Autonomous Robots (2021) 45:821–842
https://doi.org/10.1007/s10514-021-10012-x

Topological path planning for autonomous information gathering

Seth McCammon[1,2] · **Geoffrey A. Hollinger**[2]

Received: 9 December 2020 / Accepted: 21 July 2021 / Published online: 7 September 2021

Abstract
In this paper, we present two novel algorithms for information space topological planning that identify topological features in an information field and use them to plan maximally informative paths for a robot in an information gathering task. These features provide a way to rapidly incorporate global context into the informative path planning process by partitioning the state space or the path space of a robot. Our first algorithm, hierarchical hotspot information gathering, uses a topological state space partitioning by constructing a high-level map of information hotspots. We then solve a global scheduling problem over the topological graph, the solution of which is then used for path planning by a set of local greedy coverage planners within each hotspot. Our second algorithm, Topology-Aware Self Organizing Maps, extends the Self Organizing Map algorithm to discover prominent topological features in the information function. These features are used to perform a topological path space decomposition to provide a Stochastic Gradient Ascent optimization algorithm with topologically diverse initialization, improving its performance. In simulated trials and field experiments, we compare the tradeoffs of these two approaches and show that our methods that leverage topological features of the information field consistently perform competitively or better than methods that do not exploit these features, while requiring less computation time.

Keywords Informative path planning · Topological path planning · Field robotics · Environmental monitoring

1 Introduction

In field robotics applications, we often wish to deploy robots as mobile sensor platforms to move through the environment and collect useful observations in monitoring and inspection applications across marine, aerial, and ground-based domains. Planning the path that allows a robot to collect the most useful observations is known as the Informative Path Planning Problem (IPPP). One of the main challenges of the IPPP is that the amount of information collected along a path is dependent on the whole route of the path, not just the final state of the robot. Consequently, there is a link between the distribution of information throughout the environment and the distribution of high-quality trajectories through the space of all possible trajectories. Existing approaches to solving the IPPP do not take full advantage of the information available to the robot provided by this distribution. In this paper, we introduce the concept of Information Space Topological Planning (ISTP), an approach to solving the IPPP that uses topology to describe the structure of the distribution of information in an environment. Topological planning methods enable a robot to simultaneously reason over groups of trajectories that move through the environment relative to a set of topological features. In ISTP, features are drawn from the information space of the robot. By constructing a topological representation using these features, a robot can reason directly about the global structure of the distribution of information throughout its environment. Furthermore, by planning over the different topological trajectory classes, the robot is able to reduce its decision space considerably. We hypothesize that by employing ISTP, autonomous systems will be able to plan non-myopic information gathering paths more efficiently and effectively.

Existing work in topological path planning uses well-defined features created by obstacles in the environment to differentiate between different types of trajectories. Such

This work is funded in part by NSF Grant IIS-1723924 and NSF Grant IIS-1845227.

✉ Seth McCammon
smccammon@whoi.edu

Geoffrey A. Hollinger
geoff.hollinger@oregonstate.edu

[1] Woods Hole Oceanographic Institution, Woods Hole, MA, USA

[2] Oregon State University, Corvallis, OR, USA

techniques have been successfully used to build topological representations of domestic environments (Topp and Christensen 2006), for planning paths for tethered robots (Bhattacharya et al. 2010), and for identifying meaningfully different types of trajectories in a robot's workspace (Pokorny et al. 2016). However, when we consider the IPPP, the features that separate dissimilar trajectories are not just physical obstacles in the environment. They also include features of the information space (i.e. the information reward function mapped across the region of interest). In order to exploit these informational topological features, new methods are required. The key insight of ISTP is that if we can create a model that captures the distribution of information, we can exploit it to quickly identify the trajectories that are likely to be high-quality. While the processes that drive the distribution of information may be too complex or random to model, their result, the distribution of information throughout an environment, is not. Since this distribution is non-uniform, it contains features, namely peaks and voids in the information function, which can be identified and used as topological features for planning purposes. Furthermore, in many informative path planning contexts, such as marine data collection and aerial surveillance, the environment is devoid of any meaningful physical obstacles that partition a robot's trajectory space. For the remainder of this paper we will use marine data collection as an motivating problem. However the concepts introduced and the methods proposed can be applied to any IPPP instance where decision making happens in a predominantly continuous space, as opposed to being confined to a discrete set of decision points (e.g. an underground, indoor, or urban environment).

As a first step in exploring ISTP, we propose and compare two informative path planning algorithms that leverage different topological planning paradigms to build models of the information function and then exploit those models to solve the IPPP. The first of our proposed methods, Hierarchical Hotspot Information Gathering (HHIG), identifies information hotspots that are used to construct a topological graph of the information function, that can then be used in a hierarchical informative path planner. While the graph based approach enables efficient planning at the global level, it introduces additional constraints to the plans, since the edges between hotspots are fixed. We address this limitation in our second contribution, Topology-Aware Self-Organizing Map (TA-SOM). Our TA-SOM algorithm first uses an adaptation of the Self Organizing Map algorithm to identify prominent topological features in the environment. Then, we use these features to provide topologically diverse initial trajectories to a Stochastic Gradient Ascent trajectory optimizer. Our HHIG algorithm and TA-SOM algorithm both enable robots to plan informative paths more effectively than existing information gathering methods. The HHIG algorithm offers faster computation than TA-SOM, but TA-SOM does not have the same constraints on the full set of trajectories that it can consider, and as a result plans more effective paths. Previous versions of these methods have appeared in the following conference papers (McCammon and Hollinger 2018; McCammon et al. 2020). In this paper, we compare the approaches HHIG and TA-SOM take to ISTP. We incorporate improvements to the TA-SOM algorithm in the form of a new information-weighted distance function for the SOM training, and demonstrate how the topologically distinct alternatives TA-SOM produces can be leveraged for multirobot planning. We also present additional experimentation and comparisons to state-of-the art information gathering algorithms and validation of our algorithms in real-world field trials.

