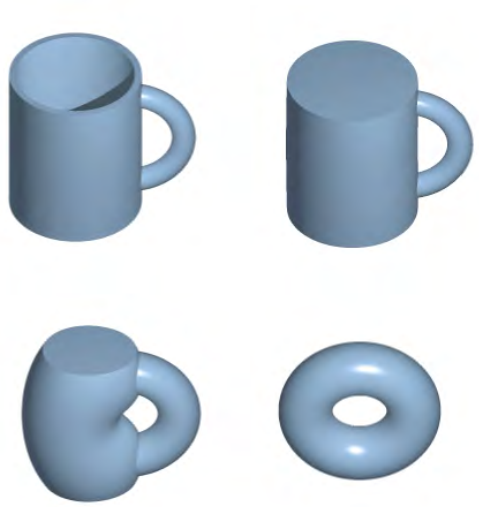
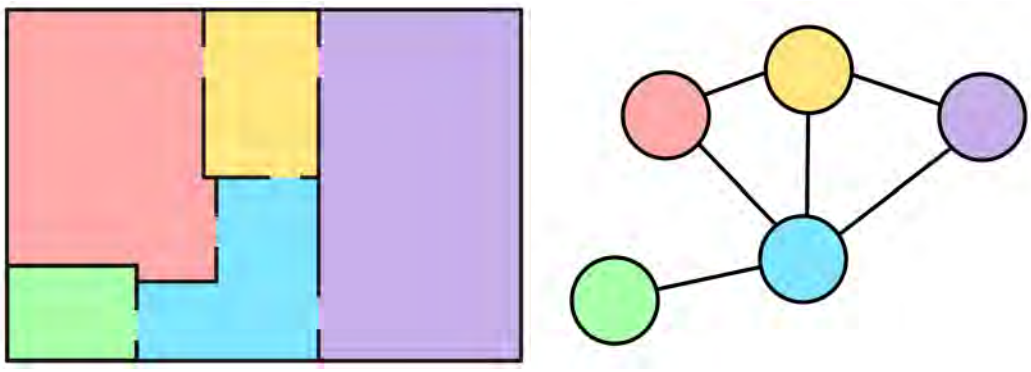


Figure 2.1: A coffee mug and a donut (torus) are homeomorphic, since one may be continuously deformed into the other without any tearing or connecting of components. Source: Adapted from [1].

provide features with which to divide the environment. Existing methods for extracting semantic maps are insufficient in less constructed environments, such as the marine domain.

2.1.1 Homeomorphism, Homotopy Classes, and Homology

Expanding the notion of homotopy equivalence to paths through a space, two paths are homotopy equivalent if one can be continuously deformed into another and they share endpoints. An illustrative example of this definition is shown in Fig. 2.3, where P_1, and P_2 are homotopy equivalent, since one can be continuously deformed into another. However, P_1 and P_2 are not homotopy equivalent with P_3, since the continuous



(a) (b)

Figure 2.2: A metric (a) and topological (b) map of a building. The metric floorplan provides information about all points in the 2D space, while the topological map only preserves information about the rooms and their connectedness via doorways.

deformation is blocked by O_2. Trajectories through a space can then be classified according to their homotopy equivalence into a set of homotopy classes.

Closely tied with the concept of homotopy classes are homology groups of spaces. Homology groups classify spaces based on the number of holes and connections in the space. Returning to the example shown in Fig. 2.1, the coffee mug and the torus both belong within the same homology group, since they are spaces in $\mathbb{R}^3$, pierced by a single hole. Classification by homology is a weaker than classification by homotopy. It is possible for two trajectories to belong to the same homology group, but different homotopy classes, since homotopic equivalence requires an invertible mapping between two paths [16]. However, if two paths belong in different homology groups, they necessarily are in different homotopy classes.

In order for a robot to reason over homotopy classes of trajectories, it first requires a method that enables it to distinguish between them. Homotopy invariants are unique

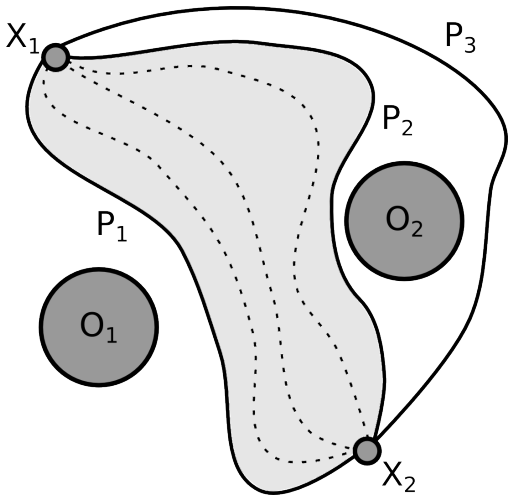


Figure 2.3: Sample trajectories in two homotopy classes. P_1 and P_2 share a homotopy class, since one can be continuously deformed into the other through the shaded area, and they share the endpoints X_1 and X_2. However, P_3 does not belong to the homotopy class shared by P_1 and P_2.

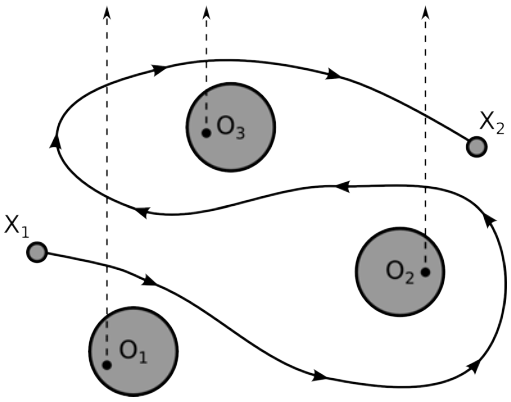


Figure 2.4: Demonstration of h-signature calculation. 1) Representative points and their rays are constructed within obstacles O_1 , O_2 , and O_3. 2) Path between X_1 and X_2 is traced and intersections with rays from (1) are recorded "$O_1, O_2^{-1}, O_1^{-1}, O_1, O_3, O_2$". 3) Eliminating adjacent and opposite terms in H-signature results in "O_1, O_2^{-1}, O_3, O_2".

identifiers of homotopy classes. One early homotopy invariant is the L-value [8], which is defined as

$$L(\tau) = \int_{\tau} \mathcal{F}(z)dz, \tag{2.1}$$

$$\mathcal{F}(z) = \frac{f_0(z)}{(z-\zeta_1) * (z-\zeta_2) * (z-\zeta_3) * ... * (z-\zeta_N)},$$

where ζ_i is a representative point inside obstacle i, z is a point along trajectory τ, and f_0 is an analytic function. Two trajectories that lie in the same homotopy class will share the same L-Value, and two trajectories in different homotopy classes will have different L-Values. An improvement on the L-Value for computing a homotopy augmented graph is the H-Signature [20]. Instead of computing an integral over an entire trajectory, the H-Signature of a point is computed using the line intersections of a trajectory and a set of parallel rays emanating from the representative point in each obstacle. The signature of a trajectory is determined by tracing it from its start to its goal. Each time the trace intersects of the rays, a symbol corresponding to the ray and direction of intersection is added to the h-signature. Without loss of generality, let a positive crossing of the ray emanating from obstacle n be a crossing from left to right, and denoted by O_n. The inverse crossing, from right to left, is denoted as O_n^{-1}. Once the trace is completed, the h-signature is then reduced by removing adjacent elements with inverse symbols along the same ray. This process is repeated until no more elements can be removed. The resulting h-signature is a homotopy invariant, like the L-value, and as such it uniquely identifies the homotopy class of a curve. The process for calculating the h-signature of a trajectory is demonstrated in Fig. 2.4.

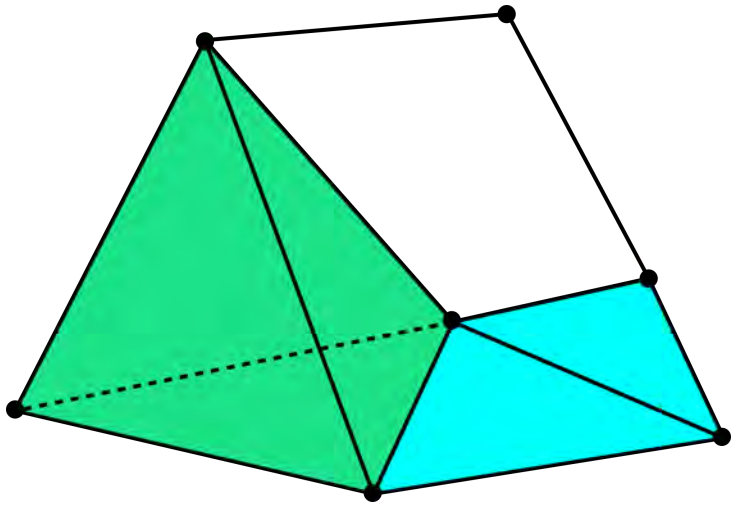


Figure 2.5: Example of a simplicial complex containing seven 0-simplices (black points), twelve 1-simplices (black lines), six 2-simplices (blue triangles), and one 3-simplex (green tetrahedron). Note that each of the faces of the 3-simplex is a 2-simplex.

2.1.2 Simplicial Homology

Computation of homotopy invariants and homotopy augmented graphs require an explicit map of the obstacles and their shapes in order to select the representative points for each obstacle. Additionally, they, like many search-based algorithms, need to perform an exhaustive search of the workspace to propagate the homotopy invariant to all sections of the graph. In many domains, this expansion can be computationally prohibitive, and in other cases, information about the locations of obstacles may not be available. To solve this problem, sampling-based approaches have been developed that leverage simplicial complexes to build a map of where obstacles lie.

A simplex is a n-dimensional generalization of a triangle. A 0-simplex is a point or vertex, a 1-simplex is a line segment or edge, 2-simplex is a triangle, 3-simplex is a tetrahedron, and so on. A simplicial complex is defined as a set of these simplices,

bound by a pair of rules [16]:

1. Any face of a simplex in a simplicial complex is also in the simplicial complex.

2. The intersection of any two simplices in a simplicial complex is either the empty set (i.e. the two simplices do not intersect), or a simplex that is a common face of each simplex.

An example of a simplicial complex containing 0-simplices, 1-simplices, 2-simplices, and 3-simplices can be seen in Fig. 2.5.

Given a simplicial complex, $\mathcal{K}$, trajectories in the space covered by the complex can be mapped onto a set of adjacent 1-simplicies. This set is called a 1-chain, and it is a sub-complex of $\mathcal{K}$ [9]. By comparing two 1-chains within $\mathcal{K}$ that begin and end at the same 0-simplex, we can efficiently determine if two trajectories belong in the same homology group by taking difference between the two 1-chains. If this difference forms the complete boundary of another sub-complex of $\mathcal{K}$ that is of one order higher than the order of the trajectories, then the two trajectories lie within the same homology class. If they do not form a complete boundary, such as if there is a hole the sub-complex, then they lie in different homology classes. Using this method, Pokorny et al. classify trajectories in up to 6-D space into homology groups [9]. Simplicial complexes have also been used construct topological maps of spaces using the communications connectivity of swarms of robots without localization capabilities [21].

While simplicial homology has been used to great success to construct models in environments where large obstacles provide the ability to easily distinguish different homotopy classes of trajectories, it has not been applied to more continuous environ-

ments. We propose combining simplicial homology with a self organizing map (SOM) to achieve better fit of a mesh to the structure of a continuous environment while simultaneously addressing a shortcoming of the SOM, namely its inability to adjust the topology of its network structure to better fit its environment.

2.2 Topological Path Planning

The way that concepts drawn from the field of topology are leveraged in robotic path planning can be broadly divided into two categories. The first of these categories encompasses methods that differentiate the topological classes of trajectories based on a set of features in the environment. The second category contains methods that produce a topological representation of an environment, which is then used in a hierarchical path planning algorithm. Both of these approaches abstract away inconsequential choices a robot must make, allowing it to concentrate it's planning effort on a small set of highly impactful decisions.

2.2.1 Planning with Topological Trajectory Classes

Topological trajectory classes are a way to define a minimal set of unique trajectories through an environment. Early attempts to plan using these classes leveraged visibility graphs to enumerate different trajectory types [22]. Although visibility graphs do not explicitly reason over topological trajectory classes through the computation of homotopy invariants, they still produce a limited set of trajectories that correspond to

the shortest-path boundaries of homotopy classes. More recent work for planning using homotopy classes use homotopy augmented graphs [8]. These graphs augment a typical planning graph, such as one built through a Probabilistic Roadmap (PRM) [23], with a homotopic dimension. These graphs have used both L-values [8] and h-signatures [24, 20, 25]. While prior work considers the homotopy class of paths while planning, it is used as the only constraint in a shortest-path problem. The contributions of this thesis expand on these concepts by integrating it into planning and inspection tasks. We also consider homotopy constraints, along with other task-related constraints, in the Travelling Salesperson Problem.

Computation of homotopy invariants and homotopy augmented graphs require an explicit map of the obstacles and their shapes in order to select the representative points for each obstacle, as shown in Fig. 2.4. Additionally, these algorithms, like many search-based algorithms, need to perform an exhaustive search of the workspace to propagate the homotopy invariant to all sections of the graph. For large domains, this expansion can be computationally prohibitive, and in other cases, information about the locations of obstacles may not be readily available. To solve this problem, sampling-based approaches have been developed that leverage simplicial complexes to build a map of where obstacles lie [9]. These methods have been used to approximate the topology of environments based on limited sensor observations, such as the positions of members of a robot swarm [21].

2.2.2 Topological Environment Representations

Topological mapping techniques have been used to derive abstract representations of environments from metric maps, as shown in Fig. 2.2. A extensive body of work in this area focuses on segmenting indoor environments into regions that correspond with rooms and hallways [26]. A variety of approaches have been proposed for this task, including graph clustering and segmentation [27, 28] as well as Voronoi-based segementation [29, 10]. In the graph-based methods, topologically distinct regions are identified by finding cliques and other clusters of nodes that collectively have low degree, meaning that they are not 'strongly' connected with other portions of the graph. These clusters, therefore are likely to represent topologically distinct regions. The Voronoi-based topological segmentation algorithms leverage Voronoi partitioning [30], to segment an area by creating a set of regions based on a set of seed points. These seed points, in turn, are created by performing an inflation of walls and other obstacles, and selecting branching and end points in the resultant structure.

Once these high-level environmental representations exist, there are a wide range of applications that benefit from the knowledge encoded within them. A natural extension of a topological map is the semantic map. By augmenting the topological regions with semantic information, robots are able to build a human-like understanding of their environments [31]. This representation facilitates interactions with humans [7], since the semantic-topological mental representation closely echoes the mental models that humans build of their surroundings [12]. Topological structure also provides a way for experts to encode sparse prior information that a robot can utilize in order to complete

its tasks. This background information has been shown to speed up robotic coverage planning in domestic environments [32]. In instances where there is no human expert to provide background knowledge, a robot can also leverage past experience to learn the topological structure of environments, and use that knowledge to predict the topological structure of as-of-yet unseen portions of a building [33].

One key issue with these approaches to building topological representations of environments is that they rely on the existence of a well-defined set of features that can be used to define the topology of the space. In indoor environments, obstacles such as walls provide this definition. In less-structured field robotics environments, it is more difficult to determine what features separate topologically distinct regions. Examples of such regions include fronts in salinity and temperature caused by coastal upwelling [34] as well as Lagrangian coherent structures [35]. Both of these create dynamically distinct regions in the ocean, separated by high gradients in temperature, salinity, or ocean currents. One approach to creating a topological representation of these and similar environments is to use a global metric to identify coherent regions within the space that share key features. Typically this process is done using a hand-tuned threshold to isolate areas of interest with isobars in $\mathbb{R}^2$ and isosurfaces in $\mathbb{R}^3$ and higher dimensional spaces [36, 37]. However, thresholding approaches require hand-tuning the threshold parameter that, in turn, requires a significant amount of domain knowledge in order to select the correct value.

2.3 Robot Information Gathering

Many applications for field robotics can be constructed as an information gathering task. These include tracking salinity fronts in the ocean [38], persistent monitoring by UAVs [39], identifying harmful algae blooms [40], or locating the source of a chemical spill [41]. In these tasks a robot or team of robots is deployed in an environment, and a reward function, described generically as 'information', is distributed across an operational environment. The robots are tasked with maximizing the amount of information collected along their path given a budget constraint. This problem can be written using the following equation:

$$P^* = \underset{P_i \in \phi}{\operatorname{argmax}} I(\mathcal{P});\ s.t.\ C(P) \leq B, \tag{2.2}$$

where P^* is the optimal path, selected from the set of all paths, ϕ, that maximizes the information utility function, $I(P)$, such that the cost of that path, $C(P)$, is less than the budget constraint, B. The total number of possible paths, $|\phi|$, increases exponentially as B increases, meaning that searching exhaustively over ϕ for P^* requires computation exponential in B. As a result, informative path planning has been shown to be NP-Hard [42]. This difficulty greatly limits the scope of informative path planning problem instances that can be solved exactly. To mitigate this difficulty, many different types of techniques have been developed to provide approximate or sub-optimal solutions. Simple methods include performing a pre-determined exhaustive coverage of the space. Typically these will use a lawnmower pattern to achieve complete and uniform sensor coverage [43]. One issue with such an approach are that information is not evenly

distributed in the environment. In addition, exhaustive coverage is time-consuming, and often incompatible with a robot's budget constraint, which is usually derived from battery capacity. Another naïve approach is to use a greedy planner, which takes a series of locally-optimal actions. In a highly-idealized information gathering task where the objective function is submodular and path constraints are ignored, greedy planners have a proven lower performance bound of 63% of optimal [44]. However, in practice, where path constraints are considered, the myopic behavior of a greedy information gathering approach is problematic, as they can quite easily become trapped in local minima. As a result, the greedy algorithm can perform significantly worse than the optimal plan, without any theoretical bounds.

Non-myopic planners instead consider the whole path instead of a single action, and therefore are required to balance exploration with exploitation in information gathering tasks. These approaches include sampling-based methods, such as Rapidly exploring Information Gathering [42], Monte Carlo Tree Search (MCTS) [45, 46], and Baeysian Optimization [47], which use stochastic sampling to efficiently cover the space of possible solutions. Rapidly exploring Information Gathering, draws inspiration from the Rapidly exploring Random Trees algorithm [48], and draws samples from the environment and connects them to an expanding tree of possible paths. The information value of trajectories along the tree can be evaluated, and by pruning paths that lack promise, the overall size of the tree can be made more manageable. One disadvantage of the Rapidly exploring Information Gathering algorithm is the need to evaluate partial paths. To address this problem, MCTS has been adapted to information gathering tasks [46]. MCTS, similar to Rapidly exploring Information Gathering, samples possible actions

that the robot could take. However, the key difference in MCTS is that instead of evaluating partial paths, it uses a default policy to perform a rollout, completing a partial path in the tree. In doing so, MCTS only has to evaluate its objective function on full paths.

An alternative to the sampling-based approaches is to use one of a variety of deterministic anytime planners. These planners, such as branch and bound [49], or Mixed Integer Linear Programming [50, 51, 39], quickly produce a viable candidate solution that iteratively improves as the algorithm searches through the space of possible solutions. By employing pruning techniques and heuristics, these methods can accelerate the planning process by avoiding searching through the large areas of the problem domain that do not lead to optimal paths. This class of algorithms as well as the discussed sampling methods are considered anytime algorithms, meaning that once they have a viable solution, their computation can be stopped at any point in time and the current best solution returned. In field robot operations, this trait is highly desirable since plans for autonomous vehicles must be ready when the vehicles are available to receive the plan. However, information gathering planners must search through the entire space of paths. Using topological representations, we can identify classes of trajectories likely to contain high-quality paths. Additionally they can be leveraged improve the rate of exploration in field robotics domains by identifying unexplored regions and trajectory classes, as they have been used to speed up exploration in domestic environments [32].

Often we are interested in planning informative paths in unknown, partially-known, or dynamic environments, meaning that the globally optimal path may pass through unexplored locations, or the value of currently known locations will become unknown as a result of dynamic processes. One of the key challenges in informative path planning in

such an environment, is balancing the exploitation of previously-collected observations with the need to explore the environment and collect new observations in the unknown regions. Typically this tradeoff is handled via a weighted sum of separate exploration and exploitation objectives [52], where a parameter is used to explicitly trade off between the two conditions. A similar approach that can be used when the environment is modelled using a probability distribution is the Upper Confidence Bound [47]. In this approach, the mean of the probability distribution is used as the exploitation term, reflecting the current best guess of the sample's value given all previous observations, while the distribution's variance provides an estimate of the value of exploring. As before, these components are combined using a weighted sum to produce the single estimate of the value at a point, which can easily be used for optimization.

As laid out in Equation 2.2, the information gathering problem is, at its heart, a search problem, requiring a planner to search over the space of possible paths for the optimal one, and searching over this space is combinatorial in the number of actions that a robot can perform. Through the use of topological representations, both of these challenges can be addressed. Topological representations are sparser and simpler than their metric counterparts, meaning that there are fewer paths to consider. Furthermore the only actions available to a robot in a topological graph are meaningful ones. As a result, searching over topological spaces is significantly more feasible than exhaustive search over metric spaces. Additionally through the addition of domain expert knowledge, topological representations can be used to prune out large sections of the environment that are not likely to contain quality paths, further improving the performance of information gathering algorithms.

Chapter 3: Planning with Topological Constraints

In this chapter we present a method for integrating topological constraints into a robotic inspection task. We do this by adopting a representation that allows the robot to explicitly reason about the environment. This representation allows us to formulate the topological constraint as a constraint in a Mixed Integer Programming formulation. This constraint, along with other non-topological constraints, is then used to plan an inspection tour of a set of Points of Interest in an offshore infrastructure inspection task for a tethered vehicle, such as the Seabotix vLBV300 vehicle shown in Fig. 3.1. The main challenge in completing this task with a tethered vehicle is the risk of entanglement presented by the tether. While the topological features used in this chapter are derived from known physical obstacles in the environment, in subsequent chapters we will expand on this to also consider features derived from the information space of the robot.

Our proposed technique leverages homotopy augmented graphs to enumerate different homotopy classes in the path-space for a tethered ROV. For a more detailed discussion of the construction of homotopy augmented graphs, we refer the reader back to Chapter 2.1.1. This enumeration allows us to specify topological constraints, such as a non-entangling, on the vehicle's motion that can be incorporated into the vehicle's path planning as it attempts to identify a minimum-cost inspection tour.

We offer some background on Mixed Integer Programming and other planning techniques for tethered vehicles in Chapter 3.1. In Chapter 3.2, we formulate the inspection

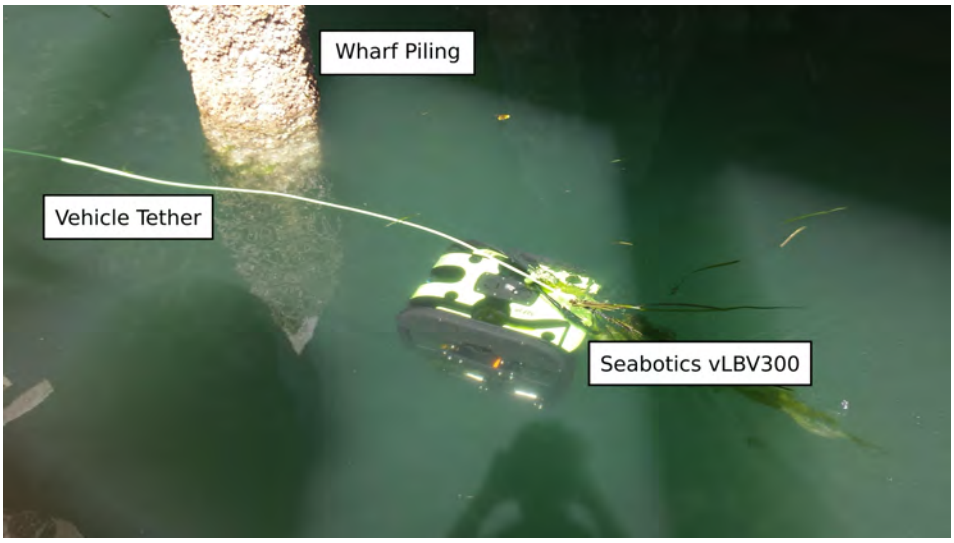


Figure 3.1: Seabotix vLBV-300 Vehicle completing a wharf inspection in Newport, Oregon. Using our method, the robot plans a path to inspect the wharf's eight pilings in a non-entangling manner.

task as an extension of the Travelling Salesperson Problem (TSP) [53] by incorporating the non-entangling topological constraint, resulting in the Non-Entangling Travelling Salesperson Problem (NE-TSP). We propose our solution to this problem in Chapter 3.3, as well as a heuristic for simplifying the robot's travel graph. Finally, in Chapter 3.4, we compare our algorithm with a stochastic optimization method and a greedy approach as well as against a human in a set of field trials. Versions of this work have been previously published in a journal paper [54], [55], and [56].

3.1 Background

Early work planning for tethered vehicles use visibility graphs to plan optimal paths for a tethered robot [22]. However this method requires polygonal obstacles, and is

limited in its applications by the computational complexity of computing the visibility graph. One potential solution to this problem is to consider the path of the tether topologically, using homotopy augmented graphs [8]. Prior work using this topological representation has focused on the length of the tether as the primary constraint in planning paths [25]. We consider the additional constraint of avoiding entanglement as the robot plans through multiple goal points. While it is possible for the robot to reverse its path to avoid any entanglement, as was done in [57] to obtain maximum coverage of a space by a tethered robot, such a behavior can lead to lengthy paths, which reduce the overall area that can be inspected in a reasonable amount of time.

A powerful tool for solving these types of constrained path planning tasks is Mixed Integer Programming (MIP). Derived from linear programming, an instance of a MIP is constructed by specifying an objective function to be minimized or maximized, along with a set of constraints formulated as inequalities [58]. In a MIP problem instance, all decision variables are constrained to be integers, which results in the solution to MIP problems falling into the class of NP-hard problems [59]. Despite this theoretical limit, many strong heuristics have been developed for solving MIPs, and are incorporated into efficient off-the-shelf solvers, using algorithms like branch-and-bound [6] to compute optimal solutions.

In the context of robotic planning, MIP-based methods have been employed in orienteering-style extensions of the Travelling Salesperson Problem (TSP) [51, 39]. MIP-based planning approaches have also been employed to plan maximally informative paths for autonomous aerial and marine vehicles while avoiding obstacles [60, 50].

3.2 Problem Definition

To plan non-entangling paths through the world, we first must define an entanglement. Since a tour of goal points consists of a loop starting and ending at g_1, obstacles in the world may be divided into two sets: those contained within the bound of the tour (the interior set), and those outside the bound (the exterior set). Any obstacles in the interior set are considered to be entangled in the tether, while obstacles in the exterior set are non-entangled. This process is illustrated in Fig. 3.2, where a path modification moves O_3 from the interior set to the exterior set. There exists a simple test for whether a given trajectory is entangled in any obstacles. We compute the h-signature of the trajectory by combining the h-signature of each of its sub-paths, and then reducing the combined h-signature as described in Chapter 2.1.1. If the resulting h-signature is empty, the trajectory is non-entangling.

3.2.1 Non-Entangling Travelling Salesperson Problem

The problem of planning optimal non-entangling paths can be seen as an extension of the TSP, with the additional constraint that the path does not cause the tether to entangle any obstacles. This extension is the Non-Entangling Travelling Salesperson Problem.

An instance of the NE-TSP consists of a map of the world that contains n obstacles $O = \{o_1, o_2, ..., o_n\}$. To allow for application to many representative environments, each obstacle o_i is defined as a vertical projection from a circle on $\mathbb{R}^2$ to $\mathbb{R}^3$. The map also contains m goals $G = \{g_1, g_2, ..., g_m\}$ where $g_i \in \mathbb{R}^3$. The initial deployment point of the robot is also its first goal g_1. A trajectory T is a complete circuit of these goal

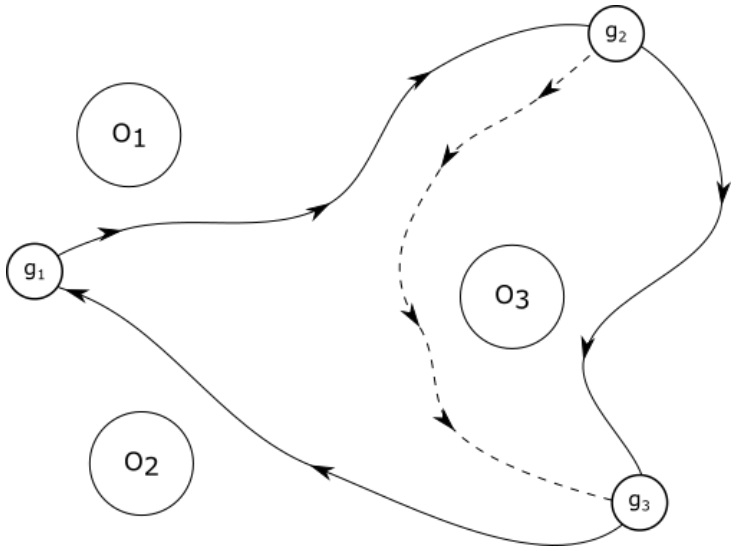


Figure 3.2: An example of a trajectory modification that avoids tether entanglement. By modifying the path subsection between g_2 and g_3 (shown with a solid line) to the dotted line, the overall path length may be increased; however, the entanglement with O_3 is eliminated, as O_3 is no longer inside the bound of the trajectory.

points, ultimately returning to the initial deployment point after visiting the final point in T.

A solution to the NE-TSP consists of a trajectory T^* that satisfies

$$T^* = \underset{T}{\operatorname{argmin}}\{L_T | \textit{h-signature}(T) = \emptyset\ , |T| = m\}, \tag{3.1}$$

where L_T is the length of the path needed to traverse all points in T and $|T|$ is the number of elements in T. T^* is the minimum-length, non-entangling trajectory that passes through all $g_i \in G$.

To compute the optimal solution to this problem, we propose a Mixed-Integer Programming model that can compute T^* given a set of goals G and obstacles O.

3.3 Method

3.3.1 Homotopy Augmented Graph

To guarantee the construction of non-entangling paths, we construct a homotopy augmented graph based at the robot's deployment point. Formulated by Bhattacharya et al. [20], a homotopy augmented graph allows the robot to plan the shortest path between two points in a given homotopy class. The homotopy augmented graph, $\mathcal{G}_{Aug} = \{V_{Aug}, E_{Aug}\}$, is constructed by augmenting a prior graph $\mathcal{G} = \{V, E\}$ with another dimension, h, which indicates the homotopy class of the path between a given vertex in $\mathcal{G}_{Aug}$ and the base point of the graph. Thus a vertex $v_i \in V_{aug}$ consists of the spatial location of v_i, as well as the homotopy class of the path between it and the base node. Using the vertices in $\mathcal{G}_{Aug}$, we can construct an augmented trajectory T_{Aug} that augments T with the homotopy classes of each of its elements.

We build on the idea of the homotopy augmented graph [20] by employing an extension of the Probabalistic Roadmap (PRM*) [61, 23] in place of the grid-based graph. PRMs provide a probabalistically complete graph-based map of an environment by taking a number of samples of the free space, and connecting nearby samples with traversable edges. PRM* extends the PRM by providing a principled method for determining which pairs of samples should be connected with edges [61]. Leveraging PRM* allows us to more easily span a 3-Dimensional environment, such as the underwater domain. However, since obstacles are projected from $\mathbb{R}^2$ to $\mathbb{R}^3$, we can compute entanglements and homotopy classes on the projection in $\mathbb{R}^2$, while distances between points and the resultant trajectory for the robot remain in $\mathbb{R}^3$.

By constructing the homotopy augmented graph using the AUV's deployment point as a starting point, we can ensure that each point in the graph is reachable by the robot. This constraint is enforced by only adding points to the graph that are within range of the robot's tether. Furthermore, during the construction of the homotopy augmented graph, we add a non-looping constraint, thus each path through the graph is both feasible and non-entangling.

Once constructed, paths can be planned over the homotopy augmented graph using standard graph-based planning algorithms, such as A*, which are both complete and optimal. This property, combined with the probabilistic completeness guarantees of PRM*, ensures that our proposed method retains the same probabilistic completeness guarantees as PRM*. Additionally, any paths generated on the homotopy augmented graph will be optimal with respect to the graph.

3.3.2 Mixed Integer Model

To compute the optimal non-entangling path for the robot over the PRM*, we model the NE-TSP with a Mixed-Integer Program. For each goal point $g_i \in G$, there is a corresponding set of all homotopy classes that reach the goal without exceeding the tether length constraint $\mathcal{H}_i = \{h_1, h_2, ...h_{z_i}\}$ with $z_i \geq 1$. Each $h_j \in \mathcal{H}_i$ corresponds to an augmented goal vertex $v_{g_i,h_j} \in V_{Aug}$. We define the set of homotopy augmented goals V_h as the union of these vertices for all $g_i \in G$:

$$\forall g_i \in G, V_i = \{v_{g_i,h_1}, v_{g_i,h_2}, ..., v_{g_i,h_{z_i}}\},$$

$$V_h = \cup_{i=1}^{m} V_i,$$

where $T_{Aug} \subseteq V_h$.

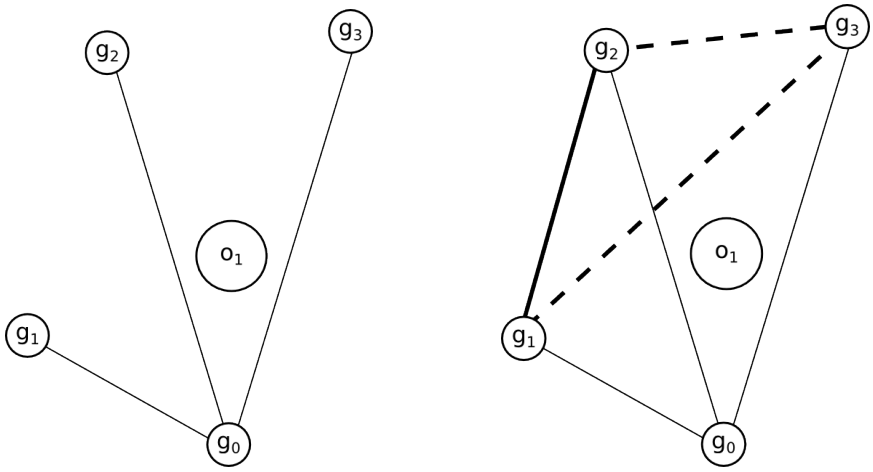
To fully define our MIP model, we need to determine the order that the goals are visited and by which homotopy class each goal is visited. We introduce two sets of binary decision variables, one to describe each of these two determinations. To solve for the homotopy class h_j of each goal, g_i, for each v_{g_i,h_j} let there be a corresponding $x_{i,j}$, where $x_{i,j} = 1$ if and only if v_{g_i,h_j} is visited by the robot and $x_{i,j} = 0$ otherwise.

The second portion of the solution, the order in which the goals are visited, is developed by determining which edges $e_{g_i,h_j,g_l,h_k} \in E_{Aug}$ are included in T. An edge e_{g_i,h_j,g_l,h_k} is the path segment between two vertices, v_{g_i,h_i} and v_{g_l,h_k}. Let $y_{i,j,k,l} = 1$ if and only if the robot travels the edge e_{g_i,h_j,g_l,h_k} and $y_{i,j,k,l} = 0$ otherwise. Each edge also has a corresponding distance d_{g_i,h_j,g_l,h_k}, which is defined as the shortest-path distance in the homotopy augmented graph between v_{g_i,h_i} and v_{g_l,h_k}.

To complete its tour of G, the robot will visit each $g_i \in G$ exactly once. Correspondingly, there is only one homotopy class at g_i that the robot will visit. This constraint can be modelled with the following summation:

$$\forall i, \sum_{j=1}^{z_i} x_{i,j} = 1. \tag{3.2}$$

In the classic TSP model, for each vertex in the graph, there is an incoming and outgoing edge. This constraint can be captured in a MIP model with a simple degree-2 constraint that requires two unique edges to connect each vertex. However, in our model, this simple constraint fails when presented with the homotopy augmentation at



(a) Initial Graph Edges

(b) Non-Entangling Graph Edges

(c) Complete Graph

(d) TSP Solution

Figure 3.3: Heuristic Map Generation method. The shortest edges between the goals $\{g_1, g_2, g_3\}$ and the base point, g_0 are added to the map in 3.3a. In Fig. 3.3b, direct paths (shown in bold) are added, while indirect paths (dashed) are replaced with the bold paths shown in Fig. 3.3c. Finally, the TSP tour of the resulting map is shown in 3.3d.

each vertex. Since not every vertex v_{g_i,h_j} will be included in the final solution, a vertex may have either degree-0 or degree-2, depending on whether or not it is visited during the tour. Furthermore, the homotopy class of both the incoming and outgoing edges must be the same at v_{g_i,h_j} for the trajectory to be continuous. This constraint is modeled by:

$$\forall i, \sum_{k,i\neq k}^{m} \sum_{j=1}^{z_i} \sum_{l=1}^{z_k} y_{i,j,k,l} \times x_{i,j} = 2. \tag{3.3}$$

Since only one $x_{i,j}$ for a given i is a part of the solution, as is given in the constraint shown in Equation 3.2, Equation 3.3 ensures that only edges that correspond with $x_{i,j}$ can be used. The first term enforces the degree-2 constraint, limiting the total number of edges connecting to a given node, while the second term ensures that any v_{g_i,h_j} can have either degree-2 or degree-0.

The final constraint in the MIP model eliminates subtours, ensuring that the solution consists of exactly one tour that visits each goal rather than multiple disjoint subtours. We implement this constraint as follows:

$$\sum_{i,k,i\neq k}^{m} \sum_{j=1}^{z_i} \sum_{l=1}^{z_k} y_{i,j,k,l} \leq |S \cap T_{Aug}|, \forall S \subset V_h, S \neq \emptyset, \tag{3.4}$$

Since it is impractical to compute all proper and nonempty subsets S of V_h, in practice the constraint modelled in Equation 3.4 is implemented using a lazy constraint. When a potentially valid solution is found, we determine whether it is a single tour, or multiple disjoint subtours. If the solution does contain subtours, a constraint is added disallowing the potential solution as a valid one.

To solve the MIP model, we employ the Gurobi Mixed Integer Solver [62]. The

Gurobi solver utilizes a branch-and-bound method to converge to the optimal solution to any MIP. The solver also has the anytime property, meaning that at any point before convergence to the optimal solution, the incumbent solution (i.e. the best potential solution found) will be a valid, though sub-optimal, solution to the model. In Chapter 3.4, we compare the optimal solution to the anytime solution generated after 2 minutes of computation. The optimality of our method is by construction. Equations 3.2 - 3.4 fully define the NE-TSP. As a result, the optimal solution on the homotopy augmented graph to these constraints also is the optimal solution to the NE-TSP.

3.3.3 Reduced MIP Heuristic

Solving the NE-TSP optimally requires a search over not only the combinatorial space of goal point orderings, but also the space of homotopy classes. To reduce this search space, we propose a heuristic that selects homotopy classes that are likely, though not guaranteed, to be a part of the optimal solution.

We accomplish this reduction of the search space by creating a subgraph of the homotopy augmented graph. This process is shown in Fig. 3.3. The graph is initialized with vertices at each $g_i \in G$. The shortest path to the base point, g_0, is computed, along with its corresponding h-signature, shown in Fig. 3.3a. Then, for each other pair of vertices, we attempt to construct a direct connecting edge if the direct edge shares a homotopy class with the path between the vertices that passes through g_0. If no such edge exists, shown by the dashed lines in Fig. 3.3b, an indirect edge that corresponds with the shortest path that does share the same homotopy class as the path through g_0 is

added.

With this reduced graph, we remove the need for the MIP model to make a decision about which homotopy class to use for a given goal, reducing the overall search space the solver will have to search. The computation time is further reduced by eliminating a set of decision variables and their corresponding constraints. The constraint defined in Equation 3.2 is eliminated entirely, and the constraint defined by Equation 3.3 is reduced to a simple linear constraint. By computing the optimal TSP solution over the reduced graph, we compute a near-optimal solution to the NE-TSP.

The path that results from this method is still non-entangling, since no obstacles are contained within each loop in the graph. By only adding edges, direct or indirect, to the graph if they share a homotopy class with a known, non-entangling path, we guarantee that any complete tour in the graph will be non-entangling.

3.3.4 Simulated Annealing

The final method we examine to solve the NE-TSP is simulated annealing. Simulated annealing is a stochastic optimization algorithm, which performs search in multidimensional space and is robust to entrapment in local optima [63]. Initialized with some random state, x, at each iteration of the algorithm, a successor state x' is created by mutating x through some function. This successor state is compared to the previous state with an evaluation function. If the mutated state has the higher score, it becomes the new state. If it has a lower score, it becomes the new state with probability:

$$p = e^{-(s-s')/\theta}, \tag{3.5}$$

where s and s' are the scores of the state and mutated state, and θ is a scaling factor that decreases with subsequent iterations. To begin, x is initialized with a randomly generated T_{Aug}, which is a random ordering of the goal points and their corresponding homotopy classes.

During the mutation step of the simulated annealing process, the first point in this trajectory remains fixed, reflecting the assumption that the robot is tethered to a fixed base station. At each iteration of the optimization process, a trajectory can undergo one of the two types of mutations, chosen at random. The first of these, goal-swapping, swaps the order of two goals on the trajectory:

$$T' = \{g_1, ..., g_{i-1}, \mathbf{g_j}, g_{i+1}, ..., g_{j-1}, \mathbf{g_i}, g_{j+1}, ..., g_m\}.$$

This mutation can either raise or lower the overall trajectory length and entanglement of the path. To ensure that the resultant path is entanglement-free, we use a second method of mutation, path-inversion. During path-inversion, the homotopy class of one element in T_{aug} is changed. The result of a path-inversion mutation is shown in Fig. 3.2.

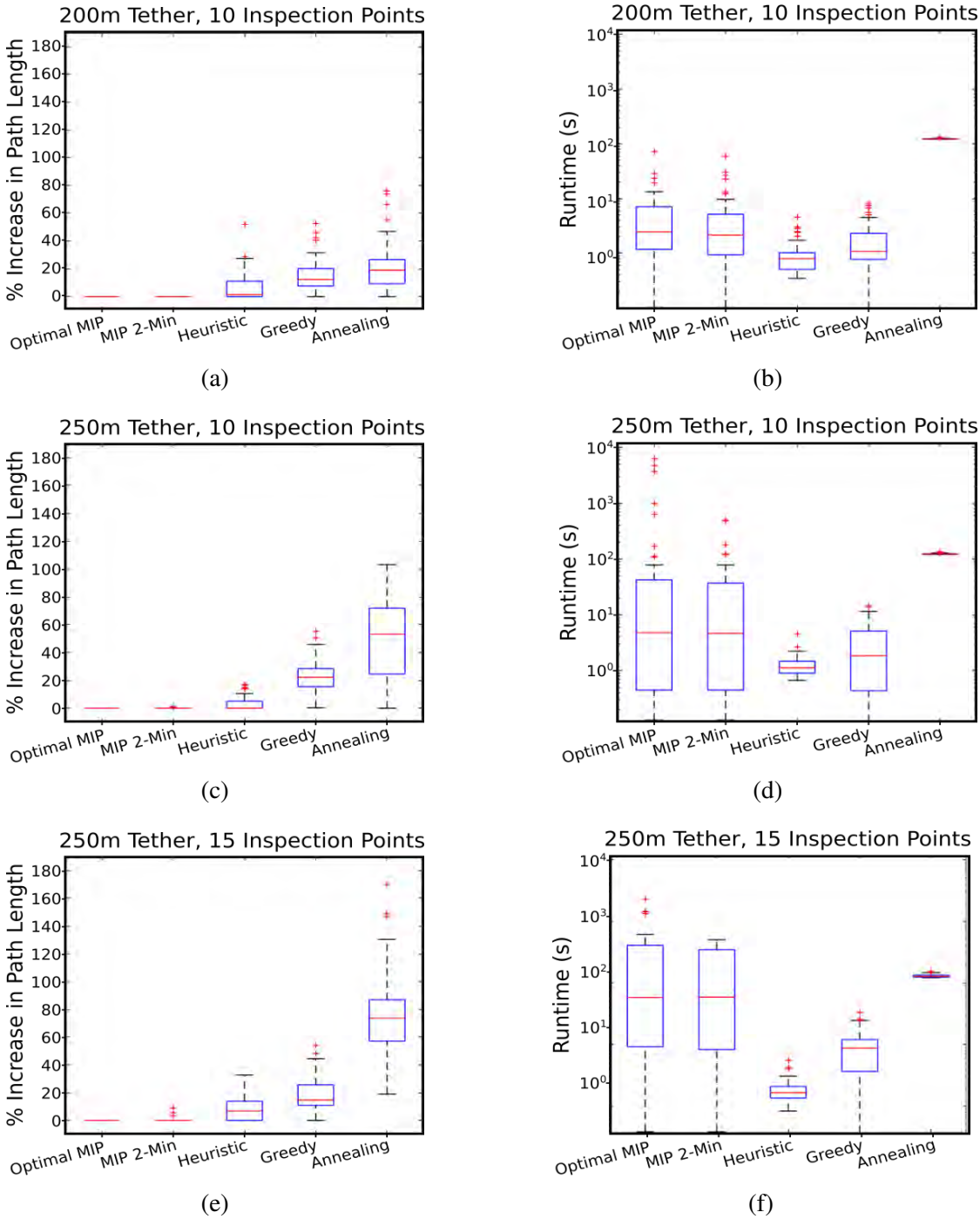


Figure 3.4: Comparison of path length and computation time. Note the logarithmic axes on computation time plots. Additionally, the times shown here includes the time taken to compute a distance matrix between all potential goal points, and so can sometimes appear to slightly exceed 2 minutes of computation time.

3.4 Results

3.4.1 Simulations

We tested our method in a series of simulated tests, comparing the optimal MIP solution to the anytime MIP solution generated after 2 minutes of computation, the simulated annealing solution, and our heuristic solution. For each method, aside from the optimal MIP solution, the computation time of the solution was limited to 2 minutes. We also compare to a greedy-backtracking method, in which the robot iteratively travels the closest point that does not violate the tether length constraint. Because of the behavior of the tether, this point may not be the closest point in Euclidean space. After travelling to the final point, the robot returns to the start location by retracing its path, which ensures that the resulting tour will be non-entangling.

The results of these simulations are shown in Fig. 3.4. For each set of tether length and number of goal points, the methods were compared over 20 randomly generated worlds. In each of these worlds, up to 20 circular obstacles are randomly placed in an environment 500 m per side. The homotopy augmented graph is constructed on a PRM* built with 1000 samples, taken uniformly over the free space, with the base point randomly selected. The goal points were randomly selected from coordinates accessible within the homotopy augmented graph. Since each method requires the use of the homotopy augmented graph, the construction time of this graph is not included in the overall planning time. Simulations were done using Python on a Quad-Core Intel i7-2620M laptop processor clocked at 2.70GHz with 8GB of RAM.

In Fig. 3.4a to Fig. 3.4e we show a comparison of path lengths between the methods

Table 3.1: Comparison of computation time for optimal MIP solution. Each element contains average time to convergence and number of trials out of 10 total completed in 5 minutes (in parentheses).

	Tether Length			
# Goals	200 m	250 m	300 m	350 m
5	0.16 s, (10)	0.66 s, (10)	2.43 s, (10)	8.47 s, (10)
10	0.65 s, (10)	12.73 s, (10)	47.86 s, (7)	74.73 s, (8)
15	8.07 s, (8)	4.69 s, (2)	186.48 s, (3)	115.28 s, (3)
20	16.20 s, (9)	40.08 s, (2)	0.80 s, (1)	65.57 s, (1)

of computing a non-entangling path. In all three tests, the MIP-based methods (optimal MIP, MIP 2-Min anytime, reduced MIP heuristic) outperform simulated annealing and greedy methods. Both MIP 2-Min anytime and heuristic maintain an average path length within 5% of the optimal path length found by allowing the MIP solver to converge. As the number of goals and the tether length increase, performance across all 4 methods begin to degrade. However, it is apparent that the MIP-based methods maintain their level of performance in the larger environments far better than either the greedy or simulated annealing approach. Though initially close, as the environment gets large, the greedy method begins to outperform simulated annealing. This change can be attributed to the restricting of simulated annealing to only 2 minutes of search. As the search space expands with more goal points and their corresponding homotopy classes, simulated annealing is able to explore proportionally less of that space, and so is less likely to find a short path.

3.4.2 Computational Performance

Computing the optimal solution to the Travelling Salesperson Problem, and, by extension, the NE-TSP is NP-Hard, meaning that no polynomial time algorithm to compute the exact solution exists (unless P = NP). The NE-TSP is made even more difficult, since the number of variables in our MIP model scales not only with the number of goal points, but also with the length of the tether and the number of obstacles in the environment. A longer tether and more obstacles allows the robot to reach the same goal point in more homotopy classes. The effects of this scaling can be seen in Fig. 3.4b to Fig. 3.4f, where as the number of goal points and the length of the tether increase, the computation time for the optimal MIP solution increases exponentially. This increase in computation time is especially clear in Fig. 3.4f, where the longest computation time for the optimal MIP solution took over 12,000 seconds.

To evaluate the viability of our method in a practical application, we evaluated the computation time for problems with increasing numbers of goal points and with increasingly longer tethers. The results of this experiment are shown in Table 3.1. Each element in the table contains the average time to convergence for the optimal solution and the number (out of 10) of paths for which the MIP solver was able to compute the optimal solution (shown in parentheses). Computing the optimal solution remains feasible for shorter tethers (i.e. $\leq$ 200 m) or for fewer than ten goal points. However, once the problem expands beyond this size, our approximation heuristic should be used.

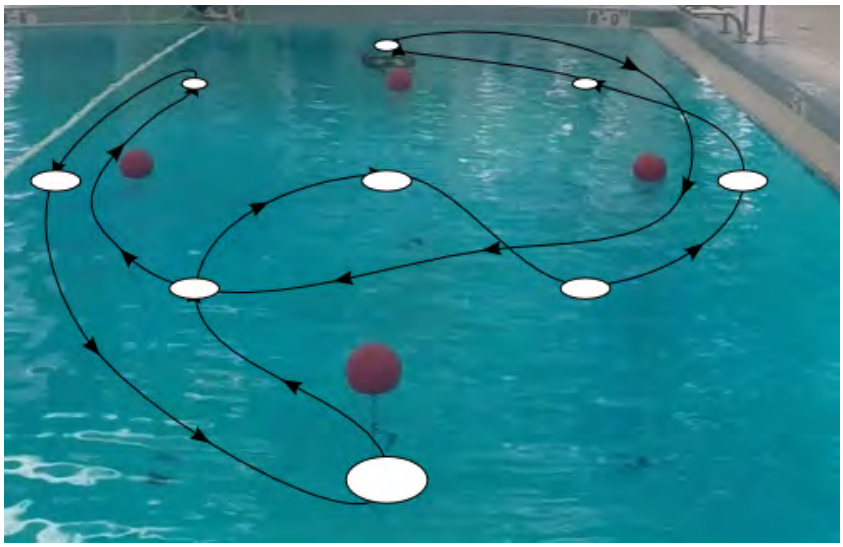
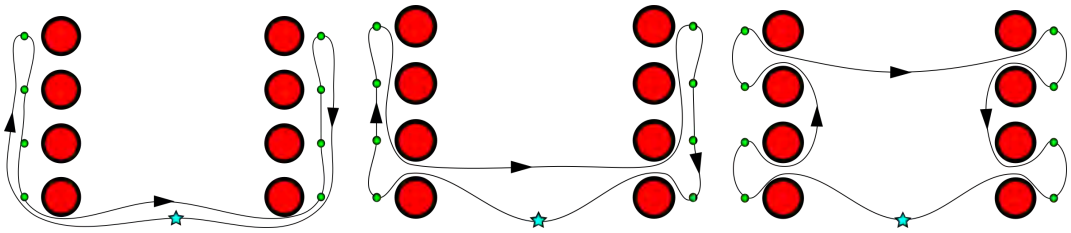


Figure 3.5: An example obstacle and goal layout for a tethered vehicle. The white circles indicate goal locations (all of which lie on the water's surface). The red buoys act as obstacles and indicators of entanglement. The black line shows the planned path for the AUV, and the direction of travel along that path.

3.4.3 Pool and Field Tests

To show that the tether behaved as expected, we conducted a set of pool tests to ensure that the paths we generated were non-entangling when executed on a tethered vehicle. We implemented the non-entangling planner on a Seabotix vLBV-300 underwater vehicle [64] equipped with the Greensea INSpect GS3 Inertial Navigation System, a Teledyne Explorer DVL, and a Tritech Gemini multibeam sonar. The SeaBotix vehicle can be controlled via a series of waypoints provided through a Robotic Operating System (ROS) interface with a command station [65]. Using the non-entangling planner, the vehicle planned paths around a set of buoys, shown in Fig 3.5, and was able to successfully execute them without becoming entangled.

Figure 3.6: Oregon State University Ship Operations Pier in Newport, Oregon used for field trials. The robot must visit each of the inspection locations indicated by green dots while avoiding becoming entangled by the wharf pilings shown as red circles. The robot is deployed from the base station located at the blue star.



(a) Systematic Path 1 (b) Systematic Path 2 (c) Optimal MIP Path

Figure 3.7: Visualization of the three paths for dock inspection task.

Table 3.2: Comparison of path lengths for hand-generated systematic paths and optimal MIP path for a wharf inspection task.

	Systematic Path 1	Systematic Path 2	MIP Path
Path Length	65.8 m	58.8 m	56.6 m

To represent an offshore inspection task, we deployed the vehicle from the Oregon State University Ship Operations pier in Newport, Oregon and conducted an inspection of the wharf's pilings. The vehicle can be seen inspecting a piling in Fig. 3.1. While beneath the wharf, the vehicle was subject to a current of up to about 2 knots. The inspection task involved maneuvering to each of 8 pilings and pausing to inspect them. Two systematic paths were also hand-generated for non-entangling inspection tours of the same points. In this trial, the MIP path outperformed both systematic paths. The path lengths of each inspection tour are shown in Table 3.2, and a visualization of the paths that the robot took is shown in Figs. 3.7a - 3.7c.

3.5 Conclusion

In this chapter, we introduced the Non-Entangling Travelling Salesperson Problem and presented a Mixed Integer Programming model that can compute the optimal solution for a tethered robot. Leveraging homotopy augmented graphs, we can maintain a non-entangling guarantee on all paths generated. To improve computation time we developed a heuristic for selecting good homotopy classes for each goal point, reducing the search space needed by the MIP model to compute a near-optimal solution. We compare the optimal MIP solution with the anytime solution generated after 2 minutes

with the MIP solver and our heuristic solution, as well as a simulated annealing and a greedy approach. To avoid the significant computational expense required to compute the optimal solution to the MIP Model, the 2-minute anytime solution and the heuristic solution were found to be close approximations of the shortest path. In field trials, we were able to plan short, non-entangling paths both in a pool environment and during a wharf inspection.

Formulating the non-entangling constraint as a topological constraint allows us to incorporate it into the larger Travelling Salesperson constrained planning problem. This example demonstrates a pipeline that can be used to incorporate general topological constraints into other planning tasks. However, constraints are not the only way that topological representations might be used to affect robotic behavior. In the next chapter we will discuss ways to leverage topological features as background knowledge to improve robot performance in an information gathering task.

Chapter 4: Information Gathering with Topological Objectives

In this chapter we present a framework that enables a team of heterogeneous robots to benefit from expert domain knowledge about the characteristics of a topological feature to improve their performance in mapping and tracking the feature. In the previous chapter, we were able to explicitly leverage topological information in the form of constraints when the robot had perfect knowledge of the topological features in the environment. In this chapter, we relax this assumption, and instead consider a scenario where human experts can provide instruction on the structure, motion, and sampling behavior for a feature. Specifically, we consider the task of mapping and tracking a salinity front: an oceanographic feature that separates a body of relatively salty water from a body of relatively fresh water. These fronts can occur as a result of coastal upwelling [34], or in environments where a river plume mixes with seawater.

Our overall mission objective is to coordinate a team of heterogeneous robots, comprised of up to four Slocum Gliders [66] and up to two Robotic Ocean Surface Sampler (ROSS) vehicles [67], operating in an unknown and time-varying environment, to find and map a salinity front as it evolves over time. In addition to building a map of the front, the robots also need to collect time-series of oceanographic data by repeatedly crossing the front for use in post-hoc analyses. The first of these objectives, building an accurate map, follows the standard form of an Informative Path Planning Problem instance, and there is a body of field robotics literature that addresses the problem of

autonomous mapping of unknown environments [42] [68] [69]. However, the second objective is more difficult to optimize for, since it is defined relative to a topological feature. While there have been systems designed for similar missions, they have not considered it in a general optimization framework. Instead, these approaches use either rules-based algorithms [38], or adaptations of hand-designed sampling patterns [70]. To combine the two objectives, we need a way to consider the topological mission objective alongside the more typical mission objective. We will accomplish this by leveraging domain knowledge to encode knowledge about the topological feature into metric models and objectives.

In Chapter 4.1, we discuss the three ways that we incorporate the expert domain knowledge provided to the system into a centralized, heterogeneous robot planning framework. First, in Chapter 4.1.1, we present our method for modelling the structure of a salinity front with a Nearest Neighbors augmented Gaussian Process (NN-GP). The NN-GP augments a standard Gaussian Process (GP) with a nearest neighbors prior to allow the NN-GP to extrapolate the existence of the front into unobserved areas. Then, in Chapter 4.1.2, we discuss how we account for the motion of the salinity front by planning in a drifting Lagrangian reference frame, which increases the longevity of observations by reducing the apparent motion of the front over time. Finally, in Chapter 4.1.3, we describe our approach for approximating the topological mission objective with a metric objective function that encodes the human experts' experience of how to best sample the front. In Chapter 4.1.4, we tie these three components together with our Sequential Allocation Monte Carlo Tree Search information gathering algorithm. By incorporating domain knowledge of the topological feature into the planning process,

we were able to show a significant improvement in the performance of the robot team in the sampling task in simulated trials as described in Chapter 4.2, as well as proved its capabilities in a two-week long field trial in the Gulf of Mexico, described in Chapter 4.3. This work has been previously published in our journal paper [71].

4.1 Method

Ocean salinity fronts are large, on the order of tens of kilometers, and evolve slowly, over the course of days. To fully map them requires a team of robots working together. However, due to the dynamic nature of the ocean, it is not feasible to make a single plan and execute it for the entire sampling mission. Instead, we can adopt a receding horizon framework for our planner. Under this framework, the continuously time-varying world is broken up temporally into a set of discrete planning intervals. Since these intervals are short relative to the timescales over which the environment changes, the world can be assumed to be static for the purposes of planning. In between planning intervals, as the robots move through the world executing their plans, they collect sensor observations from the world. These observations are used to update the model of the environment used to inform the planner and any changes observed in the world will be incorporated into the new model. Using the updated model, a new plan is generated for the next planning interval, which will be followed in turn by another execution and update step. This process of plan-act-replan-act-replan is repeated until the mission is complete. Receding horizon-based planners have been shown to perform well in dynamic [72] and partially observable [73] environments Both the environmental modelling and path plan-

ning components of the receding horizon framework are places where we can put our domain knowledge to use to improve the performance of the planner over the course of the sampling task.

4.1.1 Nearest Neighbors Augmented Gaussian Process

In order for the planning algorithm to function, it requires an estimate of the state of the environment. A commonly used tool in field robotics applications to estimate the state of a scalar variable across an environment is a GP. GPs are widely used since they can construct a continuous estimate of a variable from a set of noisy observations and they can also provide an estimate of the uncertainty of its estimate. For our task, we use a three-dimensional GP to learn a mapping from the three input dimensions (a spatial position in $\mathbb{R}^2$ and time) to the output dimension.

Formally, a GP is a collection of random variables, any finite number of which have joint Gaussian distributions [74]. A GP can be fully specified by a mean function and a kernel function. A common practice is to use a 0-mean GP, which means that the mean function is biased out of observations and re-added to the GP prediction, allowing the mean within the GP to be 0. The GP kernel is a function that defines the covariance of any two points in the environment. A commonly-used kernel, and the one used in our work, is the radial basis function or squared exponential kernel. The squared exponential kernel function for the covariance between two points x and $x' \in \mathbb{R}^n$ is given by:

$$\mathcal{K}_{\text{SE}}(x, x') = \sigma^2 \exp\left(\frac{-(x_1 - x_1')^2}{2l_1^2} + \frac{-(x_2 - x_2')^2}{2l_2^2} + ... + \frac{-(x_n - x_n')^2}{2l_n^2}\right), \quad (4.1)$$

where σ^2 is a the output variance of the function being predicted, and l_d is the length-scale, which defines the scale over points co-vary in dimension d. While using a squared exponential kernel, GPs struggle to extrapolate data beyond the convex hull of the samples wile maintaining the ability to resolve sharp features. This issue can be partially addressed through the selection of a different kernel for the GP, such as periodic or non-stationary kernels. However, these require significantly more computation to compute, limiting their applicability in real-time mapping tasks.

Instead, we can exploit our domain knowledge of ocean fronts to build a more accurate squared exponential based GP to predict their location in the world. Since we know that ocean salinity fronts are characterized by a localized sharp gradient of salinity, we can augment our GP with an additional model that can both resolve the sharp gradient while maintaining the ability to extrapolate beyond the observations. We chose a nearest neighbors model for its simplicity and ease of computation since it will be used as a component of a real-time system.