The remainder of this paper is structured as follows. Section 2 provides an overview of related work in both topological path planning and autonomous information gathering. In Sect. 3, we provide our problem formulation and state key assumptions made by our proposed methods. Section 4 describes our first method, Hierarchical Hotspot Information Gathering (HHIG). Section 5 then outlines our second method, Topology-Aware Self Organizing Maps (TA-SOM). In Sect. 6, we demonstrate the utility of the two proposed approaches as well as compare their tradeoffs in simulated and real-world field trials with state of the art information gathering algorithms. Finally, in Sect. 7, we offer some closing remarks, and discuss potential avenues for future research.

2 Related work

ISTP leverages existing ideas about how robots can use the topological structure of their environment to plan efficiently, and applies those ideas to the problem of autonomous information gathering. Topological planning techniques used in robotic path planning can be broadly divided into two categories. The first of these categories contains methods that produce a topological representation of an environment, which is then used in a hierarchical path planning algorithm. The second encompasses methods that differentiate the topological classes of trajectories based on a set of features in the environment. However, despite the advantages that topological methods offer for robot path planning, these methods have not yet been exploited for robot information gathering.

2.1 Planning with topological representations

Metric environment representations, such as cost fields or occupancy grids, are commonly used in robotics planning applications. These methods are location based, meaning that they provide a label or a value at every location in an environment. In contrast, topological representations discard specific

location information, instead only representing the connectivity of salient regions in an environment (Thrun 1998). As a result, planning on topological maps is significantly more efficient, since the number of decision points is lower.

Topological mapping techniques have been used to derive abstract representations of environments from metric maps. An extensive body of work in this area focuses on segmenting indoor environments into regions that correspond with rooms and hallways (Bormann et al. 2016). A variety of approaches have been proposed for this task, including graph clustering and segmentation (Brunskill et al. 2007; Zivkovic et al. 2006) as well as Voronoi based segmentation (Friedman et al. 2007; Thrun 1998). 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 (Aurenhammer 1991), 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. In our proposed methods, we extend these approaches with alternative means of defining seed points and branching to allow these techniques to be applied to general information fields.

Once these high level environmental representations exist, there are a wide range of ways that the knowledge encoded within them can be used by robots for path planning. 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 (Kostavelis et al. 2016). This facilitates interactions with humans (Topp and Christensen 2006), since the semantic-topological mental representation closely echoes the mental models that humans build of their surroundings (McNamara 1986). 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 (Oßwald et al. 2016). 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 (Luperto and Amigoni 2019).

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 and urban environments, obstacles such as walls and buildings 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 (Huyer 1983) as well as Lagrangian coherent structures (Michini et al. 2014). Both fronts and Lagrangian coherent structures define regions in the ocean characterized by distinct dynamics. They are 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 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 (Ji et al. 2003; Lukasczyk et al. 2015). 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. We address this limitation by developing methods that do not rely on explicit thresholds. Instead we use characteristics of the environment in combination with principles such as persistence (Pokorny et al. 2016), to automatically identify meaningful features.

2.2 Planning with topological trajectory classes

In contrast with the previously discussed methods that directly model the connectivity of the environment, topological trajectory classes can be used to indirectly capture the connectivity of a space. They do this by describing the way that trajectories move through the environment relative to features, such as obstacles. Consequently, topological trajectory classes partition the robot's path space rather than partitioning its state space, like the methods described previously in Sect. 2.1.

Early work in this area focused on planning paths for cabled and tethered robots using homotopy classes. A homotopy class describes a set of trajectories that all start and end at the same pair of points, while allowing for a continuous, unobstructed deformation between any two trajectories within the class (Basener 2006). In their work, Bhattacharya et al. developed the H-signature, a homotopy invariant that uniquely describes a trajectory's homotopy class (Bhattacharya et al. 2012). They used this to construct a homotopy augmented graph, a topological structure that enabled a robot to plan a path to a point while constraining the robot's path to belong to a pre-specified homotopy class. While homotopy classes do provide global information about a trajectory, the information that they contain is not specific to any particular trajectory, since there are an infinite number of trajectories in any given homotopy class.

Existing methods which compute 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. Generally, this information is provided by an occupancy map of the environment. Additionally, like many search based algorithms, these methods perform an exhaustive search of the workspace to propagate the homotopy invariant to all sections of the graph. Consequently, the computational complexity of enumerating all homotopy classes in an environment scales not only with the size of the environment and the length of the paths being considered, but also with the number of obstacles in the environment (McCammon and Hollinger 2017). As a result, explicit enumeration of all homotopy classes is challenging in complex environments.