The first step in our NN-GP is to construct the nearest neighbors model of the environment. This step constructs a Voronoi-type diagram, where the salinity value for each point in the continuous domains is solely determined by the salinity at the nearest observation, as can be seen in Fig. 4.1a. While the nearest neighbors reconstruction could be used directly as the mean function for a GP, it can have sharp gradients, and where there are many observations close together, it can become noisy. To address both of these issues, in the second step of the NN-GP, we train a GP using a set of pseudo-observations regularly sampled from the nearest neighbors estimation. The result of this process is a smoothed version of the nearest neighbors estimation that can be used as a mean func-

tion for a second GP, which uses the full observation set. By subtracting the estimated mean value from the true observation, the second GP becomes a zero-mean GP that learns the difference between the pseudo-observed GP and the actual observations. This process is shown in Fig. 4.1.

4.1.2 Planning in Lagrangian Reference Frame

While the NN-GP allows us to construct a more accurate model of a salinity front from a limited set of observations, as time moves advances, the NN-GP model becomes inaccurate as the front moves over time. While periodically re-building the model with fresh observations addresses this problem, we can leverage knowledge about how the fluid dynamics of the ocean control the motion of the front to increase the duration each observation remains relevant.

Lagrangian fluid dynamics models the motion of a fluid by tracking the motion of a small packets of that fluid over time. This model has been leveraged in the robotics community by using a Lagrangian surface drifter to 'tag' a fluid packet and then operating an autonomous underwater vehicle in a pre-defined pattern around the drifter to perform continuous monitoring of that water packet [75]. We expand this method in two ways. First we augment the surface drift motion obtained from the drifter with measured ocean currents using observations made by the R/V Pelican's Acoustic Doppler Current Profiler (ADCP), averaged over the top 30 m of water. Ideally a surface drifter will perfectly follow the water it is placed in; however, other forces, such as winds can cause its motion to differ from the water's motion (i.e., become quasi-Lagrangian). We observed

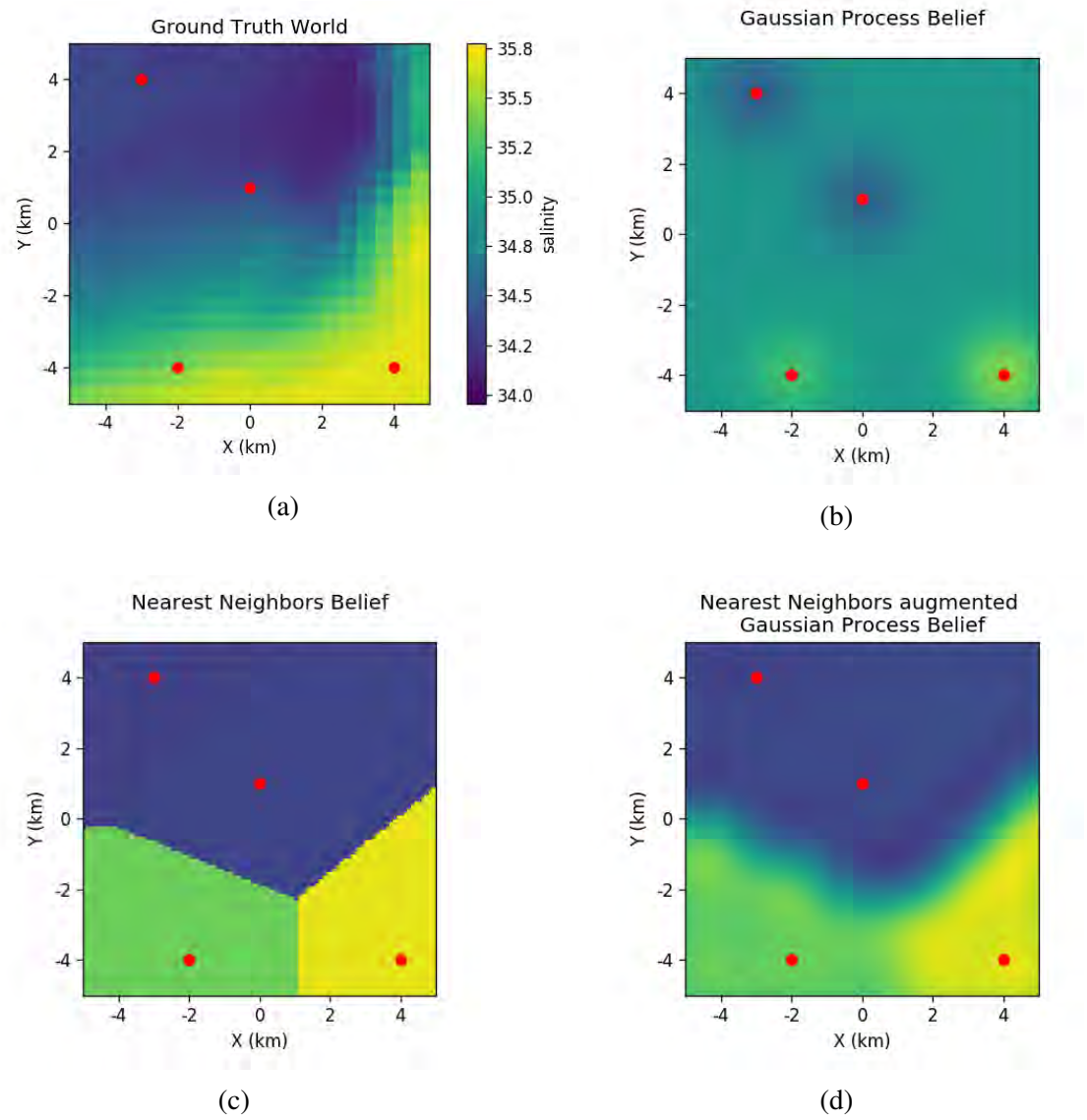
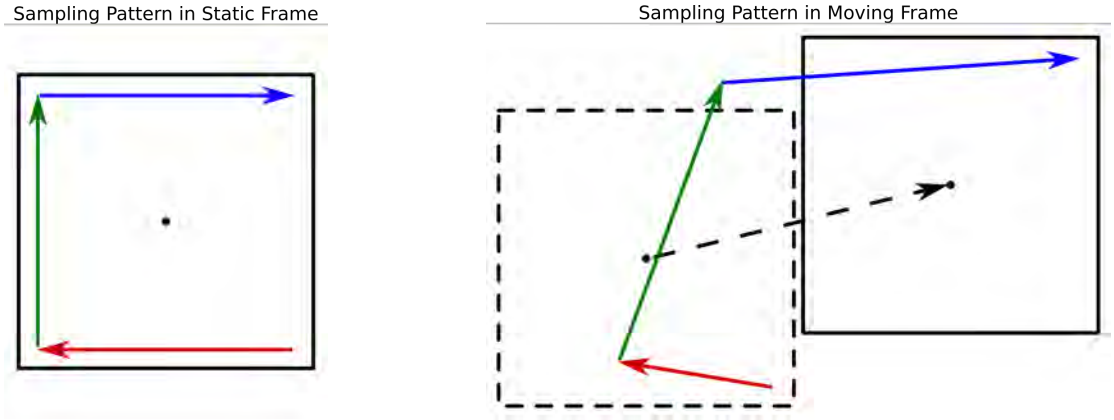


Figure 4.1: Example of the Nearest-Neighbors augmented GP from simulation. Sampling locations are shown in red. Due to the sparsity of the sampling, the standard GP (b) fails to produce a useful belief, regressing back to the mean value too quickly. Leveraging the nearest-neighbors Voronoi estimation of the environment (c) allows the nearest neighbors augmented GP (d) to extrapolate the samples, producing a more accurate estimation of the front's location.

this behavior during our deployment in the Gulf of Mexico. In relatively calm seas, the surface drifter tracked surface currents well, exhibiting a characteristic looping behavior strongly indicative of near-inertial oscillations driven by Coriolis forces. However, as the deployment progressed and stormier weather arrived, the motion of the drifter began to be dominated by the winds. This change was made evident by the motion of the drifter diverging from the Lagrangian motion that the ship's ADCP measurements would predict. To account for this discrepancy we employed a Kalman Filter [76] to merge the ADCP current velocity measurements with the GPS updates of the drifter's position to construct a better estimate of the true position of the water packet we were tracking.

The second novelty that we introduce to the Lagrangian planning frame, is that we do not use a pre-defined pattern around the drifter, but rather perform the entire receding horizon planning loop within the Lagrangian frame. This adaptation requires that the waypoints generated by the planning algorithm are translated to global coordinates to send to the individual vehicles. When a vehicle makes an observation $u_{global} = (x_{longitude}, y_{latitude}, t, s)$, it is transformed to the planning frame using an affine transform, defined by the position of the virtual drifter at time t. This transform is applied to the positions of all obstacles (both static and dynamic) to move them into the planning frame. The planner is allowed to run, producing paths that are then transformed back into the global frame in order to be passed to the vehicles as waypoints defined by latitude and longitude. As a part of this process, the paths are forward propagated in time assuming the transform maintains a constant velocity. In this way, the vehicles execute their trajectories as planned in the global and local frames.



(a) (b)

Figure 4.2: Effects of the moving planning frame on a sampling trajectory. The path shown in (a) is planned in a static frame. Using the estimated motion of the frame, given by the drifter and ADCP data, the path is forward-projected in time (b).

4.1.3 Objective Function

The topological research objective for our multi-robot team is to resolve a salinity front and in doing so, conduct repeated perpendicular transects of the front to collect a scientific dataset for analysis by ocean scientists. However, this objective function is difficult to optimize directly. Counting number of front crossings is an information-sparse objective function, as it lacks a significant gradient in the reward space that can allow an informative path planning algorithm to converge to a good solution. Resolving a front (i.e., building an accurate map of it) is another problem that cannot be optimized directly without access to the ground truth map. Methods, such as those that minimize variance in the GP, minimize overall entropy, or use an upper confidence bound to attempt to address this issue. However, these methods require significant computational resources to compute, which would have a negative impact on the ability of any algorithm to explore

the space of possible solutions, a task that is even more difficult in the multi-robot case.

To address these challenges, using significant domain expertise contributed by our ocean scientist team members, we developed an objective function to indirectly optimize our stated mission. Our objective function for a path P, considering the set of other robots' paths $\mathcal{P}$ is given by

$$I(P|\mathcal{P}) = \sum_{E \in P} \left[\left(\int_E \alpha_1 \left| \frac{\delta S}{\delta x} \right| + \alpha_2 \Omega(\mathcal{P}) + \alpha_3 \Omega(\mathcal{P}) \left| \frac{\delta S}{\delta x} \right| \, dx \right) \times T(E) \times C(E) \right], \tag{4.2}$$

where the score for a path is determined by the sum along each edge, E of that path. Within the summation, the objective function consists of five terms.

- **Gradient Score:** The first term consists of the magnitude of the salinity gradient along the edge, $|\delta S/\delta x|$. Paths that maximize this score are ones that repeatedly cross the front, exhibiting exploitative sampling behavior. This score is weighted by a scalar term, α_1, allowing the system operator to specify its relative importance compared to the novelty and novelty-weighted gradient scores.

- **Novelty Score:** The second term, $\Omega(\mathcal{P})$, weights the value of traversing an edge on the graph based on that edge's novelty (normalized NN-GP variance), given a set of paths, $\mathcal{P}$. The novelty score is computed by taking the line-integral of the novelty field along the edge. Since the novelty field is directly derived from the variance of the GP world model, paths that maximize this score explore regions in the environment with high uncertainty, either because they have not been sampled or because there are multiple conflicting samples, while avoiding regions that are

along the other robots' paths in $\mathcal{P}$. Similar to the gradient score, the novelty score carries a scalar weighting term, α_2.

- **Novelty-Weighted Gradient Score:** The third term in the objective function is the product of the gradient and salinity terms. Similar to how an upper confidence bound incentivizes exploration of states with high potential rewards in Bayesian optimization [77], the novelty-weighted gradient term encourages vehicles to explore areas believed to be on the front (i.e., have a high gradient score) that are insufficiently explored (i.e., have a high novelty score). While a weighted sum of the gradient and novelty scores can achieve the same effect, by using a product instead, we encode the idea that it is not enough to simply have high value purely from exploration or exploitation, but rather *both* are needed in order to make useful observations for ocean scientists. Like the first two terms, this term is weighted by α_3.

- **Temporal Discount Factor:** Since our planning algorithm treats the world as static for the purposes of planning, despite the ocean being a dynamic environment, as time progresses discrepancies will accumulate between the model of the world used for planning and the ground truth environment. To account for these uncertainties, we add a temporal discount $T(E)$ factor to the edges score, discounting edges' scores as a function of how far forward into the future they are. In this way, we allow our planner to be non-myopic and consider paths longer than a single planning cycle, but without over-relying on temporally distant reward.

- **Continuity Discount Factor:** Lastly, we heavily discount edges near where a

vehicle performs a turn. There are two main reasons for this discount. First, many active acoustic sensors, such as ADCPs and sonar, experience blurring when moved and rotated, greatly degrading the quality of their data. Secondly, for frequency domain analyses, the maximum resolvable frequency of a dataset is proportional to the length of that dataset. In order to ensure that the data collected by the autonomous vehicles is high-quality and of use to ocean scientists, we penalized paths that have turns to encourage the autonomy system to identify plans that consist of long, straight segments. The continuity score of an edge, $C(E)$, is $.01$ if the edge is not co-linear with the previous or next segment of the path, and it is 1 otherwise.

4.1.4 Sequential Allocation Monte Carlo Tree Search

Optimization methods provide a way to combine the domain knowledge encoded in our NN-GP constructed using Lagrangian observations with the topological mission objective represented in Equation 4.2. However, optimizing this objective function in the joint state and action space of all the members of the robot team results in a very high dimensional search problem, making it computationally infeasible to compute a good solution in a reasonable amount of time. We address this by turning to a sequential allocation framework that enables us to plan for only a single robot at a time, holding the remainder of the robots' plans static [49]. This process is repeated for each robot, until a fixed number of iterations through all the planning robots has been completed or the plans converge to a locally optimal set. In practice we found that completing 3 cycles

through each robot provided a good balance between computation time and the quality of the resulting team plan. Our sequential allocation algorithm is shown in Algorithm 1.

Algorithm 1 Sequential Allocation Monte Carlo Tree Search

1: **function** SA-MCTS($\mathcal{O}$, $\mathcal{R}$, $\tilde{\mathcal{R}}$)
2: $bel \longleftarrow$ GPNN($\mathcal{O}$)
3: $\mathcal{G} \longleftarrow$ constructGraph(bel)
4: $\mathcal{P}_t \leftarrow \begin{cases} P_{r_i} \leftarrow \emptyset \textbf{ for } r_i \in \mathcal{R} \\ P_{r_i} \leftarrow P_{r_i} \textbf{ for } r_i \in \tilde{\mathcal{R}} \end{cases}$
5: **while** $\neg$ stopping **do**
6: **for** $r_i \in \mathcal{R}$ **do**
7: $bel_{r_i} \longleftarrow$ updateNovelty($bel, \mathcal{P} - P_{r_i}$)
8: $\mathcal{G}_{r_i} \longleftarrow$ updateGraph($\mathcal{G}, bel_{r_i}, r_i$)
9: $P_{r_i} \longleftarrow$ MCTS($\mathcal{G}_{r_i}, r_i, \mathcal{P} - P_{r_i}$)
10: **return** $\mathcal{P}_t$

We begin the planning process at time t by identifying the set of N robots for which plans need to be generated, $\mathcal{R}$. A robot can require a new plan, either because it has completed its current plan or because it has been longer than its replanning interval since it last received a new plan. The gliders' replanning rates are defined by their surfacing interval (approximately two hours), while the ROSS vehicles' replanning rate was chosen to be half that of the gliders', due to their higher speed and constant access to satellite communication. Those M robots that were not selected as planning robots are considered non-planning robots, $\tilde{\mathcal{R}}$, with $\emptyset = \mathcal{R} \bigcap \tilde{\mathcal{R}}$. Though the non-planning robots will not receive new plans, they still influence the planning process through their previously generated plans, $\mathcal{P}_{t-1}$.

The next step is to generate a belief (consisting of a salinity estimate and a novelty estimate) using the nearest neighbors augmented GP described in Section 4.1.1. Using

this belief, we construct a uniform, 8-connected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ over the planning domain. Then, we compute the salinity gradient for each edge $e_{i,j} \in \mathcal{E}$ for use in our objective function, which is described in more detail in Section 4.1.3. To speed up computation, these values can be pre-calculated and stored in a lookup table. Since the scale of the paths we are interested in planning in are on the order of kilometers, the vehicle dynamics during turning are negligible compared to the effects of the ocean currents on the ability of a vehicle to traverse an edge. We incorporate this constraint into the cost to traverse each edge on the graph by defining an edge's cost as the time it takes for a vehicle to move along that edge. For a single robot, or even a homogeneous multi-robot team, this cost would simply be interchangeable with the distance along the edge and could also be pre-computed. However, for our heterogeneous robot team, we need to account for the different speeds of each vehicle, and so we define a cost function for each edge that is a function of the velocity of the robot, the velocity of the ocean currents, as well as the length of the edge. To simplify this calculation, we assume that the ocean currents are uniform along the entire edge with heading, and that a robot's ability to traverse the edge is only affected by the component of the ocean velocities parallel with the edge.

While there are no islands in our deployment region, oil rigs and other vessels create a plethora of static and dynamic obstacles, which our vehicles need to avoid. Obstacle avoidance is implemented at the graph construction step by defining exclusion zones around each obstacle. For the all obstacles, which were primarily oil rigs and ship-wrecks, we simply construct a circular zone with a 1 km radius around each obstacle's location. Using a constant velocity motion model, we project dynamic obstacles' mo-

tion forward in time for an entire planning cycle. The exclusion zone for each dynamic obstacle is the entire area swept out by the circular zone along the forward projection. Once all the exclusion zones are computed, we remove all edges that intersect one or more zones. While one of our vehicles could pass behind the vehicle once it has passed, we chose a more conservative approach to better ensure the safety of our assets as well as that of the other vehicles in the area. When operating with our drifting reference frame, we simply add the motion of the frame (again with a constant velocity assumption) and forward simulate their positions in time.

During each allocation cycle a planning robot r_i is selected from planning robots $\mathcal{R}$. We perform two preprocessing steps (Lines 6 and 7) before using Monte Carlo Tree Search (MCTS) to find a path for the robot. The first of these steps is to update the novelty estimate, based upon the paths for all the vehicles (both in $\mathcal{R}$ and $\tilde{\mathcal{R}}$) that are *not* the planning robot. What this process amounts to is forward simulating each of these robots' plans and modifying the novelty field to reduce novelty along the plan; thus disincentivizing r_i from planning a path that goes through areas that will be explored by other robots in the future.

The second preprocessing step is to calculate the novelty-based edge rewards using this updated novelty estimate and to determine the edge costs based on the planning robot's velocity. These values, like the salinity gradient rewards, are stored in a lookup table.

To perform the individual path planning for each robot, we turn to MCTS as a flexible, yet powerful optimization technique. We outline our implementation of MCTS in Algorithm 2. MCTS combines ideas from graph search and Bayesian Optimization in

order to enable it to perform well in problems where the search space is too large for exhaustive techniques like breadth-first search or branch and bound, to succeed. Like many search-based planners, MCTS expands a tree of partial solutions until some termination condition is met. However, where MCTS differs is that it biases its search toward areas of the search domain where it has encountered good solutions. This biasing is done through selecting which leaf node of the MCTS tree to expand (Line 3 in Algorithm 2) using a metric, such the selection process used in the upper confidence bound for trees algorithm [78]. Once a node is selected for expansion v_{curr}, each of its neighboring states is added to the tree as children of v_{curr}.

Algorithm 2 Monte Carlo Tree Search (MCTS)

1: **function** MCTS($\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, r, P_{other})
2: $tree \longleftarrow$ InitializeTree$(r, \mathcal{V})$
3: **while** $\neg$ stopping **do**
4: $v_{curr} \longleftarrow$ chooseExpansionNode$(tree)$
5: $tree \longleftarrow$ expandTree$(v_{curr}, \mathcal{E})$
6: $\lambda_{curr} \longleftarrow I(P_{v_{curr}}|P_{other})$
7: $tree \longleftarrow$ backpropagate$(tree, \lambda_{curr}, v_{curr})$
8: $P_r \longleftarrow$ getBestPath$(tree)$
9: **return** P_r

Since the MCTS search tree is built up incrementally, it is likely that any given leaf node contains only a partial solution to the search problem (i.e., a partial path that does not fully exhaust a vehicle's available budget). In order to evaluate and compare these partial solutions, MCTS uses a default policy to complete each partial solution. We explored three different default policies.

- **Random:** While executing the random policy, the vehicle chooses a random edge

at each visited vertex. The only constraint is that the vehicle cannot choose to return along the edge on which it arrived at the vertex in order to prevent backtracking and oscillating behavior.

- **Greedy:** The greedy policy guides the vehicle to myopically select the edge at a vertex with the highest score based on the information gain function. Similarly to the random policy, the vehicle is prevented from backtracking along the edge from which it arrived.

- **Straight Line:** The final policy, which was found to perform the best, maintains the vehicle's heading and continues travelling in a straight line. If the vehicle could not continue, for example if the vehicle would leave the domain or collide with an obstacle, a random new heading is selected.

Once a default policy rollout is completed, the full path is evaluated and a score is calculated according to our objective function. The score is backpropagated up the tree and used to update the probabilities that nodes are selected for expansion. The final step in the MCTS algorithm is to obtain the final path, which we accomplished by repeatedly selecting the action with the highest score at each node within the tree until a leaf node is reached.

4.2 Simulated Results

To quantify the performance of SA-MCTS, we performed a series of experiments in simulation to evaluate the performance of several different planning algorithms in the

front-tracking task, as well as to quantify the effects of modelling environments using our nearest neighbors augmented GP and performing the front-tracking and sampling task using a static frame and a dynamic frame.

4.2.1 Simulation Setup

For these simulated experiments we used a Regional Ocean Modelling System [79] model of the Gulf Coast[1]. Within this model, we manually identified five 30 km by 30 km regions with strong salinity fronts that persisted over 48 hours for use as simulated environments. Within these larger areas, we used a 10 km by 10 km region for the planning area. The purpose of the larger regions is to provide room for the moving planning frame to stay entirely within the simulated region over the 48 hour simulation. The initial pose of the planning area was hand-selected to ensure that it was nearby the front and that the front would remain largely within the planning area. Each simulation contained a five-vehicle team: three Slocum Gliders and two ROSS vehicles. We conducted five simulated trials in each of the five worlds with randomized vehicle starting locations. The locations of the simulations are shown in Fig. 4.3.

4.2.2 Environmental Estimation

The first set of simulated trials was to evaluate how our environmental estimation techniques affected the overall performance of the robot team in their sampling task.

[1]The Regional Ocean Modelling System environment used for these simulations was provided by the Texas A&M Physical Oceanography Numerical Group. http://barataria.tamu.edu

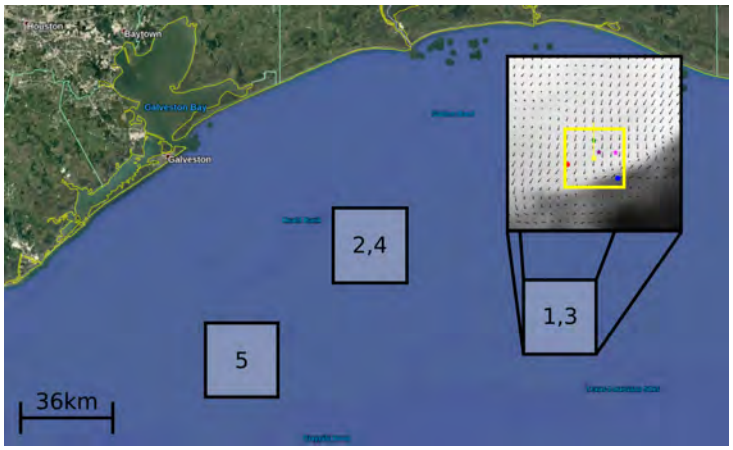
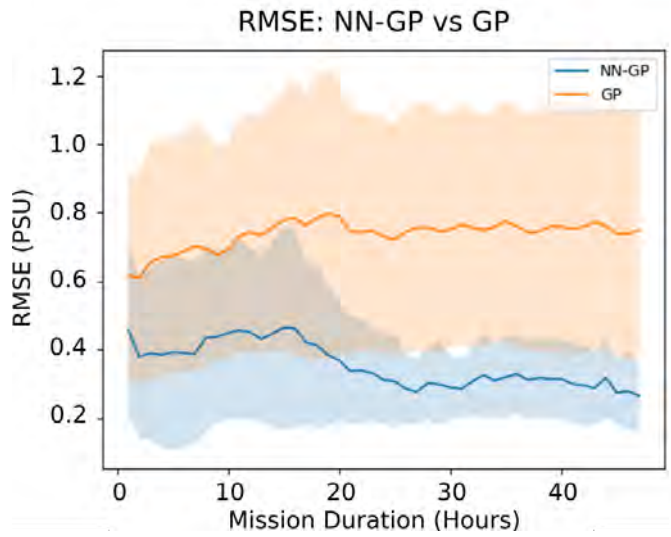


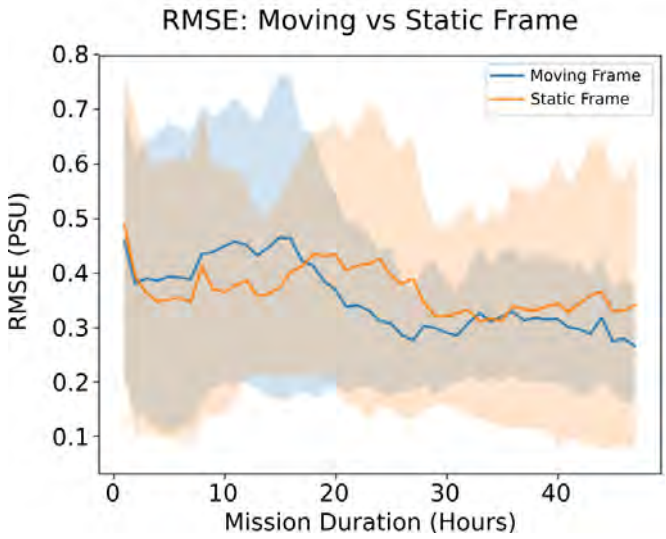
Figure 4.3: The locations of the five simulated environments in the Gulf of Mexico. Worlds 1 and 2 use ROMS model output covering Jun 1-3 2011, while worlds 3, 4, and 5 are use output from Jun 4-7, 2011. The inset shows a snapshot from our simulator, showing the salinity (background grayscale) across the planning region (yellow rectangle) along with the two simulated ROSS vehicles (colored dots) and three gliders (colored stars). Over the course of the simulation, the planning region will drift with the time-varying currents and move along with the front.

The first experiment, shown in Fig. 4.4a, compares the system's performance using our Nearest Neighbors augmented Gaussian Process (NNGP) with a standard GP. The hyperparameters for both the GP and the NNGP are identical, using a lengthscale of 1 km in space and 3 hrs in time. The results clearly show the advantage of using our NNGP over the standard GP, with a reduction in the average estimation error across the planning environment of .383 Practical Salinity Units (PSU), which corresponds to a 52% reduction. This can be attributed to the fact that the NNGP leverages additional domain knowledge of the structure of these fronts. Since we know that the fronts are much larger in scale than the 10 km by 10 km planning region, we can safely extrapolate its existence to the planning region's edges. The NNGP is capable of performing this extrapolation, while the GP cannot.

The second experiment evaluated the benefits of the drifting reference frame, comparing it against simply holding the frame fixed. As seen in Fig 4.4b, on average, the two methods performed similarly, with the moving frame only exhibiting an improvement in estimation accuracy of .012 PSU (.3%). However, where the benefit of the moving becomes apparent is in the reduction in the standard deviation of this estimation error, where it provides a 24% reduction. This result suggests that while the estimates of the salinity in the moving frame are, on average, no more accurate than under the static frame, many of the outliers have disappeared, making the estimation under the moving frame more *reliably* accurate.



(a)



(b)

Figure 4.4: Comparisons of our SA-MCTS algorithm in a simulated Regional Ocean Modelling System environment. The plot in (a) shows a comparison of our nearest neighbors augmented Gaussian Process and standard Gaussian Process, while (b) shows a comparison of the drifting reference frame with the static reference frame. The shaded region represents one standard deviation from the mean.

4.2.3 Algorithmic Performance

We also evaluated the performance of our Sequential Allocation Monte Carlo Tree Search (SA-MCTS) algorithm as a heterogeneous multi-robot informative path planner. We compare SA-MCTS against a systematic planner that was modelled on the back-and-forth coverage plans used by the oceanographers in Chapter 4.3, as well as an Evolutionary Algorithm (EA) planner that optimizes the paths of the robot teams jointly, rather than sequentially. In the front-mapping task, the entire environment is not equal as far as the utility of accurate mapping is concerned. Accuracy near the front is much more valuable than accuracy far away from it. To account for this difference, when we compare the performance of the three planners, we divide up the planning environment into high, medium and low value regions corresponding to regions containing 100-66%, 66-33%, and 33-0%, respectively, of the largest magnitude of the gradient of salinity in the planning area. The area covered in these regions is not uniform. While the exact distribution of these areas varies from world to world, on average the high-value area comprises 2.8% of the environment, the middle-value region covers 14.0%, and the low-value region is the remaining 83% of the environment, as seen in Fig. 4.5. We evaluate each algorithm using 25 trials divided among 5 worlds.