When the location of obstacles is uncertain, homotopy invariants can still be computed by applying thresholding to the uncertainty field at multiple levels, and comparing the existence of homotopy classes across multiple thresholds (Bhattacharya et al. 2015). However this adds an additional layer of computation, since a homotopy augmented graph must be constructed for each threshold level. A more efficient approach to discovering topological features in continuous fields is to use sampling based approaches that leverage simplicial complexes and persistent homology to identify prominent topological features (Edelsbrunner et al. 2000, 2006). In robotics applications filtrations of simplicial complexes have been used to identify topologically distinct trajectories (Pokorny et al. (2016)). While simplicial homology methods can efficiently determine whether two trajectories are topologically distinct, they do not enumerate all possible homology classes, limiting their use for path planning. To more efficiently identify the topological classes created by features of continuous information fields, we need to combine the elements of both types of methods. By first using persistent homology to identify prominent features, then using those features to construct a single homotopy augmented graph, we can still benefit from the abilities of the simplicial homology methods to discover features in the environment, while maintaining the ability to define all sets of homotopy classes.

2.3 Autonomous information gathering

The Informative Path Planning Problem (IPPP) is a widely researched problem in robotics. It is particularly challenging, since the size of the space of possible paths scales exponentially with mission duration and, in all but the simplest environments, the reward function over this space of paths is nonconvex, containing many local maxima. These facts make it difficult to find the globally optimal information gathering path, and the IPPP is well-known as being classified as NP-Hard (Hollinger and Sukhatme 2014).

Many algorithms have been proposed to solve the IPPP. Algorithms such as Greedy, Recursive Greedy (Singh et al. 2007), and Branch and Bound (Binney and Sukhatme 2012) approach the information gathering problem as a problem of sequential decision making. In this paradigm, each action by the robot is evaluated independently. This process means that at each iteration of the algorithm considers adding a new action to the robot's path, but it must evaluate the effects of this action as it relates to all previous actions. This necessitates the re-evaluation of the entire path reward each time a new action is considered, making the sequential path construction approach inefficient in evaluating paths in a global context. Sampling based methods, such as Rapidly exploring Information Gathering (RIG) (Hollinger and Sukhatme 2014), partially address this by sampling from the global information field; however they too result in each action being added to the information tree needing to be evaluated individually.

An alternative approach for information gathering is to use trajectory optimization techniques to refine an preexisting or naïve trajectory. In Charrow et al. (2015), the authors formulate the information gathering problem as a Sequential Quadratic Programming problem. Doing so requires them to treat each sensor measurement as independent. To allow for path dependent rewards, the authors in Popović et al. (2017) utilize an evolutionary algorithm to optimize the path of the robot. However, evolutionary algorithms are computationally expensive and do not scale well as the problem size increases.

From general trajectory optimization, a number of algorithms such as STOMP (Kalakrishnan et al. 2011), and EESTO (Jones and Hollinger 2017) have been developed for problems where analytical gradients are difficult to calculate. These methods rely on sampling to estimate a gradient for the desired function. Closest to our work is Jones et al. (2018), which uses Stochastic Gradient Ascent (SGA) to plan informative paths for a team of vehicles. However, the authors only consider a single path initialization and rely on the sampling to be large enough that the paths do not get stuck in local maxima.

In our work we aim to improve on these existing methods that solve the IPPP problem by directly incorporating global information about the topological structure of the robot's information space. Our two algorithms, HHIG and TA-SOM, fall within the topological planning paradigms discussed in Sects. 2.1 and 2.2, respectively. In our experiments, we will compare their effectiveness of these approaches, both in terms of their performance on solving the IPPP, and on their ability to reduce the computation necessary to produce effective solutions.

3 Problem formulation and assumptions

In this paper, we consider a typical formulation for the Informative Path Planning Problem (Singh et al. 2007). We are

interested in finding an optimal path for a robot, which satisfies the following equation:

$$\mathcal{P}^* = \underset{\mathcal{P}\in\Phi}{\text{argmax}}\{I(\mathcal{P})\}\ s.t.\ C(\mathcal{P}) \leq B, \tag{1}$$

where $\mathcal{P}$ is a path defined as a series of waypoints in $\mathbb{R}^2$. The optimal path, $\mathcal{P}^*$, is the path from the space of all possible paths Φ that maximizes the information function $I(\mathcal{P})$, subject to a budget constraint: that the cost of a path $C(\mathcal{P})$ must not exceed the robot's overall mission budget B. Many different cost functions, such as energy consumed, distance travelled, or time elapsed, may be used to evaluate the cost of a particular path. In this paper we assume that the robot travels at a constant velocity and that it possesses sufficient actuation that the cost of overcoming environmental disturbances is negligible. As a result, the energy, distance, and time costs are interchangeable. However, it is worth noting that our proposed method is agnostic to the particulars of the cost function used, and it is a straightforward extension to consider alternatives. In this paper, we define $\Psi(\cdot) : \mathbb{R}^2 \rightarrow [0, 1]$ as the information reward function that maps each location in the environment to the value of observing that location. The robot's information reward function, $I(\mathcal{P})$, is the total amount of information in Ψ within the robot's sensor radius as it travels along $\mathcal{P}$.