The first metric that we evaluated the three algorithms on is the percentage of total experiment time that is spent in each of the three regions. The results from this experiment can be seen in Fig 4.6a. Here we see that while all three methods perform similarly in the low and medium value regions, both the EA and our SA-MCTS method spend significantly more time in the high-value region than the systematic method. This

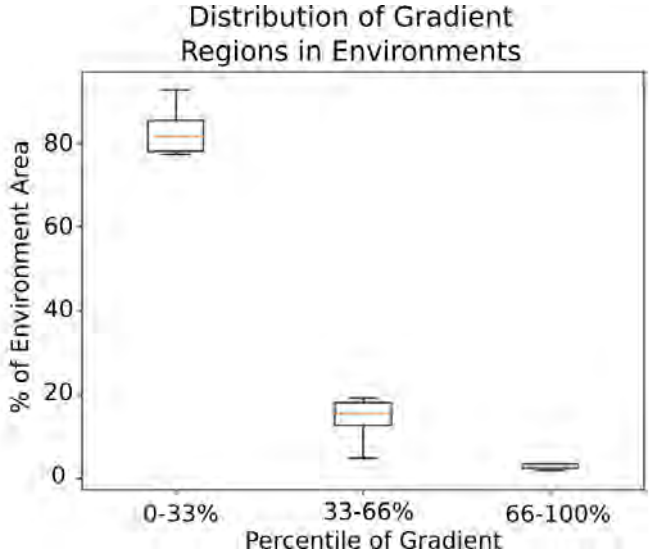
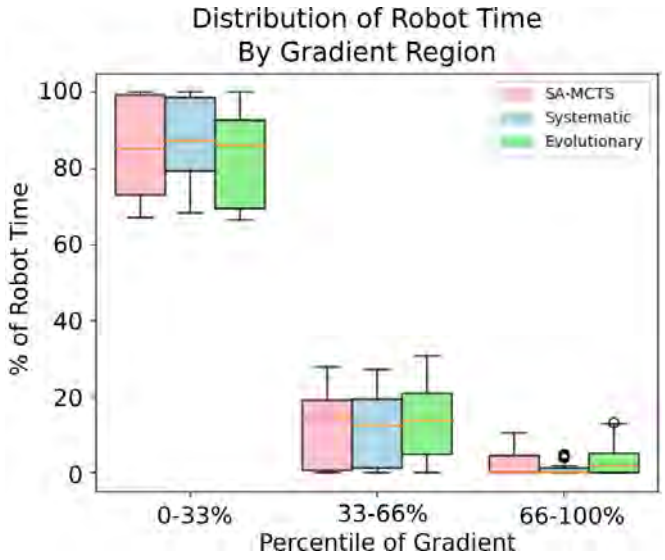


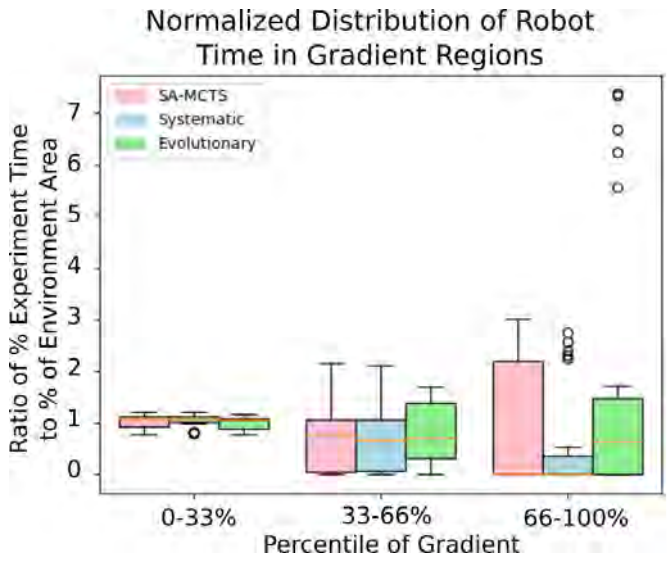
Figure 4.5: The total area covered by low, medium, and high-gradient regions in the simulated worlds.

result is to be expected, since both the EA and SA-MCTS are considering the gradient as a component of their objective function, while the systematic coverage does not. To better compare the amount of time each planner has the vehicles spending in each region, in Fig 4.6b we normalize the amount of experiment time spent in each region by the percentage of each environment covered by each region. This metric results in a ratio, where a value of 1 means that a vehicle is evenly dividing its time among all regions, and values higher and lower than 1 mean that the vehicles are spending proportionally more or less time in a given region.

We also compare the algorithms on their accuracy in estimating the salinity value in each region. While again, the performance is similar across all three algorithms, as shown in Fig. 4.7a, our SA-MCTS performs slightly better on average in the medium and high-value regions. However, our method performs significantly better than the



(a)



(b)

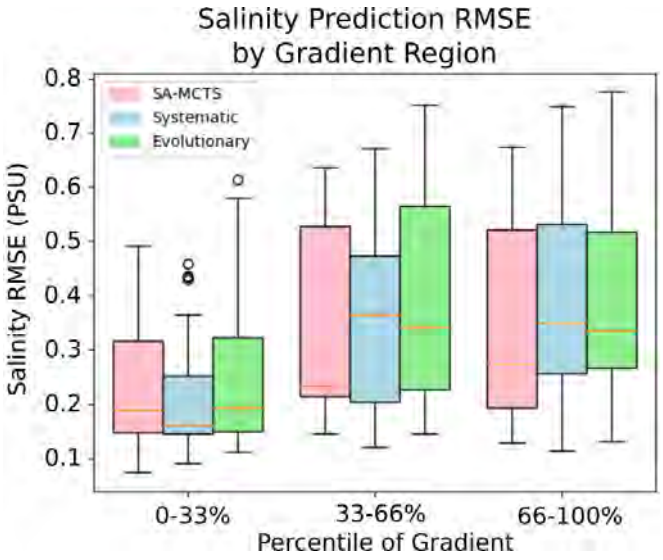
Figure 4.6: Comparison of the amount of time spent in high, middle, and low gradient regions in simulated trials. The total amount of time spent in each region (a) is normalized by the percent of the environment covered to show the effectiveness of each planner at guiding the robot to each region (b).

systematic planner and the EA in reducing uncertainty in these regions as can be seen in Fig 4.7a. What this result means is that the EA and SA-MCTS spend similar amounts of time in the high-value regions, the samples collected under SA-MCTS are more useful in resolving the front, as they enable both a more accurate mapping, as well as a greater reduction in the autonomy system's estimated uncertainty along the front.

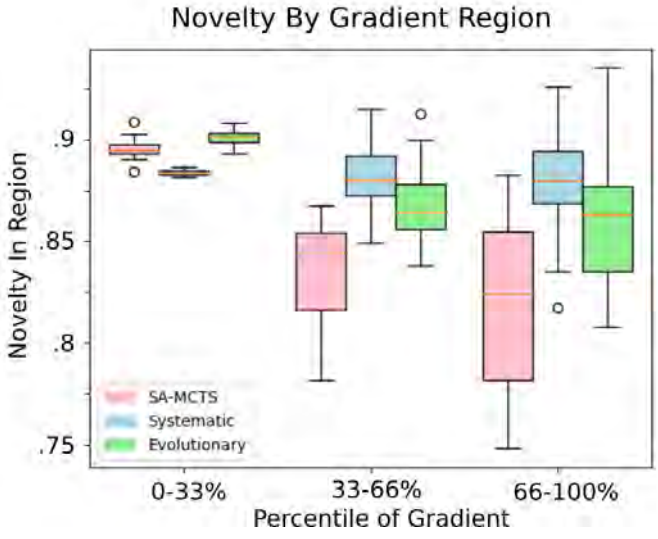
4.2.4 Parameter Robustness

Our last set of simulated trials focused on evaluating the robustness of our SA-MCTS algorithm to the parameter weighting in its objective function. The information-gathering task in an unknown environment is an inherently multi-objective optimization problem. A robot must balance the competing objectives of exploring the operational area to find high-value regions and exploiting previously-identified high-value regions. With the introduction of our gradient-weighted novelty term to the objective function, we attempt to address this tradeoff by combining the exploration and exploitation objectives into a single term. The results from a parameter sweep over our three-dimensional objective function parameter space are shown in Fig. 4.8. As before, for each individual parameter setting, we performed 25 simulated trials divided among 5 worlds.

These results clearly show the exploration-exploitation tradeoff. When the gradient-weighted novelty weight is set to zero, the robot only considers the novelty (exploration) and gradient (exploitation) terms. There is a clear sweet spot along this axis where the novelty term carries a weight of about .3, and the gradient term has a weight of about .7. However, through the introduction of the gradient-weighted novelty term, the system

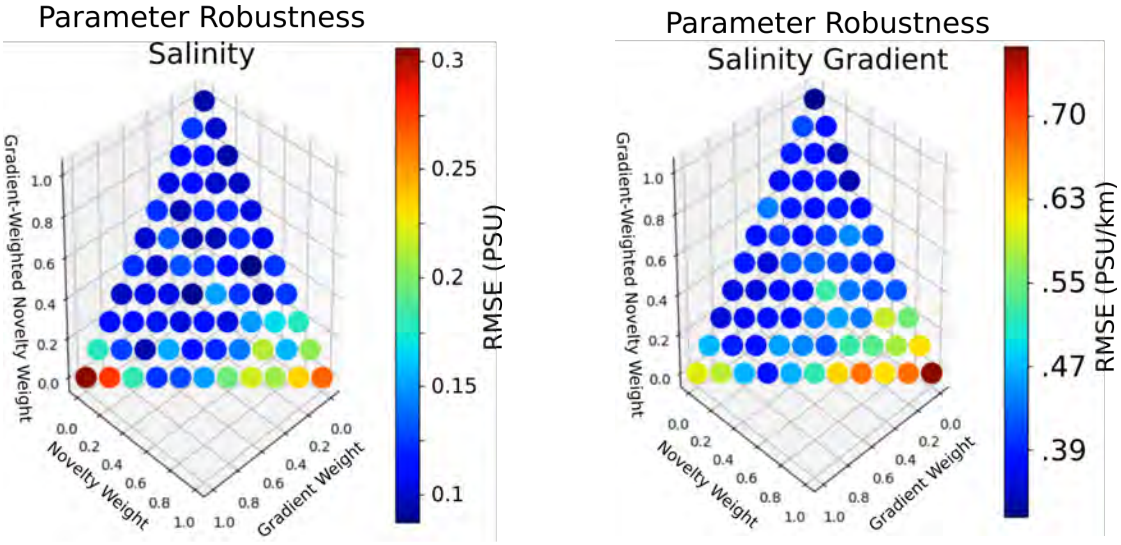


(a)



(b)

Figure 4.7: Comparison of the amount of RMSE (a) and estimated uncertainty (b) between the predicted field and the simulated ground truth in the low, medium, and high-value regions.



(a) (b)

Figure 4.8: A comparison of the performance of the MCTS algorithm for different weightings of the first three terms of the objective function outlined in Chapter 4.1.3. The sum of the weights must be 1, resulting in the 2D plane shown in the 3D parameter space. The effects of the parameters are quantified in terms of the RMSE of the salinity estimate (a) and the RMSE of the estimate of the salinity gradient (b) at the end of the 48h simulation.

achieves good performance across a wide range of parameter values, including when both the novelty and gradient term weights are set to zero. This result suggests that the gradient-weighted novelty term succeeds in its intended purpose of balancing the competing objectives of exploration and exploitation. Furthermore, the robustness of the SA-MCTS algorithm is shown, since its performance is relatively unaffected by all but the most extreme parameter settings.

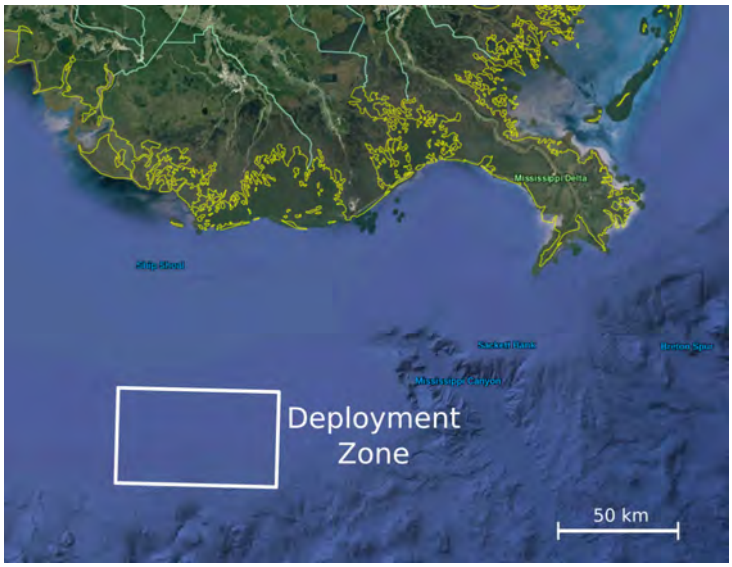


Figure 4.9: The various experiments conducted during the deployment took place in a region approximately 65 km x 38 km located off of the Louisiana coast near the Mississippi River delta.

4.3 Gulf of Mexico Deployment

We tested our system in a two-week long deployment in a 65 km by 38 km region of the Gulf of Mexico situated approximately 80 km off of the Louisiana coast. The deployment zone is shown in Fig. 4.9. For the field experiments, we used up to four Slocum G3 gliders, up to two ROSS autonomous surface vehicles, a Lagrangian drifter buoy, and our research vessel, the R/V Pelican. Over the course of the field experiments, we were fortunate to have largely good conditions, with sea states between 0 and 1, though stormier weather later in the cruise brought conditions up to sea state 3-4. The two-week cruise was broken up into four different autonomy experiments, in addition to a single manually controlled experiment where the vehicles were piloted by the ocean scientists. The manual experiment was used as a baseline to characterize the

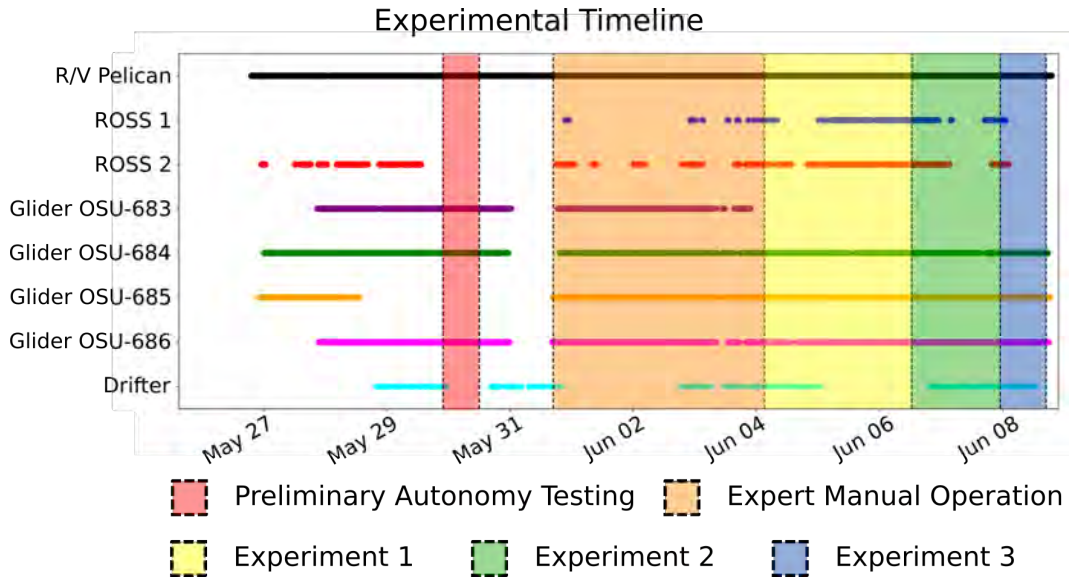


Figure 4.10: Over the course of the five different experiments, we deployed four different Slocum G3 gliders, two ROSS Vehicles, and a single Lagrangian drifter.

performance of the autonomy system. These experiments are outlined in Table 4.1 and visualized in Fig. 4.10. In between each of the autonomy experiments, we varied parameters, such as the relative weightings of terms in our objective function, the spatial and temporal length scales used in our nearest neighbors augmented GP estimator, and the replanning rates used by the different vehicles within the autonomy system. The portion of the cruise prior to the preliminary autonomy testing was a shakedown period, as various pieces of equipment were tested and minor issues in communications between the autonomous vehicles, the autonomy system, and the R/V Pelican's onboard systems were resolved. One notable challenge that we had to solve during this portion of the cruise was developing a way to prevent the ubiquitous seaweed on the ocean surface from fouling the ROSS's water intake and stalling the engine.

Table 4.1: Overview of experiments performed during the Gulf of Mexico deployment

	Duration	Asset-Hours	Assets
Preliminary Autonomy Testing	14h	57h	R/V Pelican, 3 Slocum Gliders
Expert Manual Control	82h	517h	R/V Pelican, 4 Slocum Gliders, 2 ROSS
Autonomous Experiment 1	57.5h	246h	3 Slocum Gliders, 2 ROSS
Autonomous Experiment 2	34.5h	133h	3 Slocum Gliders, 2 ROSS
Autonomous Experiment 3	18h	59h	3 Slocum Gliders, 2 ROSS

- **Preliminary Autonomy Testing:** The first autonomy system trial was a preliminary test, with the goal of debugging any issues that appeared during fully autonomous operations. Conducted using a static frame for simplicity, the test involved the three operational Slocum Gliders as well as the R/V Pelican. In this experiment, the Pelican was included as a 'robotic' asset in order to achieve a heterogeneous team, since the anti-fouling system for the ROSS was still under development. Waypoints for the Pelican were generated by the autonomy system and passed to the bridge by hand.

- **Expert Manual Operation:** During the manual operations experiment a team of expert oceanographers hand-designed the vehicles' survey patterns. This experiment used four Slocum gliders, the R/V Pelican, and up to two ROSS vehicles. Unfortunately near the end of the manual operations, glider OSU-683 suffered a mechanical failure and had to be recovered and could no longer participate in the

experimentation. A representative time-series of beliefs along with the respective novelty fields from this experiment are shown in Fig. 4.11.

- **Autonomy Experiment 1:** The first autonomy experiment was also the first test of the autonomy system using all intended components. Experiment 1 can be divided into three distinct periods based on what method we used for defining the planning frame. As seen in Fig. 4.10, the drifter was deployed during the first portion of the experiment, and during that time, the 10 km by 10 km planning frame's southeastern corner was located on the drifter's position. However, toward the end of period the drifter's motion was being overly influenced by the increasingly powerful wind instead of surface currents. To counteract the wind's effect on the drifter, we recovered the it and transitioned to a static frame, while we adapted the autonomy system to incorporate information from the Pelican's ADCP. The final portion of the experiment was conducted using a moving frame centered on a virtual drifter, the motion of which was informed by the Pelican's ADCP. During this test, we used up to two ROSS vehicles and three Slocum Gliders. An illustrative timeseries from this experiment is shown in Fig. 4.12.

- **Autonomy Experiment 2:** The second autonomy experiment was very similar to the end of the first, with two notable differences. Firstly we adjusted the weights on the objective function to omit the novelty-weighted gradient term, and secondly we began using a Kalman filter to merge the drifter position with the Pelican ADCP data in order to produce a more accurate measure of the Lagrangian drift of the ocean for the purposes of defining the motion of the planning frame.

- **Autonomy Experiment 3:** The final autonomy experiment was largely a return to the parameter set used in Experiment 1, but combined with the planning frame motion as used in Experiment 2. However, this test was significantly shorter than Experiment 1, and toward the end of the experiment, the frame was made static to prepare for the final recovery of the vehicles.

4.3.1 Evaluation of Autonomy Performance

In order to evaluate the effectiveness of the autonomy system relative to the manually controlled operations, we use two quantitative metrics. The first of these metrics is a count of the number of times a vehicle in the system crossed an ocean salinity front. To account for the fact that the various experiments were of different lengths and utilized different numbers of assets (see Fig. 4.10), we normalize this number based on the number of asset-hours of operation during each experiment. The number of front crossings is determined by counting the number of times a vehicle crosses a salinity threshold. For each experiment, the value of this threshold is determined by averaging the observed salinity values from each vehicle. This process resulted in the threshold values for each experiment falling within the range of 36 ± 0.2 PSU (Practical Salinity Units), which is in-line with our estimated value for the salinity at the front of 36 PSU. In order to ensure that the counts of front crossings were true crossings and not spurious ones caused by observation noise, we smoothed the data using a 1-hour moving average filter, and further de-noised the data by utilizing a rising/falling edge threshold with a window of 0.1 PSU. Under this scheme, a front crossing is only 'counted' if the salinity

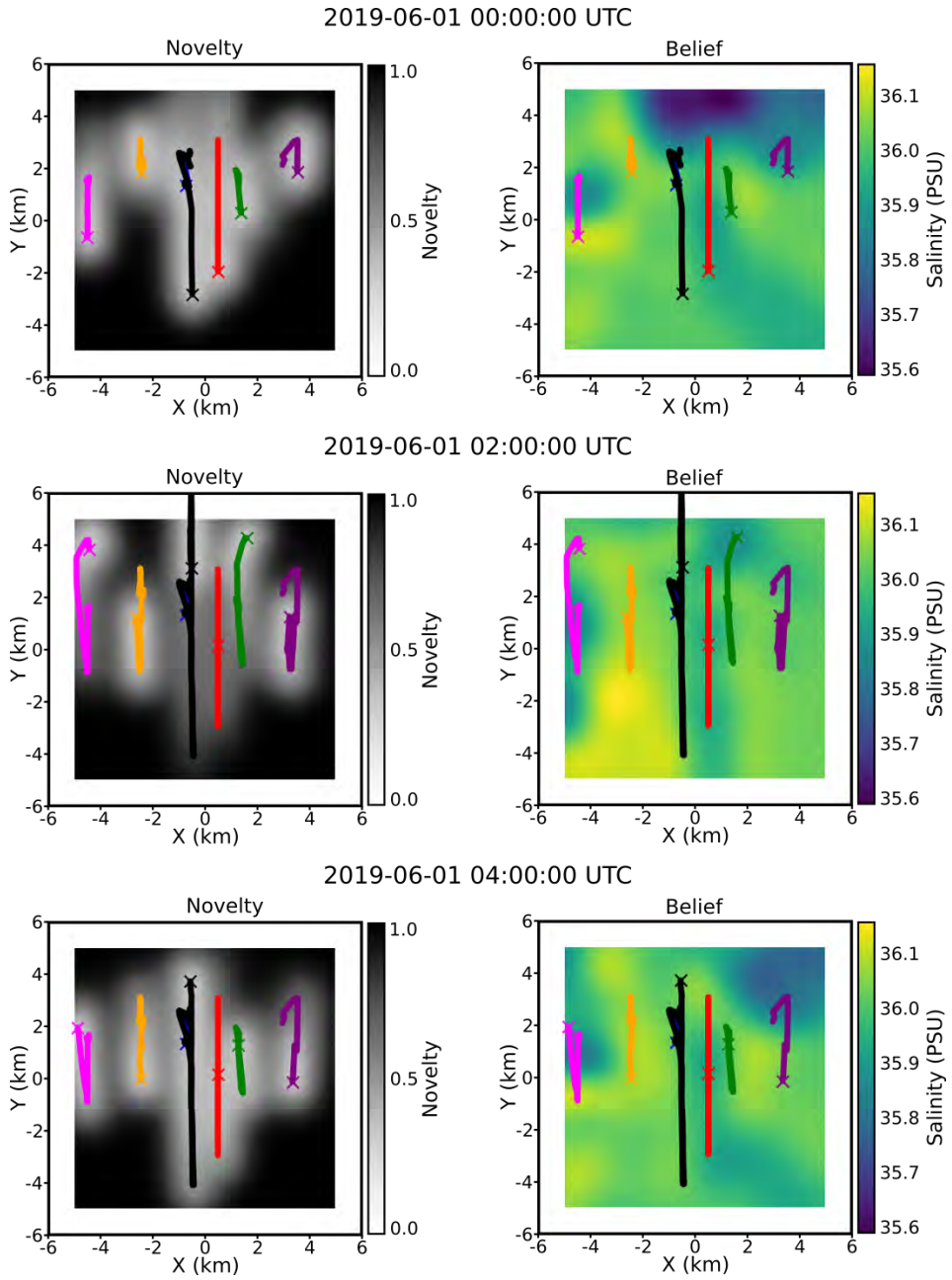


Figure 4.11: A 4-hour time series from the Expert Manual Control experiment. The left column shows the estimated salinity field in the 10 km square planning region. The right shows the corresponding novelty field for each snapshot. The paths are characteristic of the sampling pattern used by the ocean scientists: vertical, repeated sampling lines spaced at regular intervals. Each vehicle's current location is marked with an 'X'. The color code for each vehicle is the same as in Fig. 4.10

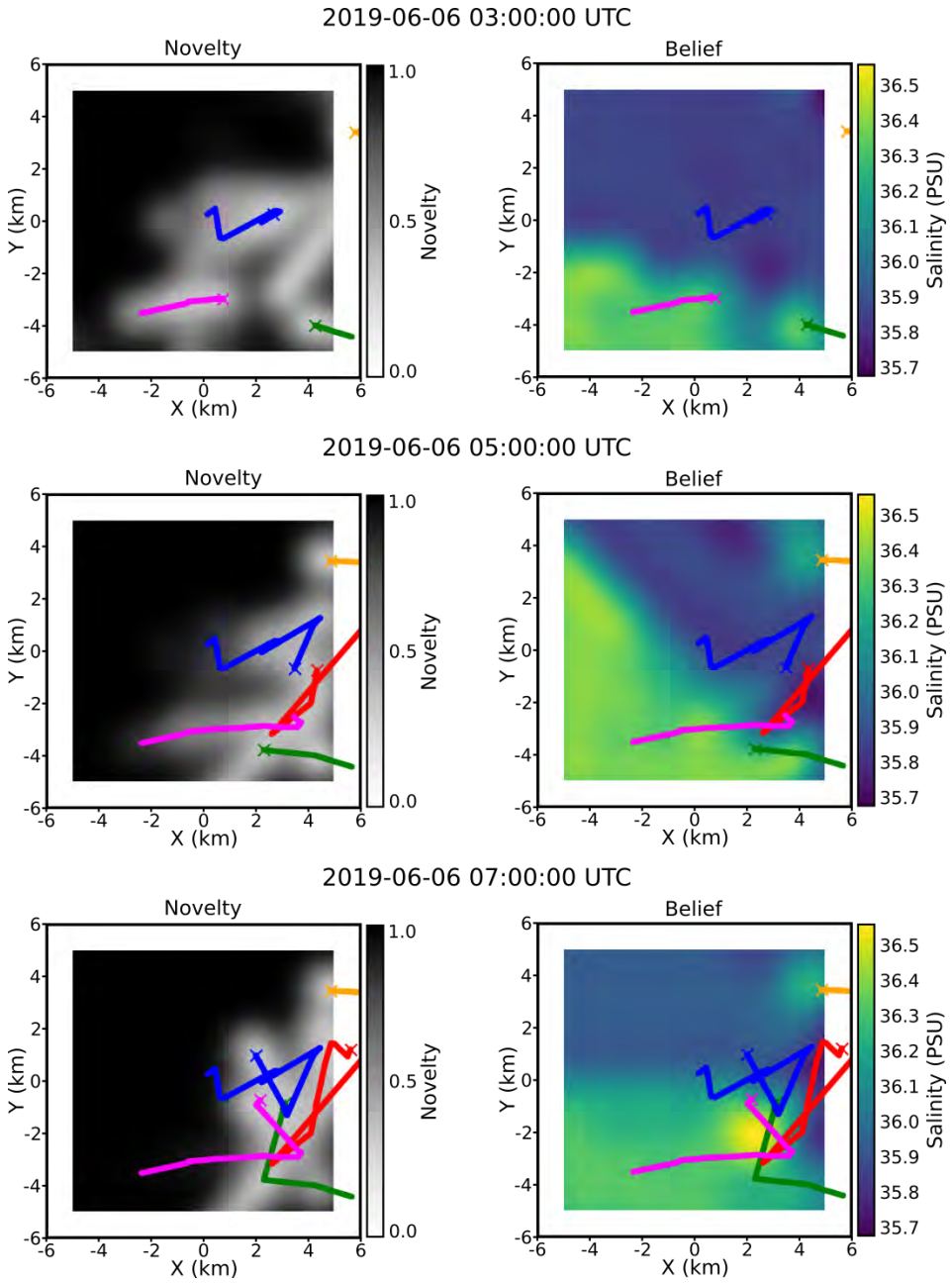
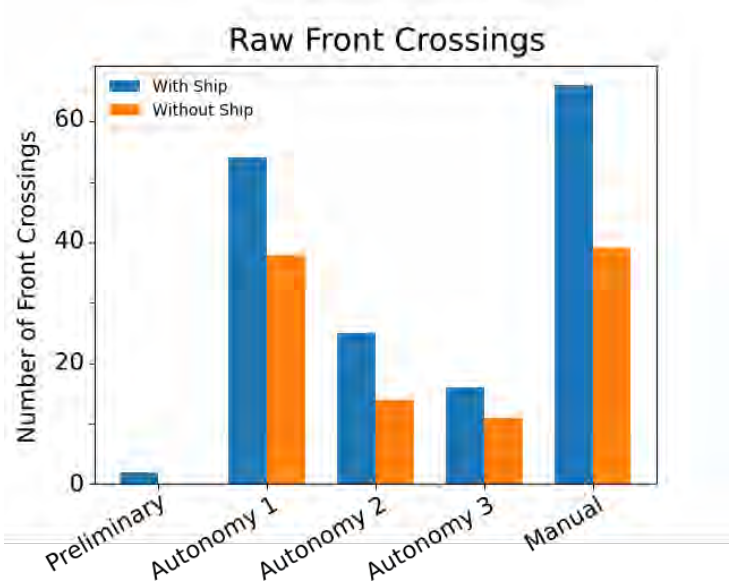


Figure 4.12: A 4-hour time series from Autonomy Experiment 1. The autonomy system is seen sampling along a salinity front. The gliders shown, in pink and green adapt their sampling, turn north to continue to track the front as the front moves through the planning frame. Since the pink and green gliders are sampling the front, the blue and red ROSS vehicles can use their higher speed to move away from the front and explore.

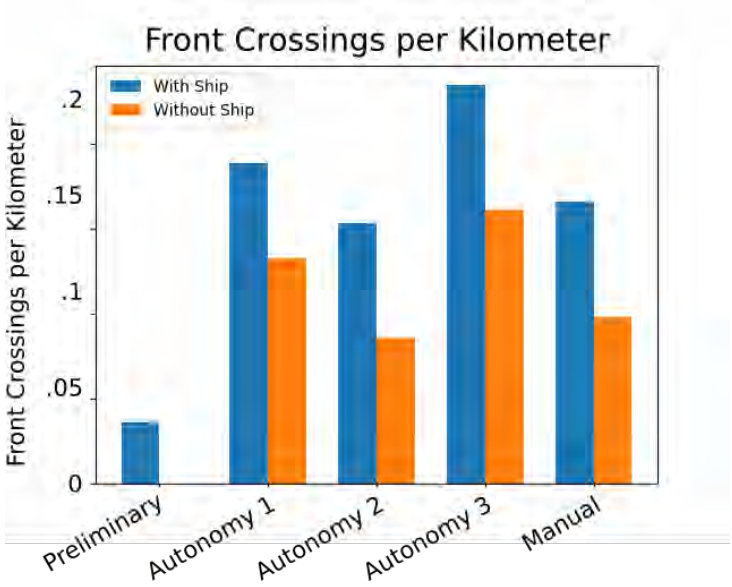
signal exceeds the threshold by -0.05 PSU for a falling salinity signal and 0.05 PSU for a rising salinity signal. Both the raw counts and the normalized front-crossing results are presented in Fig. 4.13.

Our results show that the performance of the autonomy system is competitive with that of the hand-designed survey patterns, with the performance by the autonomy system exceeding that of the manual operation in experiments 1 and 3. However, this evidence is not sufficient to conclude that the autonomy system outperforms the expert operation in all cases, since there are many confounding factors that may have influenced these results. We lack a ground-truth representation of the ocean environment in the survey area during the testing, and must rely on post-facto analysis to estimate the salinity value at the front. Furthermore, since the experiments were not conducted in parallel, the likelihood that the ocean conditions are exactly the same across trials is essentially zero. However, despite these factors, it is evident that the ability of our autonomous front-tracking system is at least capable of achieving competitive performance in terms of the number of front crossings when the vehicles were controlled by highly experienced ocean scientists.

The second metric to evaluate the performance of the autonomy system was to compute the Root Mean-Square Difference (RMSD) between the beliefs generated using only the data collected by the robotic assets (ROSS and gliders), and the beliefs generated when the data from the ship is incorporated. Apart from the preliminary autonomy testing, where the R/V Pelican was used as a stand-in for the ROSS in order to achieve heterogeneity in the deployed assets, the ship was not used as a robotic asset during the other autonomous experiments. Instead, the ship was under manual control and per-



(a)



(b)

Figure 4.13: Performance of the system across four autonomy experiments and during manual operation in terms of the number of front crossings. The plot in (a) shows the total number of front crossings for each experiment, while the plot in (b) shows the number of front crossings normalized by the distance travelled by all vehicles during each experiment.