The two approaches which we propose in this paper both take ISTP approaches to solve this problem. In general, ISTP algorithms are defined as algorithms which exploit the topological structure of the information space to plan informative paths. The key features of an ISTP algorithm are (1) a method to identify features from the information space of a robot, (2) a method to construct a high level topological representation using these features, and (3) a method for planning using the topological representation of the information space.

We assume that the distribution of information in the environment is both known and static. These assumptions are a consequence of limitations of topological planning techniques. Topological planning techniques utilize features within the environment to separate Φ into a much smaller set of trajectory classes. Since our proposed planners are both one-shot planners (i.e. they do not incorporate replanning), they must be aware of the existence of the features in the world in order to account for them in planning. While there has been some work on topological path planning in partially known or uncertain environments, the topological features used by the planners either exist entirely in the known portion of the world (Kim et al. 2013), or a predictive model is used to make assumptions about the existence of features in unknown portions of the environment (Saroya et al. 2020). In both cases, replanning is used to account for the discovery of previously unknown features. Our methods could also be used in a replanning framework, however this will be left for future research. Since our methods focus on understanding the topology of the information space of a robot, for the moment we will assume that the robot's environment lacks significant obstacles that divide Φ. For many field robotics applications, such as aerial and marine operations, this assumption is true. We leave it as future work to explore the relationship between the topologies induced by the physical environment and ISTP.

4 Hierarchical hotspot information gathering

The first ISTP algorithm we will present is Hierarchical Hotspot Information Gathering (HHIG). Similar to the methods for constructing topological graphical environment representations (Thrun 1998) from indoor environments, HHIG builds a topological representation of the information space by partitioning it into a graph of high-information hotspots.

The HHIG planner 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.

4.1 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 topological connectivity. Using the robot's estimate of the normalized information in the environment, $\Psi(\cdot)$, we will construct a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ that captures the underlying topology of $\Psi(\cdot)$.

Each $v_i \in \mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is a region in space containing one or more points of interest, which we define as local maxima of $\Psi(\cdot)$. In monitoring tasks, a relative increase in the occurrence of a phenomena (e.g. the presence phytoplankton in the marine domain or pollutant concentration in the aerial domain), can provide valuable data about the causes of larger trends in post hoc analysis of the data by human experts. It is worth noting that just because a location is identified as a point of interest, does not mean that it will ultimately be visited by the robot due to constraints on the

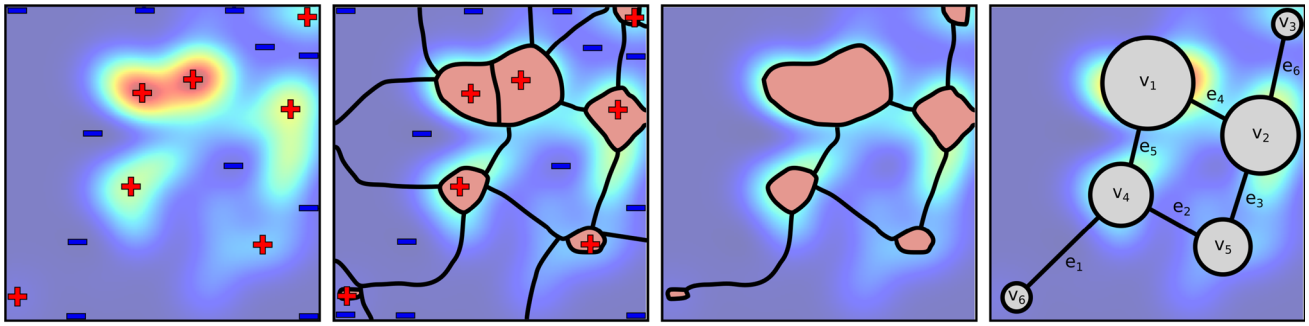


Fig. 1 **a** The selected maxima and minima. **b** The growth of the labelled regions around each s_{max} and $s_{min} \in \mathcal{S}$. **c** The merging of adjacent maxima regions and the creation of edges between each hotspot. Finally, **d** the resultant topological graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ (Color figure online)

robot's travel, such as a finite energy budget. However, identifying all such locations enables the robot to reason about them in the second step of the HHIG planner, discussed in Sect. 4.2.

Since at the topological level, we do not have a specific path, and therefore cannot compute $I(\cdot)$, we use an estimate function $\hat{I}(\cdot)$. Each $v_i \in \mathcal{V}$ has a corresponding estimate of its local reward function, $\hat{I}_i(t_i)$, which estimates the information reward gained in v_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 Yu et al. (2015), which captures the submodular nature of the information gathering task:

$$\hat{I}_i(t_i) = a_i(1 - e^{-b_i t_i}), \tag{2}$$

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

$$b_i = \frac{obs_r^2 \times \pi}{A_i}. \tag{3}$$

The vertices of $\mathcal{G}$ are connected by a set of edges, $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$, 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 is discussed further in Sect. 4.2.