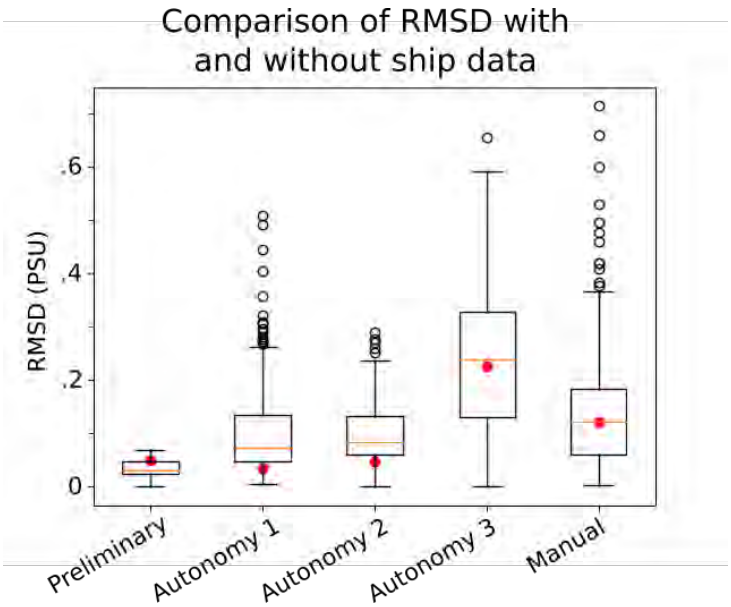


Figure 4.14: Comparison of RMSD across all experiments. The orange line in each boxplot shows the median difference, while the red dot indicates the difference at the end of each experiment.

forming profiling sections. The belief using the ship data is as close to ground truth as we can get during the field trials. The RMSD is computed by generating a pair of 10 km x 10 km beliefs, one with the ship data, and one without. These beliefs are uniformly sampled at a resolution of .1 km, and the RMSD is computed. Fig. 4.14 shows a comparison of the difference across the five experiments.

A high RMSD indicates that the ship was important in discovering the salient salinity features in the environment, while a low RMSD suggests that the non-ship vehicles were able to map the environment well on their own. In the graph, the difference caused by withholding the ship data is greater for the manual experiment than all of the autonomy experiments, save for experiment 3. This result is not unexpected, since for autonomy

experiments 1, 2, and 3, the ship's data was not used as a part of the autonomy system, and therefore, the remainder of the vehicles mapped the front independent of the behavior of the ship. However, during portions of the manual experiment, the ship operated in the sampling pattern with the other vehicles, and therefore the sampling patterns for the non-ship vehicles were designed with the ship's contribution in mind.

While the RMSD results allude to the performance of the vehicles during each of the sampling experiments, they fall short of being a true comparison to the ground truth during the experiment. Instead, these comparisons provide insight as to how robust the different sampling techniques are to the loss of the best asset. We believe that this comparison is valid, despite the differences in the systems, since during the autonomy experiments the ship was also sampling the front under expert guidance. Additionally, during both the manual experiment and the autonomy experiments the ship also had to break off from its sampling pattern for various reasons, such as to warn off other vessels from the experiment area or to recover and refuel vehicles. In the case of the autonomy system, particularly in Experiments 2 and 3, the lower median difference between using and not using the ship's data in the belief construction suggests that the autonomy algorithms do a comparable or marginally better job of conducting the frontal sampling task than the oceanographers' hand-designed sampling patterns. A difference in performance between the autonomy algorithm and the manual piloting likely can be attributed to wider coverage of the experimental area by the vehicles during the autonomy experiments as a consequence of incorporating exploration into the MCTS objective function, and the adaptive targeting, as compared to the pre-proscribed patterns during the manual sampling.

4.4 Conclusion

In this chapter we presented an autonomous system for coordinating a heterogeneous multi-robot team in a sampling task along a salinity front. Extending a general single-robot Monte Carlo Tree Search planner via Sequential Allocation, we were able to develop a system capable of robust multi-robot coordination in the front identification, tracking, and mapping task. In order to achieve good performance in this task, we incorporated expert domain knowledge through new ways of representing and tracking marine front environments, as well as through the SA-MCTS objective function. The capabilities of our system were demonstrated both in simulated trials, as well as during a two-week deployment in the Gulf of Mexico, where we showed comparable performance to hand-designed sampling patterns created by expert ocean scientists. In doing so, we have bridged the gap between the state-of-the-art information gathering algorithms tested in small-scale deployments and the limited adaptability of autonomy currently used in large-scale ocean deployments and we demonstrated that our SA-MCTS algorithm and autonomy system are sufficiently robust to handle the challenges of an extended trial at sea. Additional discussion of our autonomy system, along with its Decision Support User Interface can be found in our journal article [71].

One key requirement that enables us to incorporate domain knowledge relating to topological features, such as the salinity front addressed in this chapter, into a planning problem is that the existence of the feature must be known. There is a wide array of work that is focused on extracting topological representations of domestic environments [32, 10, 28]. However these methods cannot be applied to the marine environments

where we would like to conduct autonomous sampling missions, since they rely on the partitioning of the physical operating environment via walls and doorways. In the next chapter we will discuss a method for identifying topological features in environments where these features are not present.

Chapter 5: Information Gathering with Topological Hotspot Graphs

In this chapter, we present a novel method for extracting a topological environmental representation of the information-space of a robot in an instance of an Informative Path Planning problem. Prior work in building topological representations is mainly limited to domestic and other indoor environments, where walls and doorways provide convenient features for identifying topologically distinct regions. Once identified, a topological representation can then be used to make high-level decisions about useful regions to gather data more efficiently than an exhaustive search of the space, since a search over a topological graph scales with the complexity of the graph rather than with the size and dimensionality of the space. Our approach expands the types of problem domains where robots can benefit from topological representations, such as our methods discussed in in Chapters 3 and 4, improving their performance in information gathering tasks.

Our method builds on the Fast Marching method [14] to determine the extent in the metric space as well as the connectivity of a set of topological features. This results in a graph, the vertices of which correspond to the hotspot regions, and the edges are show their connectivity. In order to find the optimal allocation of time to the edges and vertices of this graph, we developed a closed-form solution, which uses Lagrange Multipliers [80] to solve the time allocation problem for a given set of vertices and edges. By searching over the space of all possible paths through the graph, we can identify the

optimal schedule for the robot, which is then combined with a coverage algorithm to produce the robot's final path.

Chapter 5.1, outlines the three steps of our method to construct a topological representation of the environment and use it to solve the informative path planning problem. In Chapter 5.2, we show that the topological features identified by our HHIG planner match those of the underlying environment. The benefits of our proposed method will be demonstrated, alongside our next contribution, the Topology-Aware Self Organizing Map algorithm, in Chapter 6. A preliminary version of this work has appeared in [81], and it will appear in a journal paper that is currently under preparation [82].

5.1 Method

Our approach for using a topological graph to plan an informative path can be broken down into three component steps:

1. Identify hotspots in the environment and construct a compact topological representation of these hotspots and their connectivity as a graph.

2. Plan a maximally informative path through these hotspots, deciding which hotspots are worth visiting, scheduling an amount of time to spend at each of them, and deciding which edges to use to travel between the chosen hotspots.

3. Transform this high-level plan over the graph into one that can be executed on a vehicle by creating sub-plans within each hotspot.

Each of the steps will be discussed in detail in the following section.

5.1.1 Topological Graph Construction

The first step in our approach is to reduce the space of possible paths by clustering sets of high-value locations into larger hotspot regions in a way that preserves their underlying topology. Using the robot's estimate of the likelihood of making a positive observation of a phenomenon at a particular location in the environment, $I_{global} : \mathbb{R}^2 \rightarrow [0, 1]$, we will construct a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ that captures the underlying topology of I_{global}.

Each $v_i \in \mathcal{V} = \{v_1, v_2, ..., v_n\}$ is a region in space containing one or more points of interest, which we define as local maxima of $I_{global}(x, y)$. In an ocean monitoring task, a relative increase in the occurrence of a phenomena, such as phytoplankton can provide valuable data about the causes of larger oceanic trends. Each v_i has a corresponding estimate of its local reward function, $\hat{I}_i(t_i)$, where t_i is the amount of time spent at v_i. This estimate can be any nondecreasing differentiable function, and in this work we choose to model it on the exponential reward function defined in [51], which captures the submodular nature of the information gathering task:

$$\hat{I}_i(t_i) = a_i(1 - e^{-b_i t_i}), \tag{5.1}$$

where a_i is the total amount of information contained in v_i. The accumulation rate of information at the hotspot is given by b_i and is a function of both the size of the hotspot, A_i, and the sensing radius of the robot, obs_r:

$$b_i = \frac{obs_r^2 \pi}{A_i}. \tag{5.2}$$

We assume that I_{global} is static during the planning and execution of a trajectory. If the robot visits the same vertex multiple times, then the time that the robot is considered to have spent at the vertex is the sum of all the time that it spends during each visit.

The vertices of $\mathcal{G}$ are connected by a set of edges, $\mathcal{E} = \{e_1, e_2, ..., e_m\}$, where each $e_i \in \mathcal{E}$ consists of a pair of opposite directed edges $\{\overrightarrow{e_i}, \overleftarrow{e_i}\}$. It is possible for a pair of vertices to be connected by more than one edge. A robot can only observe the information associated with a given edge, e_i once, by traversing it in either direction (by traversing either $\overrightarrow{e_i}$ or $\overleftarrow{e_i}$). The opposite edges $\overrightarrow{e_i}$ and $\overleftarrow{e_i}$ follow the same path through $\mathbb{R}^2$, and therefore have the same length. However, representing a bidirectional edge in this way will allow us to prune our path search space, offering speedups in the path planning step. This process is discussed further in Chapter 5.1.2.

To construct this graph, we begin by creating a discrete approximation of the global information function by sampling it in a regular grid pattern. Using this discrete approximation of I_{global}, we collect the local maxima and minima into two sets $\mathcal{S}_{max}$, and $\mathcal{S}_{min}$, respectively. $\mathcal{S} = \mathcal{S}_{max} \cap \mathcal{S}_{min}$. The elements of $\mathcal{S}_{max}$ are Points of Interest (PoI)s, as they represent locations where there is a relative increase in the global utility function. Conversely the elements of $\mathcal{S}_{min}$ are locations where there is a relative lack of the desired phenomena, and therefore should be avoided. An example of this step is shown in Fig. 5.1a.

Once $\mathcal{S}$ is constructed, it is used as the seed points for our modified Fast Marching expansion method. The standard FM algorithm approximates a solution to the Eikonal Equation [14][83]:

$$||\nabla u|| = \tau,$$

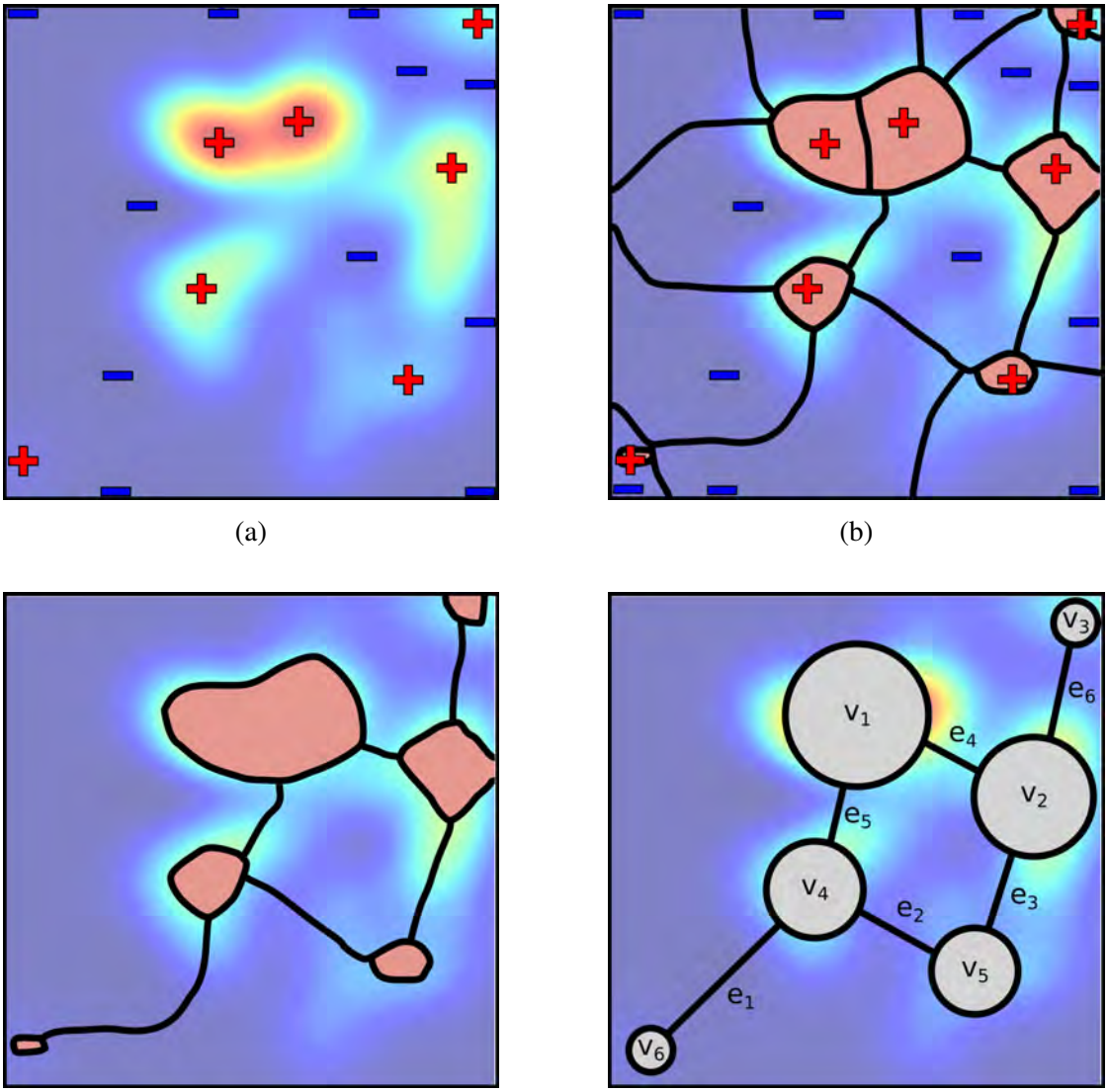


Figure 5.1: Hotspot Identification Process. Fig 5.1a shows the selected maxima and minima. Figs 5.1b shows the growth of the labelled regions around each s_{max} and $s_{min} \in \mathcal{S}$ Fig 5.1c shows the merging of adjacent maxima regions and the creation of edges between each hotspot. Finally, Fig 5.1d shows the resultant topological graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

where τ is a cost function that defines the speed of travel through the environment, and u is the function that describes the minimum cost-to-go distance between a point, $x_{i,j}$ in the environment and a starting location, where $u_{i,j} = u(x_{i,j})$. The FM algorithm leverages an upwind scheme to propagate the first-order estimate of u as a wavefront through an environment. On a Cartesian grid with spacing h, this propagation can be accomplished by estimating the magnitude of the gradient, ∇u, in both the x and y directions using

$$\begin{aligned} ||\nabla u_{i,j}||^2 \approx \tau_{i,j}^2 = [max(D_{i,j}^{-x}, -D_{i,j}^{+x}, 0)^2 + \\ max(D_{i,j}^{-y}, -D_{i,j}^{+y}, 0)^2], \end{aligned} \tag{5.3}$$

where the forward and backward steps in the x and y directions are defined as:

$$D_{i,j}^{+x} = \frac{u_{i+1,j} - u_{i,j}}{h},\ D_{i,j}^{-x} = \frac{u_{i,j} - u_{i-1,j}}{h},$$

$$D_{i,j}^{+y} = \frac{u_{i,j+1} - u_{i,j}}{h},\ D_{i,j}^{-y} = \frac{u_{i,j} - u_{i,j-1}}{h}$$

The upwind scheme uses a breadth-first update method to iteratively select a trial point that is then moved from the *frontier* set to the *accepted* set. The *accepted* set consists of all the nodes that are a part of the expanded area, while the *frontier* set consists of all the nodes that are adjacent to nodes in the *accepted* set but are not included in it. The trial node is selected as the node in the *frontier* set with minimal cost, $u_{i,j}$, since it is the next node to be visited by the wavefront as it propagates. Then, we update u for all of the trial node's neighbors, adding them to the *frontier* set if necessary.

If a neighbor, $x_{i,j}$ is adjacent to one point or one pair of opposite points in *accepted*,

termed P_1, then the time-of-first-arrival at $x_{i,j}$, $u_{i,j}$ is updated according to:

$$u_{i,j} = u_{P_1} + \tau_{i,j},$$

where $u_{P_1} = min(u(P_1))$. Similarly if there are at least two non-opposite adjacent points or pairs of points, P_1 and P_2 with corresponding minimum costs u_{P_1} and u_{P_2}, then $u_{i,j}$ is updated by

$$u_{i,j} = min(u_{P_1}, u_{P_2}) + \tau_{i,j},$$

if $\tau_{i,j} \leq |u_{P_1} - u_{P_2}|$. Otherwise the update is given by

$$u_{i,j} = \frac{1}{2}\left(u_{P_1} + u_{P_2} + \sqrt{2\tau_{i,j}^2 - (u_{P_1} - u_{P_2})^2}\right).$$

We adapt this standard FM formulation by varying $\tau_{i,j}$ based on whether $x_{i,j}$ is a descendant of a member of $\mathcal{S}_{max}$ or $\mathcal{S}_{min}$. We propagate the max and min labels from the original points of interest. Each time a node is expanded it inherits the classification of its parent in $accepted$. For an $x_{i,j}$ that descends from a $s_{max} \in \mathcal{S}_{max}$, we define $\tau_{i,j}$ as $1 - I_{global}(i,j)$. For an $x_{i,j}$ that descends from a $s_{min} \in \mathcal{S}_{min}$, $\tau_{i,j} = I_{global}(i,j)$. Regions expanding from maxima expand more easily in high-information areas, while regions expanding from minima expand more easily to cover low-information areas. This expansion process is shown in Figs. 5.1b and 5.1c.

To construct a graph from the labelled regions, we merge adjacent regions grown from maxima, as depicted in Fig. 5.1d, and label each combined region as a hotspot, and add it to $\mathcal{V}$. The interfaces between regions grown from minima become the edges

between the hotspot vertices. These interfaces are equidistant between local minima over I_{global}, and therefore correspond to relatively information-rich paths between two vertices. The resultant graph for a sample environment is shown in Fig. 5.1d.

It is possible for environments to exist where the topological graph construction fails, such as when a hotspot is completely enclosed by a single region grown from a local minima, or a local minima is enclosed by a hotspot. In the first case, we simply use Fast Marching to extend an edge from the isolated hotspot to the nearest edge or hotspot, connecting it to the graph. We do not address the situation where a hotspot encloses one or more local minima, as it has no effect on the topological structure of the resulting graph; we simply end up with a hotspot that bounds one or more areas that are not included in the hotspot. If a particular domain requires that hotspots be solid, a simplex-based method [9] could be employed to identify and eliminate the holes in a hotspot. However, depending on the domain and application, a hole that is a result of multiple local minima may or may not need be eliminated in this manner.

5.1.2 Hotspot Scheduling

In order to plan a path using the graph, the robot must decide which of the vertices it should visit, and in what order it should visit them. Similar to the orienteering-style problems discussed by Yu *et al.* [51, 39], the problem that we seek to solve is to identify an informative path, $\mathcal{P} = (\mathcal{V}_\mathcal{P}, \mathcal{E}_\mathcal{P}, \mathcal{T})$, where $\mathcal{V}_\mathcal{P} \subset \mathcal{V}$ is the set of unique vertices visited along the path, $\mathcal{E}_\mathcal{P} \subset \mathcal{E}$ is the set of edges that the robot traverses, and $\mathcal{T}$ is the set of times, t_i, spent at each $v_i \in \mathcal{V}_\mathcal{P}$. However, in our approach, we do not restrict the

path that the robot follows to be a tour. Instead, the robot can begin and end its path at any vertex.

We begin the scheduling process by constructing a tree with its root at the vertex of $\mathcal{G}$ corresponding to the robot's initial position. We expand the tree by adding child nodes corresponding to each of the node's neighbor vertices. These neighbors include vertices arrived at by following edges back to previously visited vertices, since it can be necessary to backtrack in order to visit new, unexplored areas of the graph. The tree is expanded until the budget constraint,

$$c(\mathcal{P}) = \sum_{i=1}^{|\mathcal{V}_\mathcal{P}|} t_i + \sum_{i=1}^{|\mathcal{E}_\mathcal{P}|} \frac{\ell_i}{vel_r} \leq B, \tag{5.4}$$

is met, where ℓ_i is the length of edge $e_i \in \mathcal{E}$, and vel_r is the robot's velocity.

While the number of paths can potentially be quite large, it kept relatively low by several factors. Chief among these is the fact that the graphs we develop are relatively sparse, rarely consisting of more than 10 vertices. However, graphs with a particularly high branching factor can lead to an intractable number of paths. Additionally we are able to prune paths that are guaranteed to be worse than paths already considered. Since there is no additional benefit to re-visiting a given vertex multiple times versus simply remaining at that same vertex for longer during a previous visit, we can stop expanding the tree if we would expand the same directed edge again. Attempting to expand a directed edge that has already been traversed means that we have previously visited each of the vertices incident to the edge, and that we have already observed any information contained within the edge.

Since the vertices of our graph correspond with hotspot regions, they have nonzero area, and therefore there can be some distance between the locations in $\mathbb{R}^2$ where the edges connect to the vertices. To determine the time, t_i, that is spent at a given vertex on the candidate path, we first compute the minimum amount of time that the robot is required to spend in each v_i along the path. Each time the robot visits v_i, we compute the amount of time the robot will need to take to travel between its entry and exit edges. Summed across each visit to v_i, this time, t_i^-, is the minimum amount of time that the robot is required to spend in v_i. We can then compute the amount of our budget remaining, R, using

$$R = B - \sum_{i=1}^{|\mathcal{V}_\mathcal{P}|} t_i^- - \sum_{i=1}^{|\mathcal{E}_\mathcal{P}|} \frac{\ell_i}{vel_r}. \tag{5.5}$$

In order for the robot to utilize this remaining budget, we assign each vertex an additional amount of time t_i^+ where the total time spent at v_i is $t_i = t_i^+ + t_i^-$.

We developed a closed-form solution for calculating the values for $t_1^+, t_2^+, ..., t_{|\mathcal{V}_\mathcal{P}|}^+$ that maximize

$$\sum_{i=1}^{|\mathcal{V}_\mathcal{P}|} I_i(t_i) \; s.t. \; \sum_{i=1}^{|\mathcal{V}_\mathcal{P}|} t_i^+ \leq R, \tag{5.6}$$

using a Lagrange Multiplier Method [80] to solve the resource-constraint problem inherent in allocating R among $\{t_1^+, t_2^+, ..., t_{|\mathcal{V}_\mathcal{P}|}^+\}$. We construct our Lagrange Function, $\mathcal{L}$, using the Lagrange Multiplier variable, λ,

$$\mathcal{L}(t_1^+, t_2^+, ..., t_{|\mathcal{V}_\mathcal{P}|}^+, \lambda) = \sum_{i=1}^{|\mathcal{V}_\mathcal{P}|} I_i(t_i) + \lambda(R - \sum_{i=1}^{|\mathcal{V}_\mathcal{P}|} t_i^+). \tag{5.7}$$

We take the partial derivative with respect to each $t_i^+ \in \{t_1^+, t_2^+, ..., t_{|\mathcal{V}_\mathcal{P}|}^+\}$, as well as λ.

Setting these equal to 0 yields the following system of equations:

$$\forall\ 1 \leq i \leq |\mathcal{V}_\mathcal{P}|,\ \frac{\partial \mathcal{L}}{\partial t_i^+} = a_i b_i e^{-b_i(t_i^+ + t_i^-)} - \lambda = 0 \tag{5.8}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = R - \sum_{i=1}^{|\mathcal{V}_\mathcal{P}|} t_i^+ = 0. \tag{5.9}$$

We compute the optimal solution by first selecting an arbitrary vertex. Without loss of generality, let this vertex be v_1. We may then solve for the time spent at each other vertex, t_i with respect to the time spent at this reference vertex, t_1, by setting the corresponding pair of equations in Equation 5.8 equal to each other:

$$t_i^+ = \frac{-ln\left(\frac{a_1 b_1}{a_i b_i}\right) + b_1(t_1^+ + t_1^-)}{b_i} - t_i^-. \tag{5.10}$$

Substituting Equation 5.10 into Equation 5.9 gives

$$t_1^+ = \frac{R - \sum_{i=2}^{|\mathcal{V}_\mathcal{P}|}\left[\frac{ln\left(\frac{a_1 b_1}{a_i b_i}\right)}{b_i} - \frac{b_1 t_1^-}{b_i} + t_i^-\right]}{1 + \sum_{i=2}^{|\mathcal{V}_\mathcal{P}|}\frac{b_1}{b_i}}, \tag{5.11}$$

which we can use to solve for t_1^+.

Taken together, Equations 5.10 and 5.11 can be used to compute the optimal values for all $\{t_1^+, t_2^+, ..., t_{|\mathcal{V}_\mathcal{P}|}^+\}$ along a given path.

The process used to calculate the schedule for the robot is outlined in Algorithm 1. Since each node in the tree corresponds to a unique path through the graph, for each node in the tree we can recover this candidate path by re-tracing the path through the

tree from the node to the root. By identifying the set of vertices and edges visited along this path, we can form an instance of the scheduling problem. By solving this problem for each unique path, we can select the $\mathcal{P}$ that maximizes our expected reward, thus resulting in the optimal path for the robot.

Algorithm 3 Hotspot Scheduling

1: **function** HOTSPOTSCHEDULE($\mathcal{G}$, B)
2: tree = constructTree($\mathcal{G}$, B)
3: **for each** node **in** tree **do**
4: $\mathcal{P}$ = tracePathToRoot(node, tree)
5: $\{t_1, t_2, ...t_{|\mathcal{V}_\mathcal{P}|}\}$ = schedule($\mathcal{P}$)
6: $\hat{I} = \sum_{i=1}^{|\mathcal{V}_\mathcal{P}|} \hat{I}(t_i)$
7: $\mathcal{P}^*$ = $\text{argmax}_{\mathcal{P} \in tree}(\hat{I})$
8: **return** $\mathcal{P}^*$

5.1.3 Path Planning

Once the optimal schedule for the graph is computed, and the budget for each node is assigned, the robot needs to plan a path that uses the assigned budget within each node. We use a greedy-coverage algorithm to quickly compute a coverage path for the vertex. This simple algorithm works well, since the topological hotspot identification and segmentation component of our approach identifies areas that are filled with only high-information areas, making a simple coverage approach more effective than it would otherwise be. However, more sophisticated planners, such as Branch and Bound [6], or stochastic trajectory optimizers, such as STOMP [84], may be considered to compute more optimal coverage paths at the expense of computation time.

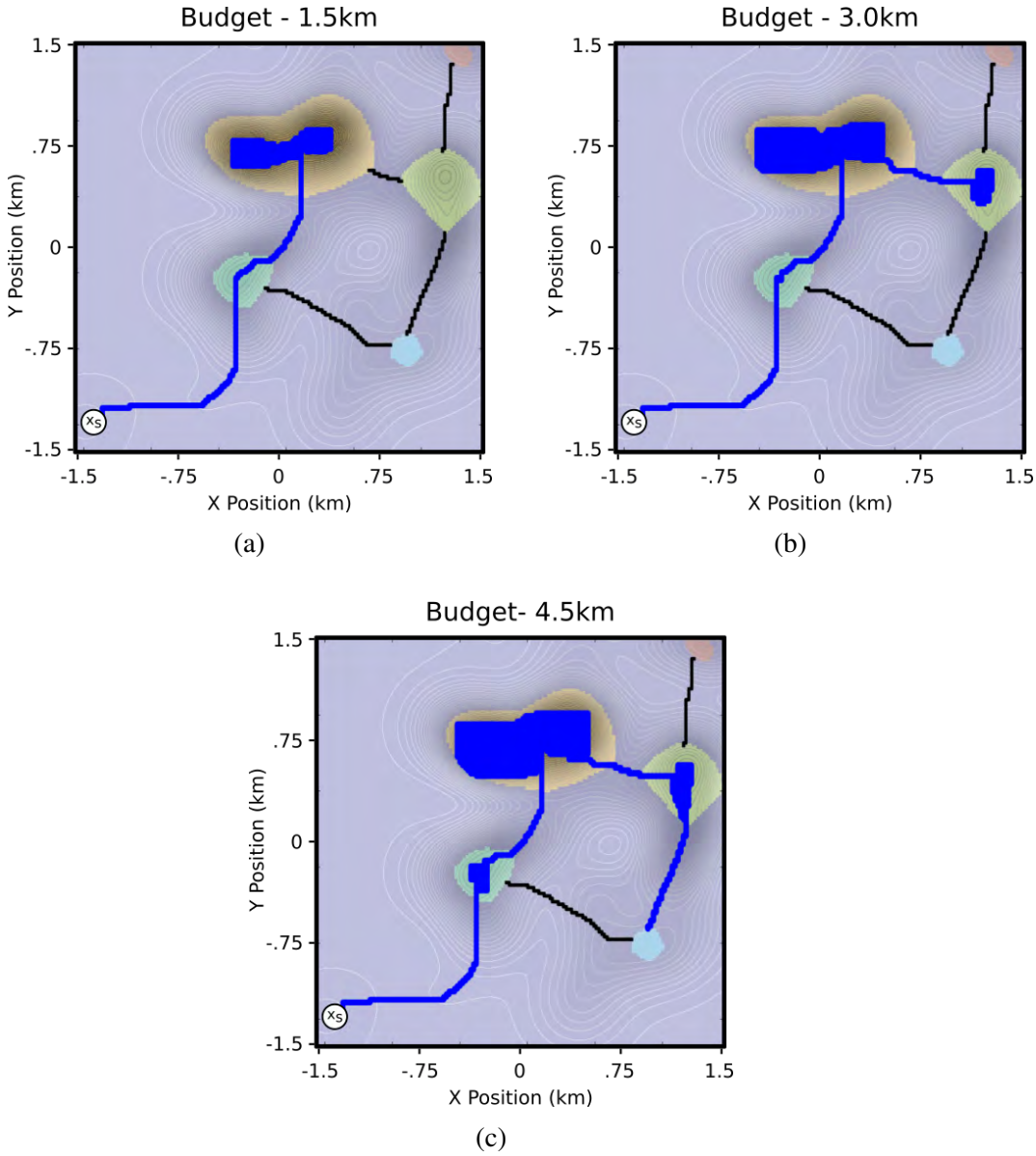


Figure 5.2: Informative paths planned using budgets of (a) 1.5 km, (b) 3.0 km, and (c) 4.5 km. The area explored by the robot is shown in blue. As the budget increases, our method is able to balance the additional information gained by continuing to explore the current hotspot with the information gained by exploring new hotspots.

To compute the greedy-coverage path, while the robot has budget remaining to travel to its goal point (if it has one), the robot greedily selects the most informative location from its unvisited neighbors. If no such neighbors exist, the algorithm selects a location randomly. Then, the robot moves to the new goal and repeats the process. Once the remaining budget is equal to the distance to the goal point, the robot moves toward the goal, preferring to move into more informative locations that it has not yet observed. If there is no goal point, such as on the last node of the path, then the robot continues to add to the path greedily until it runs out of budget.

By combining the paths along the selected edges with the greedy-coverage paths within the vertices, we obtain the final path for the robot. Some sample paths created on our Monterey Bay dataset are shown in Fig. 5.2.

5.2 Results

We evaluated our method both on a set of simulated environments and on a dataset of ocean bioacoustic activity collected using a Slocum glider [85] in Monterey Bay, CA. The simulated environments were created by taking the sum of 50 Gaussian functions randomly distributed in a 4-connected 35 km by 35 km world discretized at a grid resolution of 350 m. The information value across the field is normalized to be between 0 and 1.

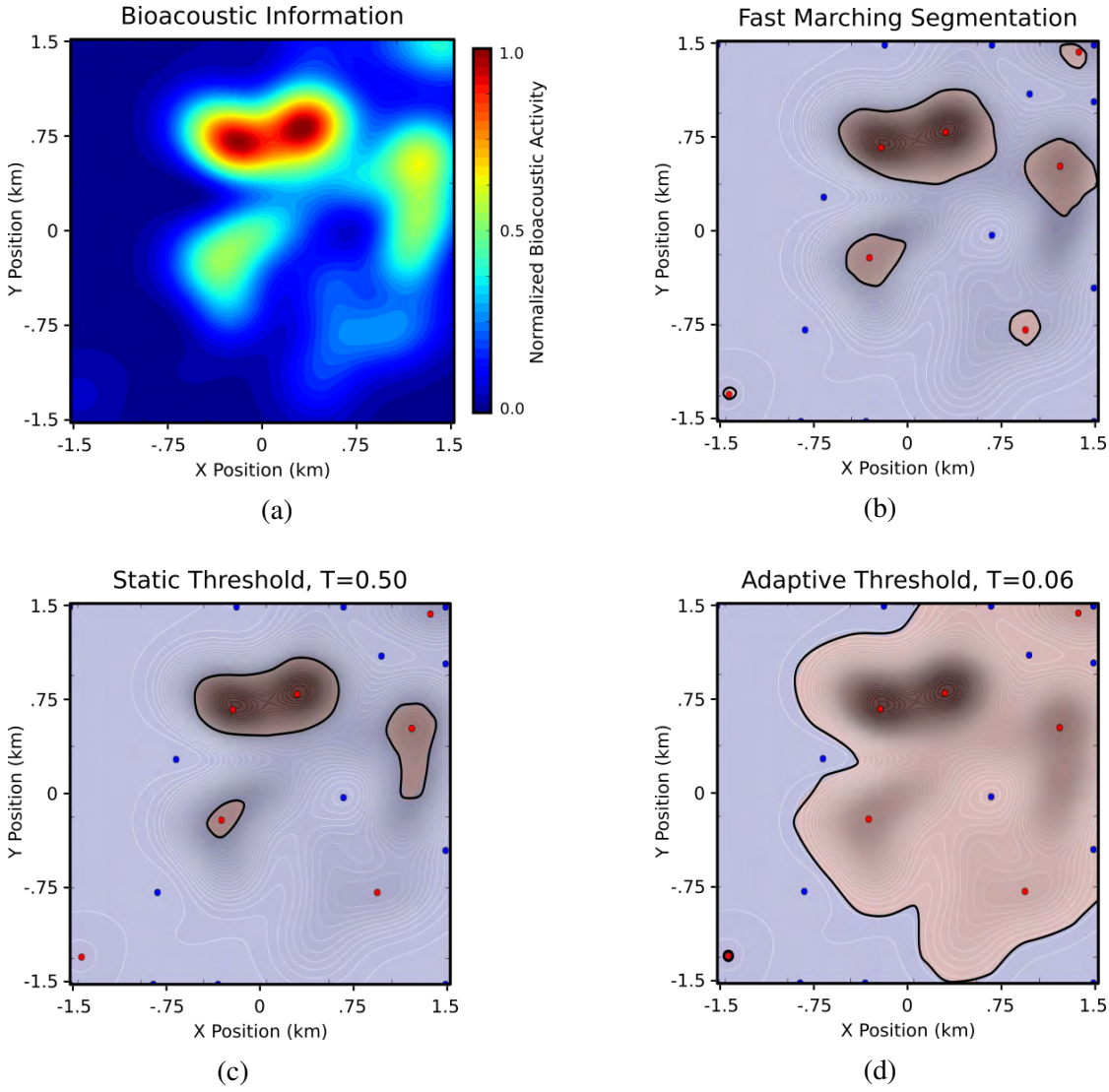
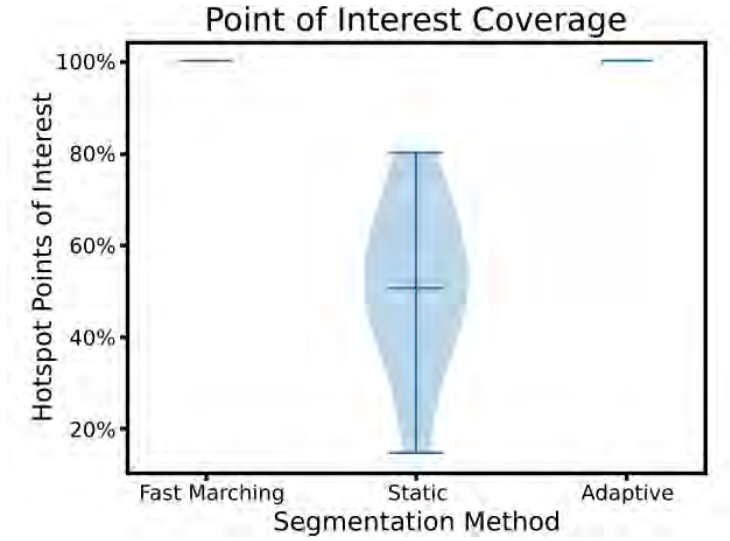
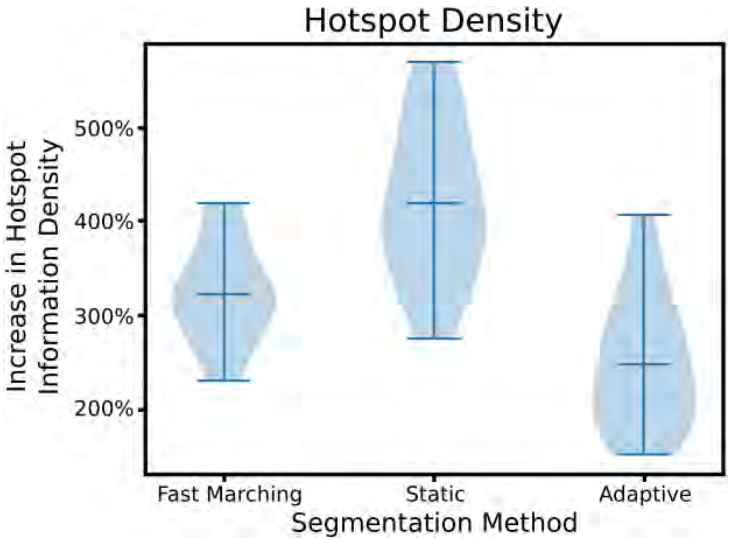


Figure 5.3: Comparison of segmentation methods. Areas labeled as hotspots are shown in red. (a) A sample map of bioacoustic activity collected during a set of trials in Monterey Bay, California. (b) The results of our our Fast Marching based method for identifying hotspots. Unlike a statically defined threshold of 0.5 (c), our approach does not omit any points of interest in the generation of hotspot regions. Additionally, our approach is more selective than simply setting the threshold low enough to capture all points of interest, as seen with the adaptive thresholding technique in (d).



(a)



(b)

Figure 5.4: Results comparing our Fast Marching Hotspot Segmentation algorithm to static thresholding (Activity> 0.5) and adaptive thresholding. Both our method and adaptive thresholding capture 100% of points of interest within regions labelled as hotspots. However, our approach outperforms the adaptive thresholding in terms of hotspot density. The static thresholding method does produce denser hotspots; however it fails to capture a significant portion of the points of interest in the environment

5.2.1 Hotspot Segmentation

To evaluate the effectiveness of our hotspot segmentation approach, we compared our method of segmentation to standard thresholding, which labels as hotspots each point in space greater than a threshold. We compare to a static threshold of 0.5, as well as an adaptive threshold, set to capture all PoIs. This adaptation is done by setting the threshold value equal to the value of the lowest PoI. Each thresholding method is demonstrated on the Monterey, CA environment in Fig. 5.3. A visual examination of the three segmentation techniques highlights the drawbacks of the two thresholding approaches to segmentation. Using a constant threshold, as in [36], and [37], set at 0.5 is shown in Fig. 5.3c fails to capture a number of the PoIs, while adjusting this threshold to capture the lowest-valued PoI (at 0.05) collects nearly all of the the environment into one giant hotspot.

We compared the three segmentation methods: Fast Marching, Static Thresholding, and Adaptive Thresholding across a set of 20 simulated environments. In each of these environments, we compared the methods on both the number of Points of Interest captured and the hotspot density, which is the percent increase in average information between the hotspots and the whole environment. The results of these trials are shown in Figs 5.4a and 5.4b, respectively. Our method shows an increase in hotspot density over the adaptive thresholding method, while maintaining 100% of Points of Interest captured in hotspot regions. While our method does produce less-dense hotspots when compared to a static threshold, the static thresholding misses a significant portion of the points of interest. This fact is problematic from an ocean science perspective, since

many of the phenomena that we are interested in monitoring are indicated by a local deviation from the norm, rather than any global value. Furthermore, since the static threshold is a hand-tuned parameter, setting it correctly requires a significant amount domain knowledge, while our Fast Marching Hotspot Segmentation method requires no such parameter tuning.

5.3 Conclusion

In this chapter we presented a method for constructing a topological environmental representation of a marine environment by identifying a set of hotspots in the environment. We were able to use this graph in our informative path planning algorithm by searching over possible paths through the graph and using a closed-form analytic solution to optimally distribute time amongst the graph vertices, creating a schedule for an autonomous vehicle to follow.

While this method is capable of building a topometric representation that captures the topological structure of the information field mapped across the robot's state space, it is limited in the paths that it can consider since the robot is limited to only traversing between hotspots along the graph edges. In the next chapter, we will present a method that is not limited in this way, and instead partitions the path-space of the robot into unique homotopy classes. This alternative paradigm will allow us to more closely tie the topologies considered to the how the information reward function is realized by the robot's motion.

Chapter 6: Learning Topological Features with Self Organizing Maps

In this chapter we present our fourth contribution: a method that uses a topological model of an information field to improve the ability of a trajectory optimization algorithm to find the maximally informative path. We modify the Self Organizing Map (SOM) algorithm to allow it to adapt its graph topology during training. While the Hierarchical Hotspot Information Gathering (HHIG) planner discussed in Chapter 5 captures the topology of information function mapped over the state space of the robot through the formation of the hotspot graph, the paths it can consider are limited by the structure of the hotspot graph. To address this, instead of partitioning the environment directly, we can instead partition the space of all possible paths, Φ, into topologically distinct homotopy classes. Once this has been accomplished, we can select a representative path from each homotopy class and use Stochastic Gradient Ascent to optimize a path within each homotopy class. Fig. 6.5 illustrates our proposed method that uses a novel Topology-Aware Self Organizing Map (TA-SOM) to build an environment model, which can be used to compute unique homotopy classes.

In many previous domains where topological techniques have been utilized (e.g. [20][9]), the techniques rely on using physical obstacles to partition the space into distinct trajectory classes. However in field robotics domains, such as marine scientific data collection or aerial surveillance, physical obstacles such as islands or mountains can be few and far-between. Instead, we observe that the information function itself can

generate distinct homotopy classes of trajectories that span the environment. To enable the gradient-based optimizer to perform most effectively, the homotopy classes should each contain a single local maxima, and therefore, the topological features should be rooted in the local minima of the objective function. However, for a very noisy information function, there can be a high number of local minima, which will result in a large number of homotopy classes. Instead, we only consider the most important features that induce topological trajectory classes. Doing so greatly reduces the total number of features while maintaining the goal of optimizing trajectories in regions with near-convex objective functions. To accomplish this, we will utilize persistent homology to quantify the importance of features in the environment.

We first provide some background on persistent homology using simplicial complexes in Chapter 6.1. Then, we outline our TA-SOM algorithm in Chapter 6.2. Then, in Chapter 6.3, we demonstrate the effectiveness of this method, along with the Hierarchical Hotspot Information Gathering planner discussed in Chapter 5, and show how using topological information can improve the performance of a robot in the information gathering task over state of the art methods that do not exploit this information. These experiments are done in simulation, using real-world data taken from the Regional Ocean Modelling System model of Monterey Bay, California, as well as in a set of field trials conducted using a small autonomous boat on a pond. A version of this work has appeared in [86], and it will appear in a journal paper that is currently under preparation [82].

6.1 Background

For our TA-SOM, we need to alter the graph connectivity around the important topological features of the environment. To quantify the importance of a topological feature, we use a measure called persistence. Persistence has been used to quantify the importance of features in uncertain obstacle fields [87], as well as to set adaptive threshold to identify topologically distinct trajectories in an environment [88][9]. Central to computing the persistence of topological features is a function that links a scalar threshold parameter to a countable set of topological features. In [87], this function function is a threshold of the uncertain obstacle field that, when combined with a homotopy augmented graph, gives all homotopy classes present in the environment. In [9], the persistence function is a filtration of a simplicial complex.

A filtration of a simplicial complex, $\mathcal{K}$, is a function that maps a scalar parameter to a subcomplex of $\mathcal{K}$. More formally, if a function, f maps $\mathcal{K}$ onto $\mathbb{R}$, and f satisfies $f(\tau) \leq f(\sigma)$ when $\tau \subset \sigma$, for two complexes τ and σ, then f^{-1} maps $\mathbb{R}$ onto the subcomplexes of $\mathcal{K}$. This inverse function, f^{-1}, is a filtration of $\mathcal{K}$.

One commonly used filtration is the Delaunay-Čech Complex [89], illustrated in Fig. 6.1. To construct a Delaunay-Čech Complex, a set of vertices, $\mathcal{V}$, in the free space of a robot are sampled. Around each member of $\mathcal{V}$, a n-ball of radius r is constructed, where n is the dimensionality of the space being considered. The simplicial complex that results from the filtration is defined as the maximal simplicial complex defined by the Delaunay triangulation of that can be constructed within the space defined by the union of all n-balls. By varying the filtration value, i.e. the radius parameter, different

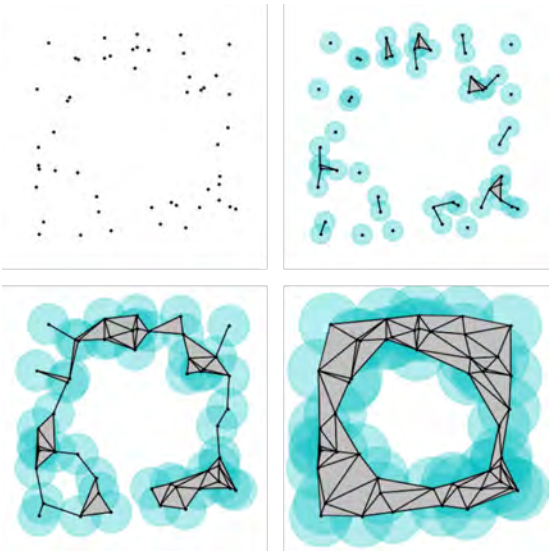


Figure 6.1: Delaunay-Čech complex for increasing values of r for a sample set of points. Areas shaded in blue are those used for the generation of the simplicial complex.

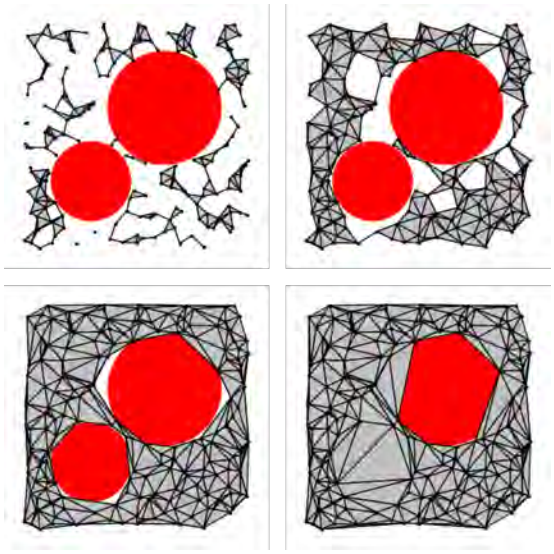


Figure 6.2: Resulting Delaunay-Čech complex for increasing values of r. True obstacles are shown in red, free space in white. However, for too large a value of r (bottom right), key topological features disappear.

simplicial complexes will result, with different topological features. An example of this is shown in Fig 6.1. One important property of the Delaunay-Čech Complex is that the resulting simplicial complex is guaranteed to retain the same topology as the union of n-balls [90]. This property is useful when computing homology groups in the resultant simplicial complex.

Persistence of topological features of a simplicial complex allows for the identification of major topological features within a space [9, 88]. As r increases, larger gaps between the sampled points are filled in by the filtration. The largest of these will correspond to obstacles in the space, where no vertices could be sampled. By constructing a

persistence diagram, these large gaps in the simplicial complex can be easily identified, and a value of r selected that preserves them [91]. Selecting too large of a value for r results in the obliteration of topological features, as shown in Fig. 6.2.

One way to visualize the persistence of features is through a persistence diagram, which documents the birth and death of topological features relative to a changing threshold parameter. As the threshold increases, topological features, such as path homotopy classes, holes, or connected components on a graph come into existence (i.e. are born). The existence of these features are tracked as the threshold continues to increase until they either cease to exist or are merged with a larger feature. At that point the feature is said to have died. The difference between the value of the threshold at the feature's birth and its death is the feature's persistence value.

An example of a persistence diagram that shows the persistence of two features on a graph is shown in Fig. 6.3. In this case, the filtration parameter is a threshold on the edge weight: defined as the line integral of the information field along each edge. Edges with weight greater than the threshold are added to the graph. As the filtration parameter increases, connected components merge into progressively larger components, causing the death of smaller components as they merge with larger ones. As components connect they can create holes in the graph triangulation that will eventually die as they are covered over by the increasing connectivity of the graph.

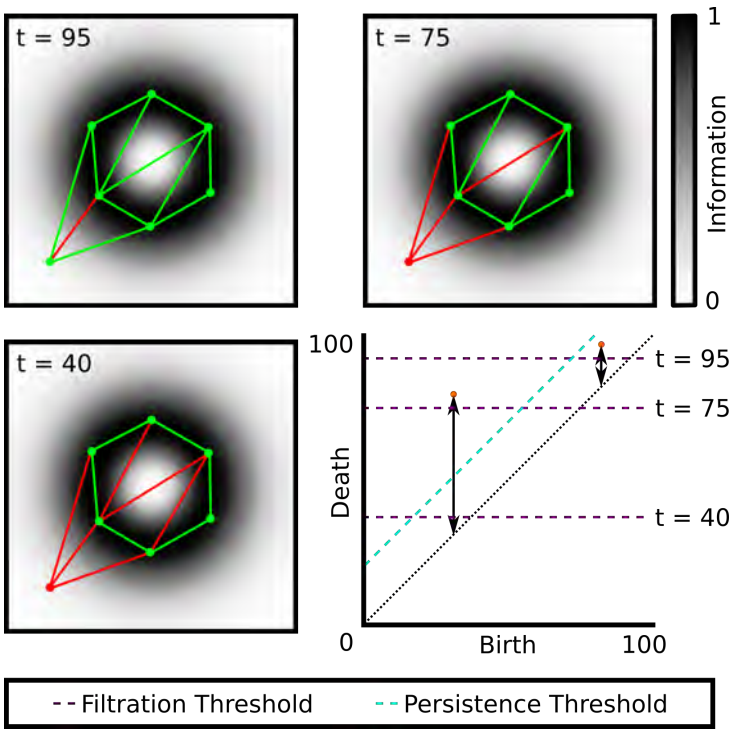


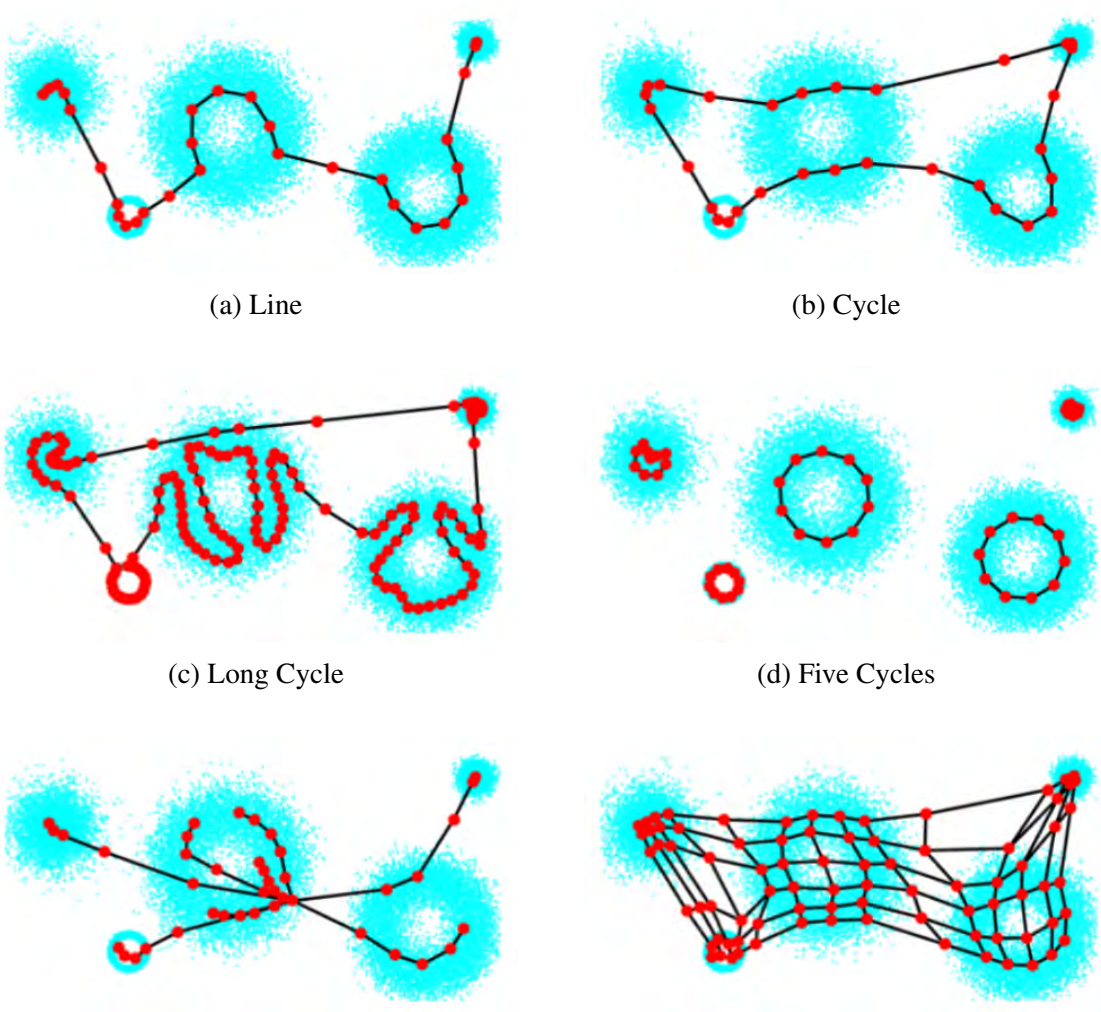
Figure 6.3: An example of a persistence diagram for a simplicial complex. Using the persistence of 1^{st} order topological features (orange dots), we identify a persistence threshold (blue), classify features as 'persistent' and 'ephemeral', and set a corresponding filtration threshold (purple), which is used to alter the SOM topology. Three different filtration thresholds and their corresponding simplicial complexes are shown.

6.2 Method

At a high level, the SOM algorithm is a method for fitting a graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, to a target function, $\Psi(\cdot)$ [92]. In robotics applications, SOMs have been used as an efficient method to approximate the solution to the Travelling Salesperson Problem [93][94], as well as the TSP's common extensions, such as the Orienteering Problem [95], and other information gathering tasks [96]. However, a significant issue with existing methods for SOMs is that the topology of the graph, $\mathcal{G}$, as defined by the structure of its connectivity is fixed prior to training. Different graph topologies can have an enormous impact on final positions of the graph vertices [2], and correctly choosing this topology can require a significant amount of expert domain knowledge. An example of the effect of different graph topologies on the final positions of $\mathcal{V}$ is shown in Fig. 6.4.

6.2.1 Topology-Aware Self Organizing Map

To address this limitation of standard SOMs, we improve on the existing capabilities of SOMs by allowing them to alter their network topology during training, so as to better mirror the structure of the underlying function. We accomplish this adaptation by interleaving the training process with a series of filtration steps, each of which modifies the graph topology around prominent features in the training distribution, removing and adding edges to the SOM's graph. Each train-filter cycle forms a training epoch. We continue training until a stopping condition is met, either a convergence criterion, or simply a maximum number of training epochs. Pseudocode is given in Algorithm 4. It relies on two sub-processes, the standard SOM training function, **TrainSOM**, and our



(a) Line

(b) Cycle

(c) Long Cycle

(d) Five Cycles

(e) Tree

(f) Mesh

Figure 6.4: The effects of varying graph topologies on the fit of a standard SOM. Source: Adapted from [2]

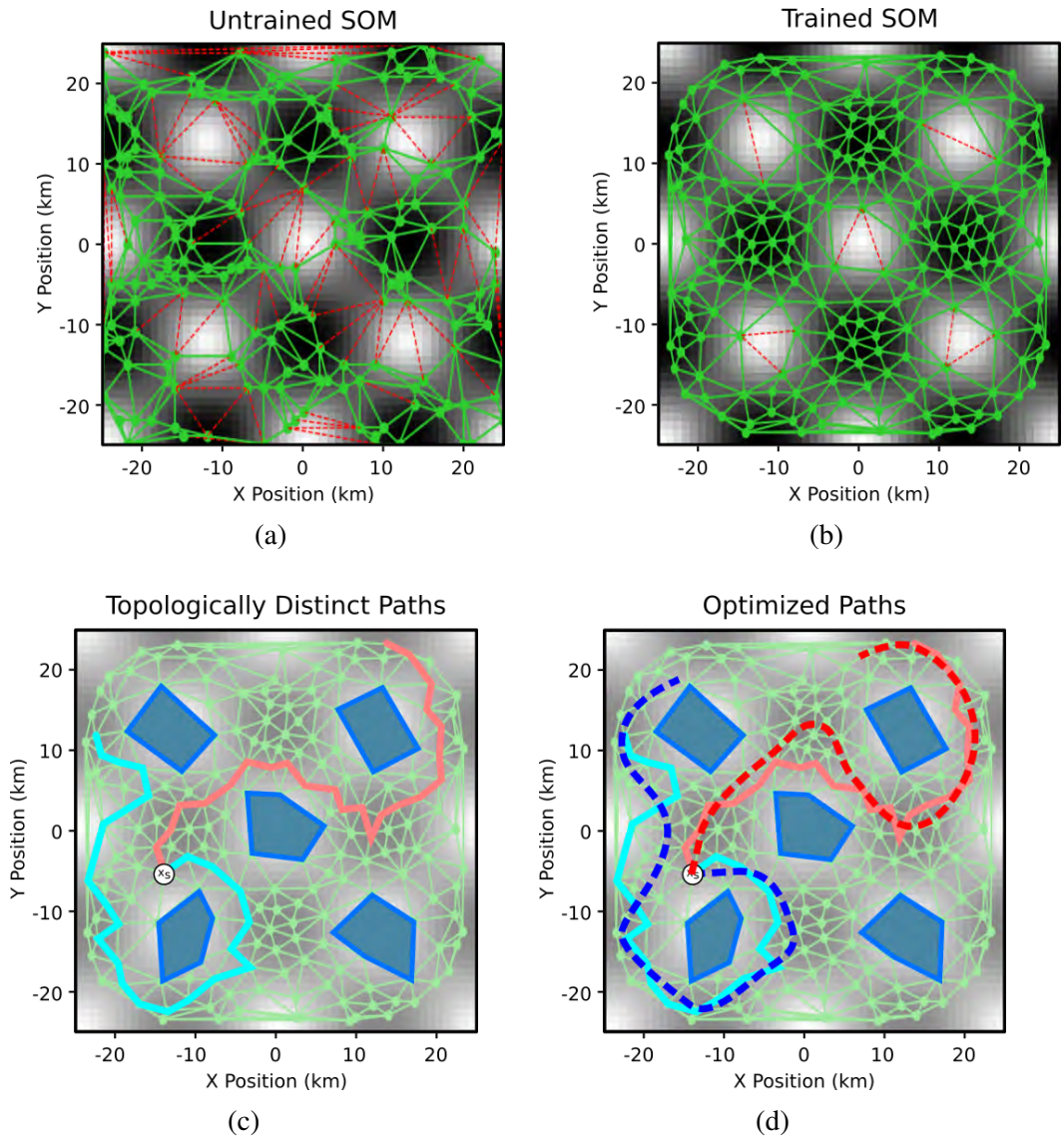


Figure 6.5: Training of an SOM on a Gyre world. Darker regions contain more information. Before training (a), the topological features identified by the SOM (areas with green edges) are driven by the sampling noise and do not correspond with the true topological features, (areas in white). Once trained (b) the identified regions align with the true features. The topology of the SOM (c) is used to identify a set of reference trajectories in unique homotopy classes. These trajectories are then optimized using Stochastic Gradient Ascent (d).

Algorithm 4 Topology-Aware Self Organizing Map

1: **function** TOPOLOGYSOM($I(\cdot), N$)
2: $\mathcal{V} \longleftarrow$ DrawSamples($N, I(\cdot)$)
3: $\mathcal{E} \longleftarrow$ DelaunayTriangulation($\mathcal{V}$)
4: $\mathcal{G} \longleftarrow (\mathcal{V}, \mathcal{E})$
5: **while** $\neg$ stopping **do**
6: $\hat{\mathcal{G}} \longleftarrow$ TrainSOM($\mathcal{G}, I(\cdot)$)
7: $\mathcal{G} \longleftarrow$ Filtration($\hat{\mathcal{G}}, I(\cdot)$)
8: **return** $\mathcal{G}$

proposed filtration function, **Filtration**.