The process of constructing a topological hotspot graph is illustrated in Fig. 1. The first step is to identify the local maxima and minima of the information field, Ψ. We begin by creating a discrete approximation of the global information function by densely sampling it in a regular grid pattern. Using this discrete approximation of Ψ, 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. The maxima and minima points for a sample environment are shown in Fig. 1a.

Once $\mathcal{S}$ is constructed, it is used as the seed points for our modified Fast Marching (FM) expansion method. As outlined in Sethian (1999) and Petres et al. (2007), the standard FM algorithm approximates a solution to the Eikonal equation:

$$||\nabla u(i, j)|| = \tau(i, j),$$

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. For our purposes, we will use FM as a variable-rate segmentation method, defining hotspots as regions where wavefronts that propagate from maxima arrive before wavefronts that arrive from minima, as shown in Fig. 1b. On the Cartesian grid with spacing h produced by our earlier discrete sampling of Ψ we can compute an estimate of the magnitude of the gradient ∇u in both the x and y directions at a point (i, j) with

$$\begin{aligned} ||\nabla u_{i,j}||^2 \approx \tau_{i,j}^2 = [&max(D_{i,j}^{-x}, -D_{i,j}^{+x}, 0)^2 \\ &+ max(D_{i,j}^{-y}, -D_{i,j}^{+y}, 0)^2], \end{aligned} \tag{4}$$

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

$$D^{+x}_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{h}, \; D^{-x}_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{h},$$
$$D^{+y}_{i,j} = \frac{u_{i,j+1} - u_{i,j}}{h}, \; D^{-y}_{i,j} = \frac{u_{i,j} - u_{i,j-1}}{h}$$

The upwind scheme uses a breadth-first update method to iteratively select a trial point that is 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 they do not already belong to it.

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

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

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 accomplish this by propagating 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 $x_{i,j}$ which descends from a $s_{max} \in \mathcal{S}_{max}$, we define $\tau_{i,j}$ as $1 - \Psi(i, j)$. For an $x_{i,j}$ which descends from a $s_{min} \in \mathcal{S}_{min}$, $\tau_{i,j} = \Psi(i, j)$. The result of this is that regions expanding from maxima expand more easily in high-information areas, and regions expanding from minima expand more easily to cover low-information areas. This expansion process is shown in Fig. 1b.

To construct a graph from the labelled regions, we merge adjacent regions grown from maxima, as depicted in Fig. 1c, and then label each combined region as a hotspot, and add it to $\mathcal{V}$. The interfaces between regions grown from minima then become the edges between the hotspot vertices. These interfaces are equidistant between local minima over Ψ, and therefore correspond to relatively information-rich paths between two vertices. The resulting topological graph for a sample environment is shown in Fig. 1d.

It is possible for the topological graph construction to produce a disconnected graph in some environments, such as one where a hotspot is completely enclosed by a single region grown from a local minima. In this case, we use Fast Marching to extend an edge from the isolated hotspot to the nearest edge or hotspot, connecting it to the graph. It is also possible for a single hotspot region to entirely enclose a local minima. In this case, the enclosed minima has no effect on the connectivity of the resulting graph, and the graph contains 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 such as the one employed in Pokorny et al. (2016) could be employed to identify and eliminate the holes in a hotspot. However it is not clear if a hole that is a result of multiple local minima should be eliminated in this manner or not. A more detailed examination of this is outside the scope of this paper.

4.2 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 in Yu et al. (2015, 2016), the problem that we seek to solve is to identify an informative schedule, $\Omega = (\mathcal{V}_\Omega, \mathcal{E}_\Omega, \mathcal{T})$, where $\mathcal{V}_\Omega \subset \mathcal{V}$ is the set of unique vertices visited along the path, $\mathcal{E}_\Omega \subset \mathcal{E}$ is the set of edges that the robot traverses, and $\mathcal{T}$ is the set of times, t_i, spent at each $v_i \in \mathcal{V}_\Omega$. 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. As stated in Sect. 3, we assume that Ψ is static during the planning and execution of a trajectory. Thus, there is no difference in the estimated information reward between visiting a vertex twice, or visiting it once for twice as long. 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.

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 then 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,

$$c(\Omega) = \sum_{i=1}^{|\mathcal{V}_\Omega|} t_i + \sum_{i=1}^{|\mathcal{E}_\Omega|} \frac{\ell_i}{vel_r} \leq B, \tag{5}$$

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

While this can be a potentially large number of paths, this number is kept relatively low by several factors. Chief among these is the fact that since we are planning using a graph constructed from topology of a smoothly varying information function, the graphs we develop will have no more vertices than the information function has local maxima. Typically it will have fewer, since a single hotspot often is produced from multiple local maxima. In practice, we found that these graphs rarely consisted of more than 10 vertices. Despite the low number of paths, graphs with a particularly high branching factor can lead to an intractable number of possible paths. To combat this, we 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. Using this, we can compute the amount of our budget remaining, R, using

$$R = B - \sum_{i=1}^{|\mathcal{V}_\Omega|} t_i^- - \sum_{i=1}^{|\mathcal{E}_\Omega|} \frac{\ell_i}{vel_r}. \tag{6}$$