The first step in **TrainSOM** is to draw a random sample, ψ, from Ψ. Then, the closest vertex in $\mathcal{V}$ to ψ, v^*, is computed. Once v^* is known, all the vertices of $\mathcal{G}$ (including v^*) are moved toward ψ. The distance each vertex $v_i \in \mathcal{V}$ is moved toward v^* is based on both an Information-weighted Euclidean distance between v_i and v^*, as well as on a neighborhood function. The information-weighted euclidean distance is given by

$$\text{InformationDist}(\psi, v) = \frac{|\overline{v\psi}|}{\int_{\overline{v\psi}} I(s)ds}, \tag{6.1}$$

where $\overline{v\psi}$ is the line segment between v and ψ. The information distance penalizes both long edges as well as edges that move through low-information regions. The neighborhood function captures the graph distance along $\mathcal{G}$ (i.e. the number of edges between v_i and v^*) to the range $[0, 1]$. We used a common form for the neighborhood function:

$$\text{Neighborhood}(v_0, v_1, \mathcal{G}) = \frac{1}{1 + \text{GraphDist}(v_0, v_1, \mathcal{G})^\gamma},$$

where γ is a hand-tuned weighting parameter that controls the decay of the signal propa-

Algorithm 5 Self Organizing Map

function TRAINSOM($\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\Psi(\cdot)$)
  while $\neg$ stopping **do**
    $\psi \longleftarrow$ DrawSamples$(1, \Psi)$
    $v^* \longleftarrow \text{argmin}_{v \in \mathcal{V}}$ (InformationDist(ψ, v))
    for $v \in \mathcal{V}$ **do**
      $\bar{v} \longleftarrow (v - \phi) \times \lambda \times$ Neighborhood$(v, v^*, \mathcal{G})$
      $v \longleftarrow v + \bar{v}$
  return $\mathcal{G}$

gation along the graph. We set γ to 5 such that approximately 50% of error is propagated to the immediate neighbors of v^* and 3% of the error signal is propagated to vertices 2 edges away.

This process of sampling and moving vertices is repeated until a stopping condition is met. Here the stopping condition is given by

$$\sum_{i=0}^{|\mathcal{V}|} ||\bar{v}_i|| \leq T,$$

where T is a small threshold number, in our case $T = 0.1$. To facilitate convergence, a decreasing discount factor, λ, is used to slowly reduce the magnitude of perturbations to each vertex during a training epoch. This training process is outlined in Algorithm 5.

As previously mentioned, there is no provision in the training of an SOM to allow the topology of $\mathcal{G}$ to change over the course of training. We address this limitation in our **Filtration** function that determines which edges in $\mathcal{E}$ to keep as a part of the graph, and which edges to prune away. We want to remove edges that traverse prominent gaps in the information function, i.e. large, low-information areas, and keep edges in high-information regions. We begin by asserting that our graph forms a simplicial complex,

where the vertices in the graph are 0-simplices, the edges in the graph are 1-simplices, and the triangles bounded by cyclic trios of edges are 2-simplices [16]. This assertion is valid, since the edges of our graph are constructed using a Delaunay Triangulation of the vertices. With a simplical complex, we can easily construct a persistence diagram using the Gudhi Topology Library [97], charting the lifespan of the 1 and 2-dimensional topological features. The next step is to identify a filtration of the simplicial complex that alters the graph topology around the persistent features of the environment, while ignoring ephemeral features that might arise as artifacts of the triangulation process. Since the ephemeral features greatly outnumber the persistent ones, identifying the persistent features becomes a problem of outlier detection. To determine which features are ephemeral and which are persistent, we fit a Weibull distribution to the first-order persistence values. Weibull distributions are used to model the degradation of systems over time [98], and they have semi-infinite support (i.e. they are supported over the range $[0, +\infty)$. We define features with a persistence value beyond the $\alpha\%$ interval of the fitted Weibull (i.e. the range of the distribution that contains $\alpha\%$ of the total distribution) as persistent, while each feature with a persistence value within the $\alpha\%$ interval is ephemeral. The parameter α is a hand-tuned one, and in practice we found that using a value of 75% resulted in good performance.

This operation results in a diagonal persistence threshold, as seen in Fig. 6.3. However, this threshold cannot be used directly to perform the filtration, since it is a property of the triangulation, not the individual edges. To remove edges from the graph, we require a horizontal filtration threshold. To map the persistence threshold to a corresponding filtration threshold, we compute the set of possible values for the filtration

threshold that maximizes the number of persistent features in existence. Then, from these, we select the value that minimizes the number of ephemeral features that exist simultaneously. Once the filtration set, we remove edges with a value greater than the filtration threshold. This process is shown in Fig. 6.3. Applying the filtration alters topology of the SOM to be closer to that of the underlying function, allowing it to fit the function better during subsequent training.

Once the Topology-Aware SOM is trained, it can be used to enumerate the possible homotopy classes of trajectories. To accomplish this, we use the homotopy augmented graph proposed in [20]. The topological features identified during training are used as 'obstacles' in the creation of this graph. Using the robot's current location as a root, we expand a homotopy augmented graph. To keep the size of the homotopy augmented graph manageable, we utilize a non-looping constraint, preventing the expansion of paths that loop more than once around any given obstacle. We also prevent the expansion of any vertex beyond the robot's movement budget, instead adding those vertices to a boundary set. With the homotopy augmented graph, we determine the set of homotopy classes that contain trajectories of interest by applying a quotient map to the unexpanded neighbors of the boundary vertices, mapping them all to a single point. We then determine all homotopy classes between the root point and the quotient point. For each of these homotopy classes, we select its representative path: the path in the homotopy class that maximizes the objective function, $I(\cdot)$.

6.2.2 Stochastic Gradient Ascent

Once a representative path from each of the homotopy classes has been identified, we can then proceed to refine the representative paths using an optimization algorithm. To improve performance, we examined several different heuristics for choosing the order in which to perform optimization on the representative paths. Experimentally, we found that the best predictor for the quality of the optimized path was the quality of the unoptimized path. Other metrics that we considered were the average path quality within each homotopy class as well as the number of trajectories in each homotopy class. However, we found that the average path quality had a weaker correlation than best path quality, and that the number of paths within a homotopy class was uncorrelated with the quality of the best optimized path.

We use Stochastic Gradient Ascent (SGA) algorithm as the local optimization function, since the gradient of the information gathering objective function is difficult to calculate analytically due to the path dependence of the reward [15]. At a high level, SGA operates by estimating the gradient by sampling perturbations and recombining them using a weighting based upon the objective function. Psuedocode for the SGA algorithm is presented in Algorithm 6.

SGA requires an initial path, $\mathcal{P}$, and an information objective function $I(\cdot)$, which computes the path dependent reward for executing the $\mathcal{P}$ in the environment. Then SGA iterates through each of the waypoints in $\mathcal{P}$, and for each $x_i \in \mathcal{P}$ a set of K perturbations is generated. Each perturbation is generated by drawing from a distribution $\mathcal{D}$. This distribution, $\mathcal{D}$, can take on many different forms but is typically a zero-mean normal

distribution. In this work we define $\mathcal{D}$ as a multivariate normal distribution:

$$\mathcal{D} = \mathcal{N}(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_x & 0 \\ 0 & \sigma_y \end{bmatrix})$$

with zero mean and covariance matrix defined by σ_x and σ_y, which are the variation in the x and y directions respectively. Note that here we have assumed the perturbations to be independent, but this assumption is not required for our formulation, and $\mathcal{D}$ can be any covariance matrix. On each iteration through $\mathcal{P}$, we consider the vertices in a random order to avoid undesirable effects of a particular ordering of the path.

After the set of perturbations, $\boldsymbol{\epsilon}$, is generated, each of these perturbations needs to be scored using the information function, $I(\cdot)$. Each of the perturbations, ϵ_i in $\boldsymbol{\epsilon}$ is independently applied to $\mathcal{P}$ at the given index to generate perturbed path $\hat{\mathcal{P}}_k$. Each of these $\hat{\mathcal{P}}_k$ is then scored using $I(\cdot)$ to generate a score vector $\mathbf{s}$. This score vector, $\mathbf{s}$, is then used in conjunction with $\boldsymbol{\epsilon}$ to calculate the update to that waypoint as:

$$\Delta = \frac{1}{K} \sum_{k=1}^{|K|} w_k \times \epsilon_k,$$

where

$$w_k = e^{-h\left(\frac{s_k - \min \mathbf{s}}{\max \mathbf{s} - \min \mathbf{s}}\right)},$$

is the weighting factor for perturbation ϵ_k comparing the score for ϵ_k to the maximum and minimum scores calculated and h is a weighting factor set to 1 in this work. As in the SOM training algorithm, a discount factor, λ, is used to facilitate convergence.

Algorithm 6 Stochastic Gradient Ascent (SGA)

1: **function** SGA($\mathcal{P}$, $I(\cdot)$)
2: **while** $\neg$ stopping **do**
3: **for** $p \in \mathcal{P}$ **do**
4: $\boldsymbol{\epsilon} \longleftarrow$ genPertubations$(\mathcal{D}, K)$
5: $\mathbf{s} \longleftarrow$ getScores$(\mathcal{P}, \boldsymbol{\epsilon}, p, I(\cdot))$
6: $\Delta \longleftarrow$ calcGrad$(\mathbf{s}, \boldsymbol{\epsilon})$
7: $p \longleftarrow p + \Delta \times \lambda$
8: **return** $\mathcal{P}$

6.2.3 Analysis

SGA is guaranteed to almost surely converge to a local maxima [99] given a large number of samples. Our method seeks to improve the likelihood of SGA converging to the global maxima instead of being trapped in a local maxima by partitioning the space of paths into sets of paths with higher local convexity. Since the globally optimal path is guaranteed to lie in one of the enumerated homotopy classes, by sequentially applying optimization within each homotopy class, we hypothesize that our algorithm is more likely to find the globally optimal path than blindly performing an equivalent number of random restarts. In Chapter 6.3, we confirm this hypothesis empirically through comparisons with an SGA variant that is randomly initialized without topological information.

6.2.4 Extension to Multirobot Planning

Topologically distinct trajectories offer an elegant way to distribute different members of a multirobot team to explore an environment by assigning different robots to different homotopy classes, as described in [87]. By applying these ideas to our TA-

SOM algorithm, we extend the algorithm developed previously in this section to plan for multiple robots.

The basic premise for the multirobot informative path planning is the same as the single robot case. We use the TA-SOM algorithm defined in Algorithm 4 to identify the positions of the persistent topological features within the environment. Then, instead of identifying the most promising homotopy class for a single robot, in the multirobot case we are interested in finding the most promising *combination* of homotopy classes for the robot team. Thus, once the TA-SOM has been trained, for each member of the robot team we use a Homotopy Augmented Graph [20] to identify all possible homotopy classes of trajectories. Then, we produce a representative trajectory for each of these homotopy classes. The result of this process is a set of topologically distinct representative trajectories for each member of the robot team. At this point, any number of multirobot planning paradigms, such as sequential allocation, auction-based methods, or joint optimization could be employed to merge each of these single-robot plans into a collective multirobot plan [100]. In this work, we chose to simply use joint optimization, since the primary focus of this contribution is information gathering with topological features, not multirobot coordination. To perform our joint optimization, we evaluate each combination of the five most promising representative trajectories for each robot and select the five best joint plans. While the time required for this step does scale exponentially with the number of robots, we found that for small numbers of robots ($N \leq 5$), this step did not take more than a few seconds. We then use the SGA optimizer described in Chapter 6.2.2 to optimize the joint plans. In the single-robot case, while generating perturbations (Algorithm 6, Line 4), we perturb each waypoint in $\mathcal{P}$ in a random order.

To avoid biases in the multirobot optimization, we not only randomize the order of the waypoints within a path, but also the order of all waypoints across all robots' paths. Once the optimization has converged for each of the five sets of plans proposed to it, we select the best-scoring set for execution.

6.3 Results

To demonstrate the benefits of considering topological features while solving the IPPP, we performed several experiments both in simulation, using real-world datasets, as well as in hardware with an autonomous boat on a local lake. The first set of experiments we perform demonstrates the ability of our proposed methods to accurately capture the salient topological features of the information field. Then, we evaluate the performance of our methods on the IPPP.

6.3.1 Evaluation Datasets

Our primary simulated dataset consists of 20 worlds built using real-world data taken from the Regional Ocean Modelling System (ROMS) [79]. [1] Each ROMS world is created from a 35 km by 35 km section in the center of Monterey Bay, California and is resolved at a grid resolution of 700 m. Each of the twenty worlds is at a randomly chosen time throughout 2017. The information function for these worlds was defined as the magnitude of the surface salinity gradient, a key identification marker for the

[1] The Monterey Bay ROMS model output is provided by the Cooperative Ocean Prediction System (COPS), and is available through their website at http://west.rssoffice.com/ca_roms_nowcast_300m.

localization of upwelling fronts.

In addition, we use two other datasets to produce worlds for illustrative purposes. The Gyre world, shown in Fig. 6.5b, is a hand-constructed environment that contain a quadruple-gyre system, similar to the worlds used in [101] for planning in flows. The Gyre world is the same size as the random and ROMS worlds, 35 km by 35 km at 700m resolution. The information function in this world is defined as the magnitude of the current flow. Since this world was hand-constructed from well-defined topological features, its topology is known *a priori*, and therefore can be used to perform quantitative evaluations. Finally, we use a dataset of bio-acoustic data collected by Slocum Gliders in 2016 in Monterey Bay, California as a real-world example of phenomena that exhibit hotspot tendencies [102]. This dataset can be seen in Fig. 5.1a.

6.3.2 Topological Feature Detection

To evaluate the ability of our TA-SOM to learn the topological features of an environment, we trained a TA-SOM twenty times using random initialization on the gyre world. Since the gyre world has a well-defined topology, we can compare the number of topological features identified by the TA-SOM with the true number. We evaluated the performance of the TA-SOM for zero to five training epochs and using 100, 200, 300, and 500 vertices. The results for these experiments can be seen in Fig 6.6. At 100 vertices the TA-SOM struggles to consistently find all of the features present in the environment. In the remainder, the TA-SOM is able to smoothly converge to the correct number of features. Additionally, the amount of time required to train each of these

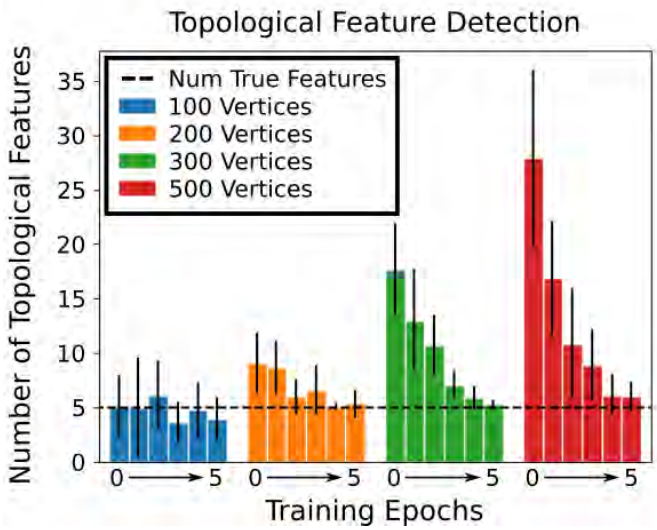


Figure 6.6: Number of topological features found by the TA-SOM in the gyre world for different numbers of vertices across up to five training epochs. The black dashed line at five is the true number of topological features in this environment.

maps is shown in Fig 6.7. As expected, as the number of vertices increases the amount of time required to train the TA-SOM increases. We also note that with no training, our TA-SOM is simply a randomly constructed PRM, and it is equivalent to the simplicial complexes used in [9]. These results clearly show that by using the training process of a SOM to refine the simplical complex graph, we are able to improve the performance of persistence-based simplical complex feature detection. Based on these results, we chose to use 200 vertices and three training epochs for a balance of quality-of-fit and computation time.

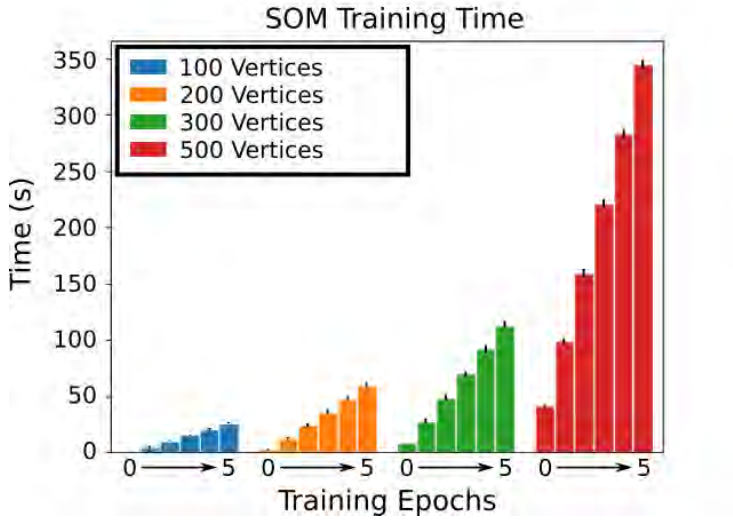


Figure 6.7: Amount of time to train the TA-SOM for different numbers of vertices and training epochs in the gyre world.

6.3.3 Single Robot Information Gathering

In our simulated experiments, we compared the performance of six different information gathering algorithms outlined below:

- **HHIG** - Our Hierarchical Hotspot Information Gathering Planner described in Chapter 5.
- **TA-SOM** - Our Topology-Aware Self Organizing Map planner described in Chapter 6.2, using 3 training epochs and 200 vertices. The 5 most promising homotopy classes identified by the TA-SOM are optimized using SGA.
- **Path Persistence** - A modification of our the TA-SOM method. Instead of using a TA-SOM to identify topological features in the environment, this method directly

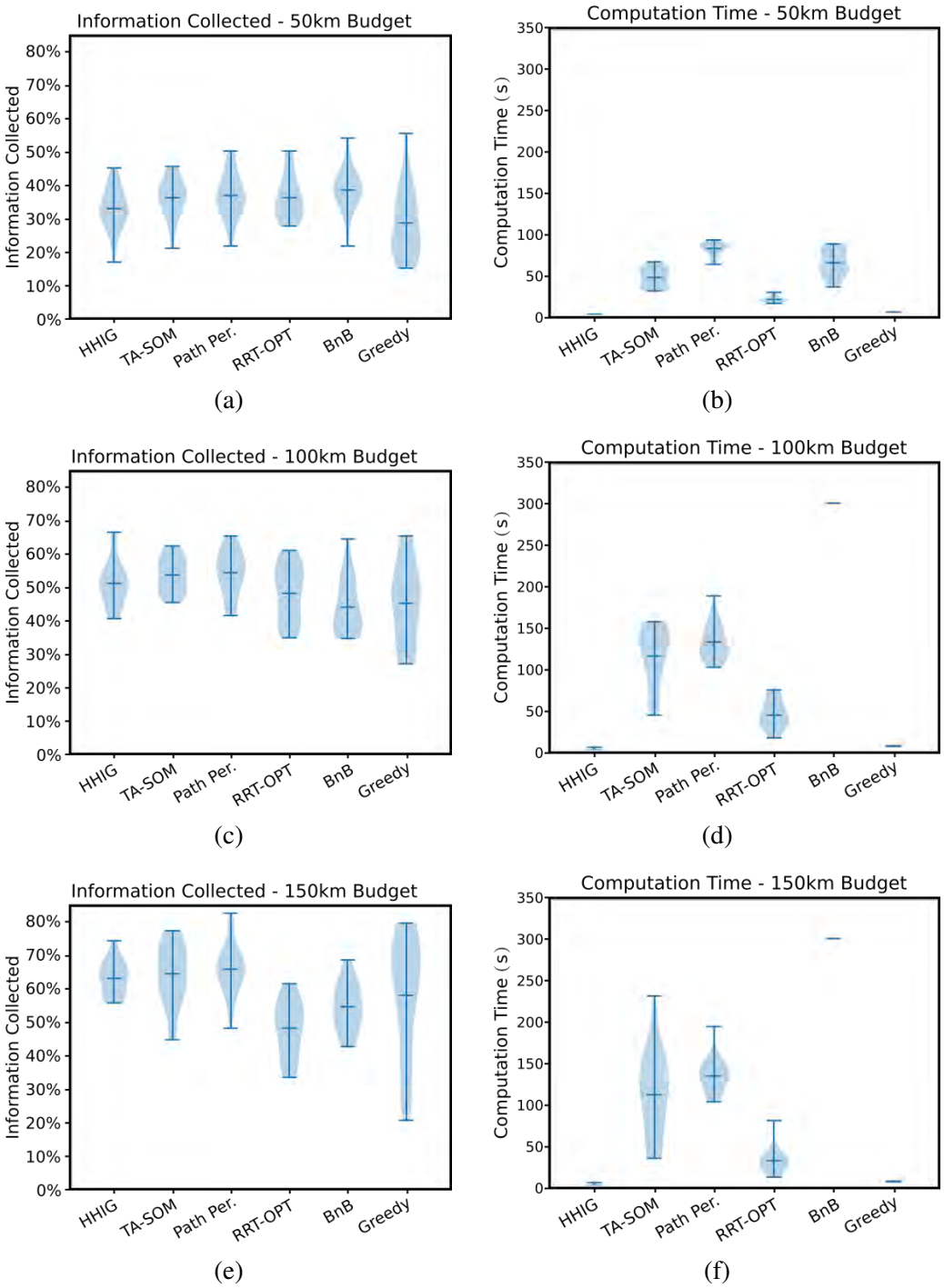


Figure 6.8: Violin plots showing the percentage of information collected by a single robot (a,c,e) and computation time required for planning (b,d,f) for mission budgets of 35 km, 70 km, and 105 km.

uses the persistence of homology classes of paths across multiple thresholds to identify persistent features [103]. Representative trajectories from the five most persistent homotopy classes are then optimized using our SGA framework.

- **RRT-OPT** - A method that uses a Rapidly exploring Random Tree (RRT) to quickly build a large number of paths through the environment. Then, the top five scoring paths to leaf nodes are optimized by SGA. The RRT provides a set of ranked initializations for the SGA framework that do not use topological information. RRT-OPT expands a RRT with 200 vertices.

- **BnB** - The Branch and Bound informative path planner [6]. Planning on a 700 m resolution grid proved computationally intractable for BnB, so the results reported here are planned over a grid with resolution of 3.5 km.

- **Greedy** - The myopic greedy planner.

We evaluated each algorithm at using three different budgets: 35 km, 70 km, and 105 km. Each robot has a sensor radius of 3.5 km, and is evaluated on the total amount of the information within the environment the robot observed. In planning for marine autonomy, particularly for underwater vehicles, minimizing time spent on the surface not only increases the amount of time the robot is conducting its sampling mission, but also minimizes the risk to the robot. To incorporate this constraint into our planning, we enforced an upper limit of 300 seconds of planning time. Since the TA-SOM, Path Persistence, RRT-OPT, and BnB methods are anytime algorithms, if the time limit is reached, the planner uses the best path produced. Each of the planners that incorporate our SGA optimizer used 25 optimization iterations to refine a trajectory to its final form.

The results from these trials are shown in Fig 6.8. The first trend that we notice is, predictably, as the budget increases from 35 km to 105 km, the average score attained by each robot as well as the computation time required to produce each path increases. Secondly, the three topological methods, HHIG, TA-SOM, and Path Persistence all maintain competitive performance with each other, and outperform the non-topological methods at the higher budgets. At lower budgets, (i.e. with shorter paths), in a given environment there are fewer topological decisions to make. As a result, the impact that considering topological features can have relative to the non-topological methods is reduced, making their performance closer to the non-topological methods. The opposite is also true. Longer planning budgets increase the number of unique topological classes in an environment. As a result, reasoning over the space of possible classes is more informative, and therefore more valuable, leading to increased performance of the topological methods, as is evident in Fig. 6.8e. Due to the size of the environment, Branch and Bound struggles, particularly with higher planning budgets, where it failed to converge to its graph-optimal path within the 300 second planning time. It is worth noting that Branch and Bound uses a graph resolution five times coarser than that used by the HHIG or Greedy algorithms. At an equal resolution, the size of the decision problem is intractable for Branch and Bound, and it failed to converge for any of the three budgets tested.

The three topological methods, HHIG, TA-SOM, and Path Persistence, all have competitive performance across all planning budgets. However, they are differentiated by the amount of computation required to achieve their levels of performance. The HHIG algorithm requires significantly less computation than TA-SOM or Path Persistence. This difference can be attributed to the fact that the HHIG algorithm uses a different topo-

logical planning paradigm than the other two topological methods. HHIG constructs a sparse topological graph that captures the adjacency of different information hotspots. The key decision making and scheduling happens on this graph. In practice, we found that the size and degree of these graphs tend to be small, usually with 4 or 5 hotspots and a degree of about 3. Even though HHIG must search through all possible schedules on this graph, the small size makes this search manageable. In contrast, both the TA-SOM and Path Persistence algorithms rely on homotopy augmented graphs [20] to enumerate the topological trajectory classes created by the features in the environment. In the TA-SOM algorithm, we create a single homotopy augmented graph after the TA-SOM has been trained. In contrast, the Path Persistence algorithm, since it is using the persistence of homotopy classes across multiple threshold levels, requires a homotopy augmented graph for each threshold of the information function. Consequently constructing a homotopy augmented graph must be repeated a number of times determined by the number of thresholds used to compute path feature persistence. Increasing the resolution of these thresholds provides a more accurate measure of the path persistence at the cost of additional computation time.

6.3.4 Multi-robot Information Gathering

The final set of experiments examined the ability of the TA-SOM and Path Persistence methods to scale in multirobot information gathering. Since the core of both methods utilizes homotopy information to divide the space of possible paths into topologically distinct trajectory classes, they each have a natural extension to the multirobot

planning problem. We compare these two methods with a baseline Greedy algorithm. We do not extend our HHIG planner to the multirobot case, since the inclusion of multiple robots into the topological graph scheduling problem is less of a straightforward extension, and beyond the scope of the work presented here. The results from these experiments showing both the information collected and the computation time is shown in Fig. 6.9 and Fig. 6.10, respectively.

In terms of performance in the information gathering task, the multirobot results mirror the single-robot results. Generally, both topological methods outperform the non-topological baseline. However, there is one exception. At a team size of five robots, the Path Persistence algorithm achieved a score of zero. The reason for this outlier can be seen in the computation time results. As the number of robots increases, the computation times for all three methods increases. While our TA-SOM method manages to stay beneath the 300 second time limit for all team sizes, the Path Persistence quickly scales to the maximum computation time. Both the TA-SOM and Path Persistence methods are comprised of two main steps: first a processing step to identify unique homotopy classes for each robot, and second an optimization step where the best homotopy classes are jointly optimized. For the TA-SOM planner, the computational expense of the first step is largely independent of the number of robots since all robots can use the same trained TA-SOM. As a result, Homotopy information only needs to be computed once for each robot. In contrast, the Path Persistence method must compute a homotopy augmented graph multiple times for each robot. The effects of this difference becomes apparent when the robot team size grows to the point where the Path Persistence algorithm is unable to complete the first step in the allotted time. It is unable to produce any paths,

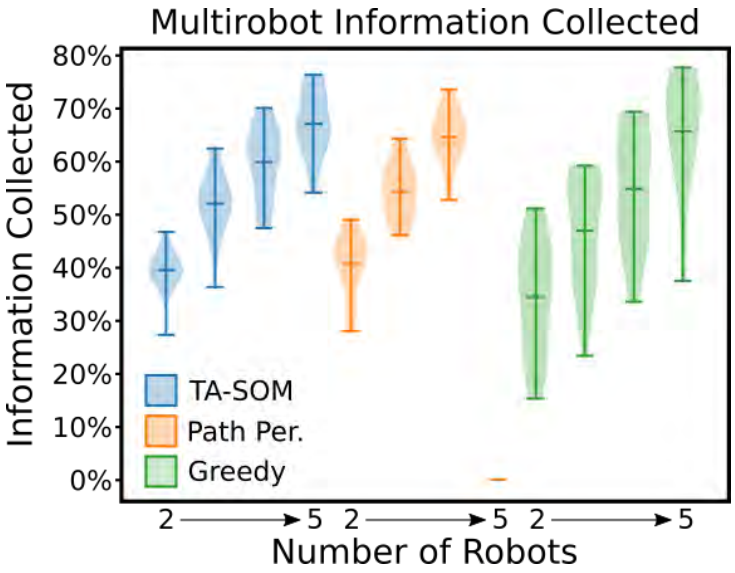


Figure 6.9: Multirobot Score. With a team size of 5 robots, the Path Persistence method failed to produce a plan in all 20 of the trials, resulting in an average score of 0.

and therefore achieves a score of zero.

6.3.5 Field Testing

To validate the performance of our topological planning algorithms, we deployed them on a Platypus Lutra autonomous boat [3], shown in Fig. 6.11, at Ireland Lane Pond, near Corvallis, Oregon. We tested four different algorithms, our HHIG algorithm, our TA-SOM algorithm, and Greedy, which we use as a baseline. We conducted three trials for each algorithm, one from each of three different starting locations: the first near our deployment point on the north shore of the pond, the second near a backup deployment point on the southeastern shore, and the third in the center of our deployment region. For the information function, we used a map of the magnitude bottom gradient,

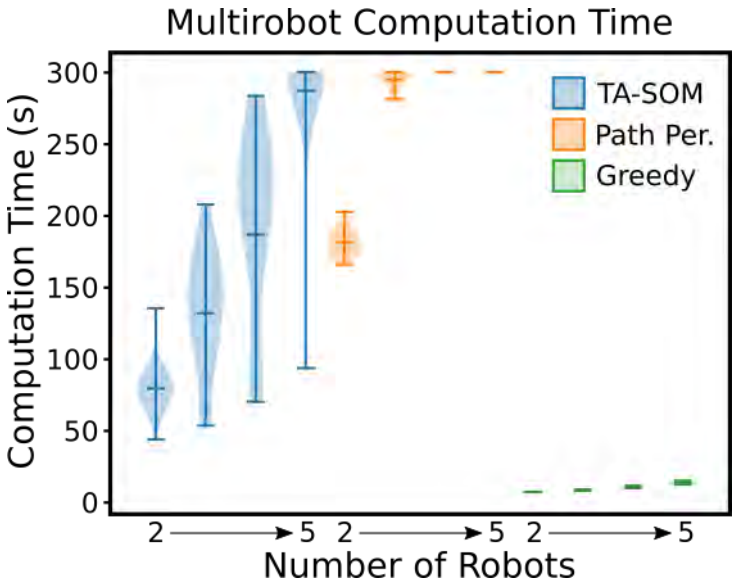


Figure 6.10: Multirobot Computation Time. As the number of robots increases, TA-SOM scales to the maximum computation time more efficiently.

taken within a bounded region of the lake and normalized into the range [0,1]. This map was produced by completing a dense survey of the pond with the Lutra's sonar and combining the observations into a single map using a Gaussian Process with an RBF kernel. All nine of the plans were planned offline, then executed using the Lutra. The results from all nine paths are shown in Table 6.1, and the best-performing trajectories, those from the northern deployment location are shown in Fig. 6.14. The results from these trials mirror our simulated results, with the three topological methods outperforming the greedy baseline, both in actual information collected and in the expected reward from the planned paths.