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 $\mathcal{T}^+ = \{t_1^+, t_2^+, \ldots, t_{|\mathcal{V}_\Omega|}^+\}$ that maximize

$$\sum_{i=1}^{|\mathcal{V}_\Omega|} I_i(t_i) \; s.t. \; \sum_{i=1}^{|\mathcal{V}_\Omega|} t_i^+ \leq R, \tag{7}$$

using a Lagrange Multiplier Method (Everett III 1963) to solve the resource-constraint problem inherent in allocating R among $t_i^+ \in \mathcal{T}^+$. We construct our Lagrange Function, $\mathcal{L}$, using the Lagrange Multiplier variable, λ,

$$\mathcal{L}(t_1^+, t_2^+, \ldots, t_{|\mathcal{V}_\Omega|}^+, \lambda) = \sum_{i=1}^{|\mathcal{V}_\Omega|} I_i(t_i) + \lambda \left(R - \sum_{i=1}^{|\mathcal{V}_\Omega|} t_i^+ \right). \tag{8}$$

We then take the partial derivative with respect to each $t_i^+ \in \mathcal{T}^+$, as well as λ. Setting these equal to 0 yields the following system of equations:

$$\forall\, 1 \leq i \leq |\mathcal{V}_\Omega|, \; \frac{\partial \mathcal{L}}{\partial t_i^+} = a_i b_i e^{-b_i (t_i^+ + t_i^-)} - \lambda = 0 \tag{9}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = R - \sum_{i=1}^{|\mathcal{V}_\Omega|} t_i^+ = 0. \tag{10}$$

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 Eq. 9 equal to each other:

$$t_i^+ = \frac{-ln\left(\frac{a_1 b_1}{a_i b_i}\right) + b_1 (t_1^+ + t_1^-)}{b_i} - t_i^-. \tag{11}$$

Substituting Eq. 11 into Eq. 10 yields:

$$t_1^+ = \frac{R - \sum_{i=2}^{|\mathcal{V}_\Omega|} \left[\frac{ln\left(\frac{a_1 b_1}{a_i b_i}\right)}{b_i} - \frac{b_1 t_1^-}{b_i} + t_i^- \right]}{1 + \sum_{i=2}^{|\mathcal{V}_\Omega|} \frac{b_1}{b_i}}, \tag{12}$$

which we can then use to solve for t_1^+.

Taken together, Eqs. 11 and 12 can be used to compute the optimal values for all $t_i^+ \in \mathcal{T}^+$ along a given schedule, Ω.

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 Ω^* that maximizes our expected reward, thus resulting in the optimal schedule for the robot. The effects of this process can be seen in Fig. 2. As the mission budget increases, the portion of the environment covered by the robot's sensor (the region

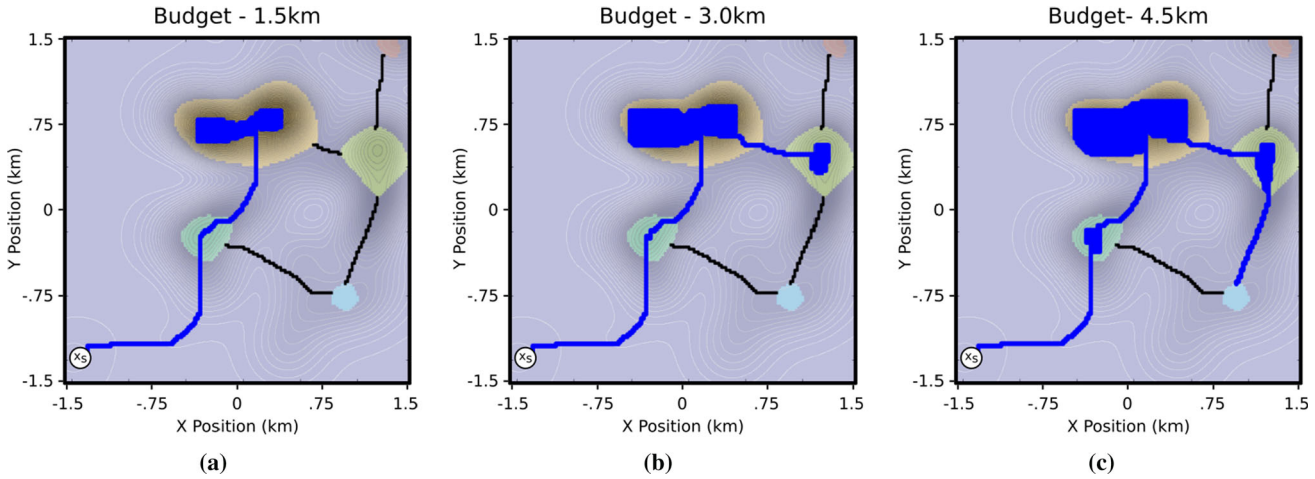


Fig. 2 Informative paths planned by HHIG with budgets of **a** 1.5 km, **b** 3.0 km, and **c** 4.5 km. The area explored by the robot is shown in blue. As the budget increases, our method is able to balance the additional information gained by continuing to explore the current hotspot with the information gained by exploring new hotspots (Color figure online)

Algorithm 1 Hotspot Scheduling

function HOTSPOTSCHEDULE($\mathcal{G}$, B)
  tree = constructTree($\mathcal{G}$, B)
  for each node **in** tree **do**
    Ω = tracePathToRoot(node, tree)
    $\{t_1, t_2, \ldots, t_{|\mathcal{V}_\Omega|}\}$ = schedule(Ω)
    $\hat{I} = \sum_{i=1}^{|\mathcal{V}_\Omega|} \hat{I}(t_i)$
  Ω^* = argmax$_{\Omega \in tree}(\hat{I})$
return Ω^*

shaded in blue) grows. When the budget is increased from from 1.5 to 3.0 km, the additional budget is spent exploring the largest hotspot, as well as exploring a brand new hotspot. As the budget increases again to 4.5 km, the robot is able to spend more time in the first hotspot, which at lower budgets it had simply passed through.

4.3 Path planning

The result of the scheduling algorithm is the optimal topological plan over the graph based on the estimated information content of each hotspot, Ω^*. In order for a robot to execute Ω^*, it must be translated back to a set of actions for the robot to take in the metric space. Within a vertex v_i, the robot must plan a path, $\mathcal{P}^*$ that utilizes the allocated budget, t_i^+, to collect the maximum amount of information within the hotspot vertex, v_i.

We use a greedy-coverage algorithm to quickly compute a path, $\mathcal{P}_{v_i}$, for the robot within v_i. This 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 naïve information gathering coverage approach more effective than it would otherwise be. However, more sophisticated planners, such as Branch and Bound (Binney and Sukhatme 2012), or stochastic trajectory optimizers, such as STOMP (Kalakrishnan et al. 2011), could be used to compute better information gathering paths within a vertex at the expense of additional computation time.

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, it 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.

The final path, $\mathcal{P}^*$, is produced by combining the paths along the selected edges with the greedy-coverage paths within the vertices. Beginning at the robot's starting location, we append the metric space path for each vertex, $\mathcal{P}_{v_i}$, along with the metric space path for each edge, $\mathcal{P}_{e_i}$, for each hotspot vertex, v_i, and edge, e_i, in the order that they appear in the optimal schedule, Ω^*.

5 Topology-aware self-organizing map

While our HHIG planner captures the topology of information 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

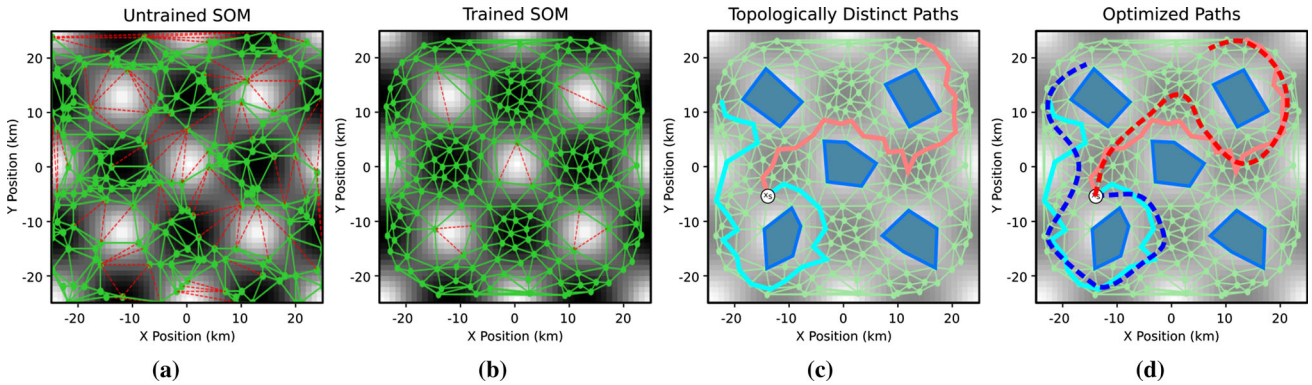


Fig. 3 Training of an SOM on a Gyre world. Darker regions contain more information. Before training (**a**), the topological features identified by the SOM (areas with green edges) are driven by the sampling noise and indistinguishable from the true topological features, (areas in white). Once trained **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**) (Color figure online)

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. Figure 3 illustrates our proposed method, which uses a novel Topology-Aware Self Organizing Map (TA-SOM) to build an environment model that can be used to compute unique homotopy classes.

In many previous domains where topological techniques have been utilized (e.g. Bhattacharya et al. 2012; Pokorny et al. 2016), 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 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.

5.1 Persistent homology

Persistence is a measure of the importance of a topological feature. It is computed by taking the difference between the birth time and death time of a feature. Persistence has been used to plan paths in uncertain obstacle fields (Kim et al. 2013), as well as to act as an adaptive threshold to identify topologically distinct trajectories in an environment (Zomorodian and Carlsson 2005; Pokorny et al. 2016). This is accomplished 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.

One way to visualize the persistence of multiple topological features is through the use of a persistence diagram. An example diagram that identifies two features for a graph is shown in Fig. 4. 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.