The topological representations built by our algorithms are shown in Fig. 6.12 and Fig. 6.13. Qualitatively, both representations capture the underlying structure of the in-

Table 6.1: Percentage of total information collected by the robot at each of three starting locations. Expected information based on planned path is shown in parentheses.

	Central	Southeast	North
HHIG	19.88 (20.66)	18.56 (17.14)	20.21 (19.36)
TA-SOM	**22.56** (22.96)	19.62 (20.40)	**22.51** (21.28)
Path Per.	22.23 **(23.42)**	**21.85 (21.45)**	21.23 **(21.82)**
Greedy	16.97 (18.28)	15.43 (15.42)	17.63 (15.99)

formation field. In the hotspot map, the contours of the hotspots follow those of the darker high-information regions, while in the trained TA-SOM, our algorithm has discovered information voids in the low-information region in the southern half of the experiment region. The effects of these representations can be seen in their corresponding paths in Fig. 6.14. The behaviors shown there are representative of the behavior of all three algorithms at the other two starting locations. The path produced by the TA-SOM planner wraps around the western side of the void, skirting it, while reaching the high information region in the south. Meanwhile, the path produced by the hotspot algorithm diverts to the high information region in the west, before continuing to the area in the south. In contrast to the topological methods, the greedy algorithm fails to realize the existence of the information to the west, and becomes stuck in the local maxima at the south.

6.4 Conclusion

In this chapter we presented a method for using a Topology-Aware Self Organizing Map to allow identify prominent features in an environment. Using these features, we

Figure 6.11: Platypus Lutra Autonomous Boat with Lowrance depth sonar used in field trial experiments [3]

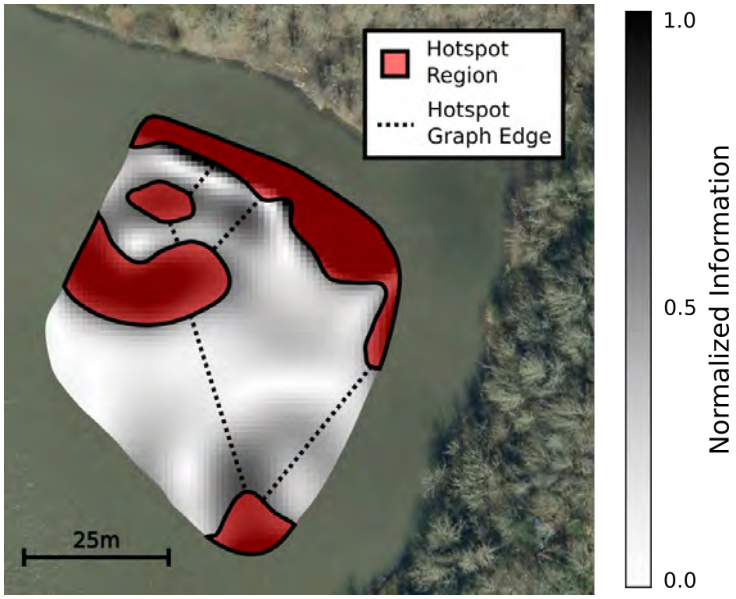


Figure 6.12: Hotspots on Ireland Lane Pond, Corvallis, Oregon

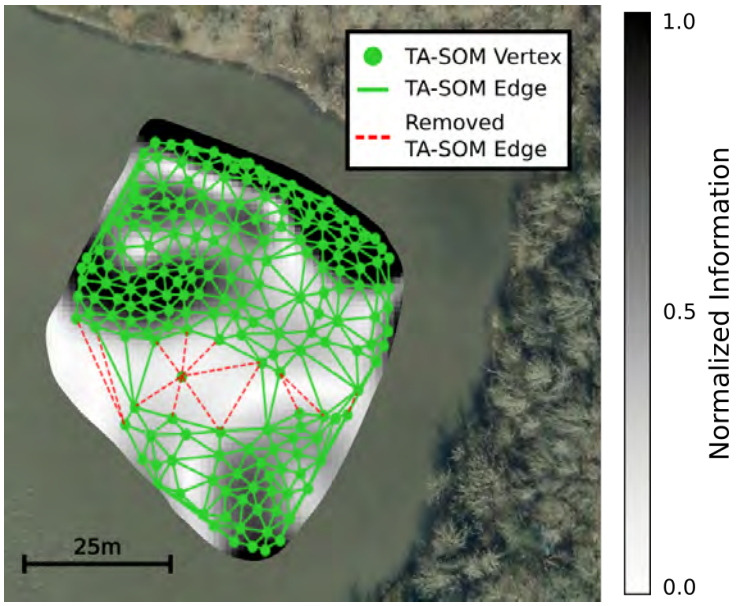


Figure 6.13: Trained TA-SOM on Ireland Lane Pond, Corvallis, Oregon

can identify a set of topologically distinct trajectory classes that, in turn, can be utilized to generate a set of reference trajectories that span the local maxima of the space of possible paths in the information gathering task.These trajectories enable improved performance from a local optimizer, Stochastic Gradient Ascent, allowing it to more easily find paths closer to a global optimum.

In our experiments on ROMS model output we were able to show that by incorporating topological information into an informative path planner, we were able to improve the performance of a robot in the IPPP. Furthermore, we showed that this additional benefit does not require a significant computational cost, as our HHIG planner is able to realize these benefits while maintaining a computation time that is comparable to the naïve greedy algorithm.

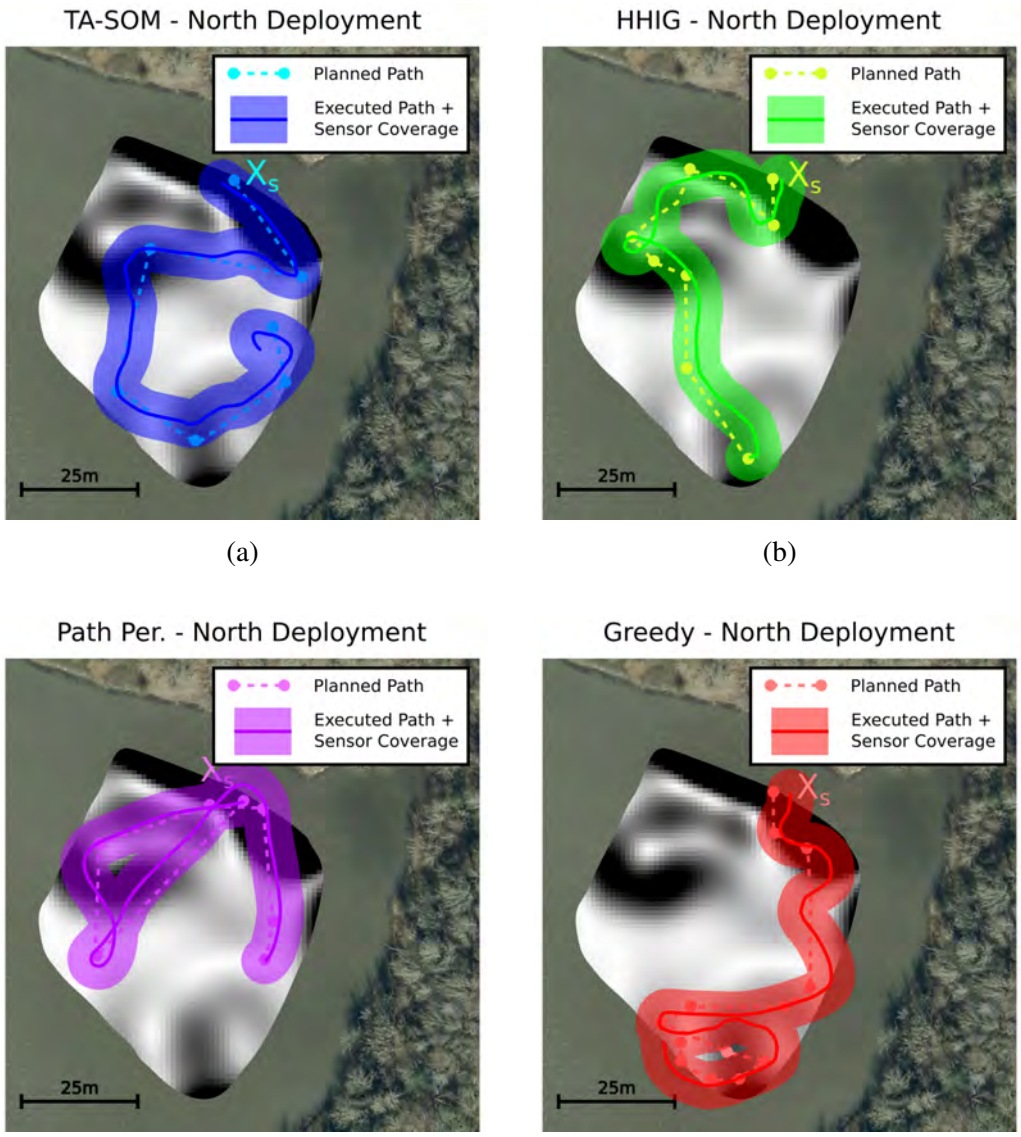


Figure 6.14: Sample Paths from the northern starting location. The shaded regions show the area covered by the Lutra's 5m sensing radius as it traveled along its path. While the Greedy path (c) gets stuck in the southern corner after travelling down the east side of the experiment region, the TA-SOM (a) and HHIG (b) planners both visit the high-information area in the west.

Chapter 7: Conclusion

In this thesis, we have provided four contributions that taken together provide a framework for enabling topologically guided robotic information gathering. First, we demonstrated that explicit topological constraints can be be incorporated into a robot mission planning algorithm by developing a method that enables a tethered vehicle to avoid becoming entangled by its tether during an infrastructure inspection mission. Second, we relaxed the assumption that the robot is given an explicit map of the features, and instead examine how descriptions of the topological features and desirable robot behavior can be incorporated into robotic planning. We developed a multirobot path planning framework uses this domain knowledge in the construction of an information objective function and a topological environment model. We showed that planning using a topological environment model improves the performance of the robot team in a front-mapping task. Additionally, we showed that planning using these models enables autonomous control of the robot team to match or exceed the performance of human experts on a front-sampling task.

While these first two contributions expand the abilities of robots to reason about the topology of their environment while planning information gathering missions, we found existing methods for incorporating topology in planning were lacking in problems where the topological features are not given from external information. To address this issue, we developed two techniques that use the characteristics of the information

gathering problem itself to identify topological features in a the information objective function and then exploit them to improve the ability of a robot to plan informative paths. The first of these methods, our Hierarchical Hotspot Information Gathering planner, partitions the information field into a set of discrete information hotspots that form a topological graph. By first planning the information gathering mission on this graph, we can rapidly incorporate global context about the structure of the information field into the planning process. The second method, Topology-Aware Self Organizing Maps, forms a topological representation of the space of possible paths through the information field. Using this representation, we can identify promising local maxima in the robot's path space and exploit them using a Stochastic Gradient Ascent trajectory optimizer to quickly find the global maximum.

Across our four contributions, we demonstrated the benefits of leveraging topological models for robotic information gathering and inspection. Since these models are inherently global, they enable robots to reason over their whole path. As a result, robots can consider new constraints on their paths that cannot be modeled using purely metric representations. Furthermore, the additional global context provided by the topological models enables robots to reason at a high level about their environments, which helps them avoid becoming trapped in local maxima in the information objective function. In our experiments, we showed that even when robots are not provided with external definitions of topological features, they can use characteristics of the information gathering problem itself to build topological representations of the information objective function. These representations improve the performance of the robots on the information gathering task over state-of-the-art methods that do not incorporate topology.

7.1 Future Research Directions

One of the main limitations of topological techniques is that they require a significant amount of prior information about the environment in order to build a meaningful topological representation of it. In this work, we either assumed that the robot was provided with a map of the topological features in an environment, descriptions of topological features from human experts, or a map of the information field itself. In some applications, such as those where the robot can benefit from human experts, satellite data, or prior surveys, these assumptions may be valid. However, this limitation diminishes the applicability of topological methods in unknown environments.

7.1.1 Topology of Partially-Known Environments

In the near term, we would like to reexamine this assumption and investigate ways that topological features could be identified in real-time so they can be exploited in unknown and partially known environments. A related avenue for future research is to apply topological techniques in time-varying environments. Since topological representations contain unique information about the structure of the environment that is independent of the layout in metric space, plans made in a topological space may remain useful even if the metric environment changes. Furthermore, changes in the topological structure of an environment may provide insights about when a robot should replan.

7.1.2 Explainability and Introspection via Topology

Taking a broader view, the contributions of this thesis have enabled robots to reason about their environments at a high level during information gathering tasks. By understanding the elements of the Informative Path Planning Problem not as a metric motion planning problem, but instead as a high-level decision making problem, we have laid the groundwork for future robots to introspect about and communicate their own decision making. When working in teams, this model will allow robots to more concisely communicate their plans, goals, and reasoning with each other, improving their ability to work together and coordinate as members of a distributed multirobot team. Furthermore, increased transparency and explainability in decision making could open up new ways for humans collaborate with robots, allowing robots to benefit from decades of human expertise and experience as well as human creativity and insight.

Bibliography

[1] L. Vieira, *Coffee Mug Torus Homeomorphism*. Wikipedia, 2007. Accessed on: Mar. 2016. [Online]. Available: en.wikipedia.org/wiki/Homeomorphism#/media/File:Mug_and_Torus_morph.gif.

[2] G. Best, *Planning Algorithms for Multi-Robot Active Perception*. PhD thesis, University of Sydney, 2019.

[3] Platypus, LLC, *The Lutra Prop*, 2014. Accessed on Dec. 2020. [Online]. Available: senseplatypus.com/lutra-prop.

[4] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2015.

[5] P. L. Shick, *Topology: point-set and geometric*, vol. 83. John Wiley & Sons, 2011.

[6] J. Binney and G. S. Sukhatme, "Branch and bound for informative path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota*, pp. 2147–2154, 2012.

[7] E. A. Topp and H. I. Christensen, "Topological modelling for human augmented mapping," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China*, pp. 2257–2263, 2006.

[8] S. Bhattacharya, V. Kumar, and M. Likhachev, "Search-based path planning with homotopy class constraints," in *Proceedings of the AAAI Conference on Artificial Intelligence, Atlanta, Georgia*, pp. 1230–1237, 2010.

[9] F. T. Pokorny, M. Hawasly, and S. Ramamoorthy, "Topological trajectory classification with filtrations of simplicial complexes and persistent homology," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 204–223, 2016.

[10] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, 1998.

[11] R. Sandström, D. Uwacu, J. Denny, and N. M. Amato, "Topology-guided roadmap construction with dynamic region sampling," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6161–6168, 2020.

[12] T. P. McNamara, "Mental representations of spatial relations," *Cognitive psychology*, vol. 18, no. 1, pp. 87–121, 1986.

[13] M. Fox, D. Long, and D. Magazzeni, "Explainable planning," *arXiv preprint arXiv:1709.10256*, 2017.

[14] J. A. Sethian, *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, vol. 3. Cambridge University Press, 1999.

[15] D. Jones, M. J. Kuhlman, D. A. Sofge, S. K. Gupta, and G. A. Hollinger, "Stochastic optimization for autonomous vehicles with limited control authority," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, Spain*, pp. 2395–2401, 2018.

[16] W. F. Basener, *Topology and its Applications*. John Wiley & Sons Inc., 2006.

[17] J. H. Hubbard, *Differential Equations: A Dynamical Systems Approach: Higher-Dimensional Systems*. Texts in applied mathematics ; 18, New York, NY: Springer New York, 1995.

[18] A. Kitanov and V. Indelman, "Topological multi-robot belief space planning in unknown environments," in *Proceedings of the IEEE International Conference on Robotics and Automation, Brisbane, Australia*, pp. 5726–5732, 2018.

[19] C. Galindo, J.-A. Fernández-Madrigal, J. González, and A. Saffiotti, "Robot task planning using semantic maps," *Robotics and autonomous systems*, vol. 56, no. 11, pp. 955–966, 2008.

[20] S. Bhattacharya, M. Likhachev, and V. Kumar, "Topological constraints in search-based robot path planning," *Autonomous Robots*, vol. 33, no. 3, pp. 273–290, 2012.

[21] R. K. Ramachandran, S. Wilson, and S. Berman, "A probabilistic approach to automated construction of topological maps using a stochastic robotic swarm," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 616–623, 2017.

[22] P. G. Xavier, "Shortest path planning for a tethered robot or an anchored cable," in *Proceedings of the IEEE International Conference on Robotics and Automation, Detroit, Michigan*, pp. 1011–1017, 1999.

[23] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[24] S. Bhattacharya, S. Kim, H. Heidarsson, G. S. Sukhatme, and V. Kumar, "A topological approach to using cables to separate and manipulate sets of objects," *The International Journal of Robotics Research*, vol. 34, no. 6, pp. 799–815, 2015.

[25] S. Kim, S. Bhattacharya, and V. Kumar, "Path planning for a tethered mobile robot," in *Proceedings of the IEEE International Conference on Robotics and Automation, Hong Kong, China*, pp. 1132–1139, 2014.

[26] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägele, "Room segmentation: Survey, implementation, and analysis," in *Proceedings of the IEEE International Conference on Robotics and Automation, Stockholm, Sweden*, pp. 1019–1026, 2016.

[27] E. Brunskill, T. Kollar, and N. Roy, "Topological mapping using spectral clustering and classification," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, California*, pp. 3491–3496, 2007.

[28] Z. Zivkovic, B. Bakker, and B. Krose, "Hierarchical map building and planning based on graph partitioning," in *Proceedings of the IEEE International Conference on Robotics and Automation, Orlando, Florida*, pp. 803–809, 2006.

[29] S. Friedman, H. Pasula, and D. Fox, "Voronoi random fields: Extracting topological structure of indoor environments via place labeling.," in *Proceedings of the International Joint Conference on Artificial Intelligence, Hyderabad, India*, vol. 7, pp. 2109–2114, 2007.

[30] F. Aurenhammer, "Voronoi diagrams - a survey of a fundamental geometric data structure," *ACM Computing Surveys (CSUR)*, vol. 23, no. 3, pp. 345–405, 1991.

[31] I. Kostavelis, K. Charalampous, A. Gasteratos, and J. K. Tsotsos, "Robot navigation via spatial and temporal coherent semantic maps," *Engineering Applications of Artificial Intelligence*, vol. 48, pp. 173–187, 2016.

[32] S. Oßwald, M. Bennewitz, W. Burgard, and C. Stachniss, "Speeding-up robot exploration by exploiting background information," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 716–723, 2016.

[33] M. Luperto and F. Amigoni, "Predicting the global structure of indoor environments: A constructive machine learning approach," *Autonomous Robots*, vol. 43, no. 4, pp. 813–835, 2019.

[34] A. Huyer, "Coastal upwelling in the california current system," *Progress in oceanography*, vol. 12, no. 3, pp. 259–284, 1983.

[35] M. Michini, M. A. Hsieh, E. Forgoston, and I. B. Schwartz, "Robotic tracking of coherent structures in flows," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 593–603, 2014.

[36] G. Ji, H.-W. Shen, and R. Wenger, "Volume tracking using higher dimensional isosurfacing," in *Proceedings of the IEEE Computer Society Visualization Conference, Seattle, Washington*, pp. 28–36, 2003.

[37] J. Lukasczyk, R. Maciejewski, C. Garth, and H. Hagen, "Understanding hotspots: a topological visual analytics approach," in *Proceedings of the SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, Washington*, pp. 36–46, ACM, 2015.

[38] Y. Zhang, J. G. Bellingham, J. P. Ryan, B. Kieft, and M. J. Stanway, "Autonomous four-dimensional mapping and tracking of a coastal upwelling front by an autonomous underwater vehicle," *Journal of Field Robotics*, vol. 33, no. 1, pp. 67–81, 2016.

[39] J. Yu, M. Schwager, and D. Rus, "Correlated orienteering problem and its application to persistent monitoring tasks," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1106–1118, 2016.

[40] B. N. Seegers, J. M. Birch, R. Marin III, C. A. Scholin, D. A. Caron, E. L. Seubert, M. D. A. Howard, G. L. Robertson, and B. H. Jones, "Subsurface seeding of surface harmful algal blooms observed through the integration of autonomous gliders, moored environmental sample processors, and satellite remote sensing in Southern California," *Limnology and Oceanography*, vol. 60, no. 3, pp. 754–764, 2015.

[41] M. Hutchinson, H. Oh, and W.-H. Chen, "A review of source term estimation methods for atmospheric dispersion events using static or mobile sensors," *Information Fusion*, vol. 36, pp. 130–148, 2017.

[42] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1271–1287, 2014.

[43] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[44] A. Krause and C. Guestrin, "Near-optimal observation selection using submodular functions," in *Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, Canada*, vol. 7, pp. 1650–1654, 2007.

[45] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-mcts: Decentralized planning for multi-robot active perception," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 316–337, 2019.

[46] J. L. Nguyen, N. R. J. Lawrance, R. Fitch, and S. Sukkarieh, "Real-time path planning for long-term information gathering with an aerial glider," *Autonomous Robots*, vol. 40, pp. 1017–1039, Aug 2016.

[47] R. Marchant and F. Ramos, "Bayesian optimisation for informative continuous path planning," in *Proceedings of the IEEE International Conference on Robotics and Automation, Hong Kong, China*, pp. 6136–6143, 2014.

[48] S. M. LaValle, *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Citeseer, 1998.

[49] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, pp. 707–755, 2009.

[50] N. K. Yilmaz, C. Evangelinos, P. F. J. Lermusiaux, and N. M. Patrikalakis, "Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming," *IEEE Journal of Oceanic Engineering*, vol. 33, no. 4, pp. 522–537, 2008.

[51] J. Yu, J. Aslam, S. Karaman, and D. Rus, "Anytime planning of optimal schedules for a mobile sensing robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Hamburg, Germany*, pp. 5279–5286, 2015.

[52] A. Singh, A. Krause, and W. J. Kaiser, "Nonmyopic adaptive informative path planning for multiple robots," in *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence, Pasadena, California*, pp. 1843–1850, 2009.

[53] M. Held and R. M. Karp, "The traveling-salesman problem and minimum spanning trees," *Operations Research*, vol. 18, no. 6, pp. 1138–1162, 1970.

[54] N. R. J. Lawrance, R. DeBortoli, D. Jones, S. McCammon, L. Milliken, A. Nicolai, T. Somers, and G. A. Hollinger, "Shared autonomy for low-cost underwater vehicles," *Journal of Field Robotics*, vol. 36, no. 3, pp. 495–516, 2019.

[55] S. McCammon and G. A. Hollinger, "Planning non-entangling paths for tethered underwater robots using simulated annealing," in *Proceedings of the Robot Learning and Planning Workshop at Robotics: Science and Systems, Ann Arbor, Michigan*, pp. 1–4, 2016.

[56] S. McCammon and G. A. Hollinger, "Planning and executing optimal non-entangling paths for tethered underwater vehicles," in *Proceedings of the IEEE International Conference on Robotics and Automation, Singapore*, pp. 3040–3046, 2017.

[57] I. Shnaps and E. Rimon, "Online coverage by a tethered autonomous mobile robot in planar unknown environments," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 966–974, 2014.

[58] S. Burer and A. N. Letchford, "Non-convex mixed-integer nonlinear programming: A survey," *Surveys in Operations Research and Management Science*, vol. 17, no. 2, pp. 97–106, 2012.

[59] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*, pp. 85–103, Springer, 1972.

[60] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *Proceedings of the IEEE American Control Conference, Anchorage, Alaska*, vol. 3, pp. 1936–1941, 2002.

[61] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[62] Gurobi Optimization, Inc., *Gurobi Optimizer Reference Manual*, 2014. Accessed on: Jan. 2016. [Online]. Available: gurobi.com.

[63] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[64] SeaBotix Inc., *vLBV Operator's Manual*, 2015. Accessed on: Dec. 2020. [Online]. Available: teledynemarine.com/vlbv300.

[65] N. R. J. Lawrance, T. Somers, D. Jones, S. McCammon, and G. A. Hollinger, "Ocean deployment and testing of a semi-autonomous underwater vehicle," in *Proceedings of OCEANS MTS/IEEE, Monterey, California*, pp. 1–6, 2016.

[66] O. Schofield *et al.*, "Slocum gliders: Robust and ready," *Journal of Field Robotics*, vol. 24, no. 6, pp. 473–485, 2007.

[67] J. A. MacKinnon *et al.*, "A tale of two spicy seas," *Oceanography*, vol. 29, no. 2, pp. 50–61, 2016.

[68] A. S. Ferreira, M. Costa, F. Py, J. Pinto, M. A. Silva, A. Nimmo-Smith, T. A. Johansen, J. B. de Sousa, and K. Rajan, "Advancing multi-vehicle deployments in oceanographic field experiments," *Autonomous Robots*, vol. 43, no. 6, pp. 1555–1574, 2019.

[69] K.-C. Ma, L. Liu, H. K. Heidarsson, and G. S. Sukhatme, "Data-driven learning and planning for environmental sampling," *Journal of Field Robotics*, vol. 35, no. 5, pp. 643–661, 2018.

[70] R. N. Smith, P. Cooksey, F. Py, G. S. Sukhatme, and K. Rajan, "Adaptive path planning for tracking ocean fronts with an autonomous underwater vehicle," in *Proceedings of the 14th International Symposium on Experimental Robotics, Tokyo, Japan*, pp. 761–775, Springer, 2016.

[71] S. McCammon, G. Marcon dos Santos, M. Frantz, T. P. Welch, G. Best, R. K. Shearman, J. D. Nash, J. A. Barth, J. A. Adams, and G. A. Hollinger, "Ocean front detection and tracking using a team of heterogeneous marine vehicles," *under review at the Journal of Field Robotics*, 2019.

[72] S.-H. Yoo, A. Stuntz, Y. Zhang, R. Rothschild, G. A. Hollinger, and R. N. Smith, "Experimental analysis of receding horizon planning algorithms for marine monitoring," in *Proceedings of the Conference on Field and Service Robotics, Toronto, Canada*, pp. 31–44, Springer, 2016.

[73] M. Watterson and V. Kumar, "Safe receding horizon control for aggressive mav flight with limited range sensing," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3235–3240, 2015.

[74] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning, Canberra, Australia*, pp. 63–71, Springer, 2003.

[75] J. Das, F. Py, T. Maughan, T. O'reilly, M. Messié, J. Ryan, G. S. Sukhatme, and K. Rajan, "Coordinated sampling of dynamic oceanographic features with underwater vehicles and drifters," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 626–646, 2012.

[76] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[77] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[78] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of Monte Carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[79] A. F. Shchepetkin and J. C. McWilliams, "The Regional Oceanic Modeling Mystem (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model," *Ocean modelling*, vol. 9, no. 4, pp. 347–404, 2005.

[80] H. Everett III, "Generalized lagrange multiplier method for solving problems of optimum allocation of resources," *Operations Research*, vol. 11, no. 3, pp. 399–417, 1963.

[81] S. McCammon and G. A. Hollinger, "Topological hotspot identification for informative path planning with a marine robot," in *Proceedings of the IEEE International Conference on Robotics and Automation, Brisbane, Australia*, pp. 4865–4872, 2018.

[82] S. McCammon and G. A. Hollinger, "Topological path planning for autonomous information gathering," *In preparation for Autonomous Robots*, 2020.

[83] C. Petres *et al.*, "Path planning for autonomous underwater vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 331–341, 2007.

[84] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, California*, pp. 4569–4574, 2011.

[85] C. Jones, E. Creed, S. Glenn, J. Kerfoot, J. Kohut, C. Mudgal, and O. Schofield, "Slocum gliders - a component of operational oceanography," in *Proceedings of the International Symposium on Unmanned Untethered Submersible Technology, Lee, New Hampshire*, pp. 21–24, 2005.

[86] S. McCammon, D. Jones, and G. A. Hollinger, "Topology-aware self organizing maps for robotic information gathering," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, Nevada (Virtual)*, pp. 1717–1724, 2020.

[87] S. Kim, S. Bhattacharya, R. Ghrist, and V. Kumar, "Topological exploration of unknown and partially known environments," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan*, pp. 3851–3858, 2013.

[88] A. Zomorodian and G. Carlsson, "Computing persistent homology," *Discrete & Computational Geometry*, vol. 33, no. 2, pp. 249–274, 2005.

[89] U. Bauer and H. Edelsbrunner, "The Morse theory of Čech and Delaunay filtrations," in *Proceedings of the ACM Thirtieth Annual Symposium on Computational Geometry, New York, New York*, pp. 484–490, 2014.

[90] K. Borsuk, "On the imbedding of systems of compacta in simplicial complexes," *Fundamenta Mathematicae*, vol. 35, no. 1, pp. 217–234, 1948.

[91] H. Edelsbrunner and J. Harer, "Persistent homology - a survey," *Contemporary Mathematics*, vol. 453, pp. 257–282, 2008.

[92] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[93] J. Faigl, M. Kulich, V. Vonásek, and L. Přeučil, "An application of the self-organizing map in the non-Euclidean traveling salesman problem," *Neurocomputing*, vol. 74, no. 5, pp. 671–679, 2011.

[94] S. Somhom, A. Modares, and T. Enkawa, "A self-organising model for the travelling salesman problem," *Journal of the Operational Research Society*, vol. 48, no. 9, pp. 919–928, 1997.

[95] J. Faigl, R. Pěnička, and G. Best, "Self-organizing map-based solution for the orienteering problem with neighborhoods," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Budapest, Hungary*, pp. 1315–1321, 2016.

[96] J. Faigl and G. A. Hollinger, "Autonomous data collection using a self-organizing map," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 5, pp. 1703–1715, 2017.

[97] The GUDHI Project, *GUDHI User and Reference Manual*, 2014. Accessed on: Jan. 2020. [Online]. Available: gudhi.inria.fr.

[98] H. Rinne, *The Weibull distribution: a handbook*. CRC press, 2008.

[99] K. C. Kiwiel, "Convergence and efficiency of subgradient methods for quasiconvex minimization," *Mathematical programming*, vol. 90, no. 1, pp. 1–25, 2001.

[100] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International journal of robotics research*, vol. 23, no. 9, pp. 939–954, 2004.

[101] D. Kularatne, S. Bhattacharya, and M. A. Hsieh, "Going with the flow: a graph based approach to optimal path planning in general flows," *Autonomous Robots*, vol. 42, no. 7, pp. 1369–1387, 2018.

[102] K. J. Benoit-Bird, T. P. Welch, C. M. Waluk, J. A. Barth, I. Wangen, P. McGill, C. Okuda, G. A. Hollinger, M. Sato, and S. McCammon, "Equipping an underwater glider with a new echosounder to explore ocean ecosystems," *Limnology and Oceanography: Methods*, vol. 16, no. 11, pp. 734–749, 2018.

[103] S. Bhattacharya, R. Ghrist, and V. Kumar, "Persistent homology for path planning in uncertain environments," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 578–590, 2015.