5.2 Identifying homotopy classes with self-organizing maps

At a high level, the Self Organizing Map algorithm is a method for fitting a graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, to a target function, $\Psi(\cdot)$ (Kohonen 1990). In robotics applications, SOMs

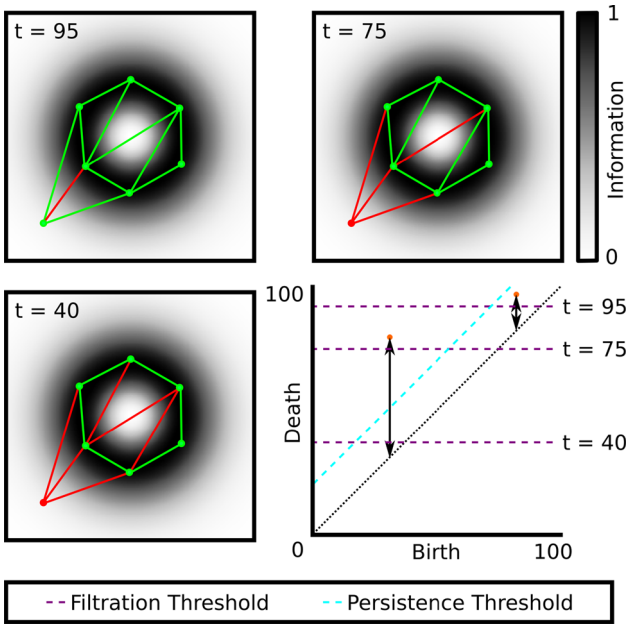


Fig. 4 An example of a persistence diagram for a simplicial complex. Using the persistence of 1st order topological features (orange dots), we identify a persistence threshold (cyan), classify features as 'persistent' and 'ephemeral', and set a corresponding horizontal filtration threshold (purple) that is used to alter the SOM topology. Three different filtration thresholds and their corresponding simplicial complexes are shown (Color figure online)

have been used as a method to solve the Travelling Salesperson Problem (Faigl et al. 2011; Somhom et al. 1997), as well as the TSP's common extensions, such as the Orienteering Problem (Faigl et al. 2016), and other information gathering tasks (Faigl and Hollinger 2017). However, a significant issue with existing methods for SOMs is that the topology of the graph, $\mathcal{G}$, is fixed prior to training. Different graph topologies can have an enormous impact on final positions of the graph vertices (Best 2019). Consequently, choosing the correct one can require a significant amount of domain knowledge.

We propose improving 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 by interleaving the training process with a series of filtration steps, each of which modifies the graph topology, removing and adding edges. 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 2. It relies on two sub-processes, the standard SOM training function, **TrainSOM**, and our 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

Algorithm 2 Topology-Augmented Self-Organizing Map

```
function TopologySOM(I(·), N)
    V ⟵ DrawSamples(N, I(·))
    E ⟵ DelaunayTriangulation(V)
    G ⟵ (V, E)
    while ¬ stopping do
        Ĝ ⟵ TrainSOM(G, I(·))
        G ⟵ Filtration(Ĝ, I(·))
    return G
```

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

$$\text{InformationDist}(\psi, v) = \frac{|\overline{v\psi}|}{\int_{\overline{v\psi}} I(s)ds}, \tag{13}$$

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:

$$\text{Neighborhood}(v_0, v_1, \mathcal{G}) = \frac{1}{1 + \text{GraphDist}(v_0, v_1, \mathcal{G})^{\gamma}}, \tag{14}$$

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

$$\sum_{i=0}^{|\mathcal{V}|} ||\bar{v}_i|| \leq T, \tag{15}$$

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 3.

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 in our **Filtration** function, which determines the 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

Algorithm 3 Self-Organizing Map

```
function TrainSOM(G = (V, E), Ψ(·))
    while ¬ stopping do
        ψ ⟵ DrawSamples(1, Ψ)
        v* ⟵ argmin_{v∈V} (InformationDist(ψ, v))
        for v ∈ V do
            v̄ ⟵ (v − φ) × λ × Neighborhood(v, v*, G)
            v ⟵ v + v̄
    return G
```

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 (Basener 2006). This is true, since the edges 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 (The GUDHI Project 2014), 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, this 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 (Rinne 2008), 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 α 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 α 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. 4. 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. 4. 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 a TA-SOM is trained (Fig. 3b), it can be used to enumerate the possible homotopy classes of trajectories. To accomplish this, we use the homotopy augmented graph proposed in Bhattacharya et al. (2012). 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, $I(\cdot)$. Some example reference trajectories in different topological classes are shown in Fig. 3c.

While the TA-SOM method does contain several parameters that need to be set by hand, many of these, such as T, λ, and the number of training epochs are similar to ones found in a wide range of optimization algorithms, and they regulate the speed of convergence of the SOM to avoid local minima. The key parameters that affect the TA-SOM's ability to distinguish true topological features from ephemeral ones are the persistence threshold parameter, α, as well as N, the number of vertices in the TA-SOM. If α is too low, the filtration step will be too permissive, and will result in some ephemeral features being classified as persistent. This can be counteracted by increasing N. With additional vertices in the TA-SOM, it is easier to distinguish the ephemeral features from the persistent ones, reducing the importance of setting α correctly. However, as is shown in Figs. 7 and 8, increasing the number of vertices also increases the training time required.

5.3 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, as shown in Fig. 3d. 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 (Jones et al. 2018). At a high level, SGA operates by estimating the gradient by sampling perturbations and recombining them using a weighting based upon the objective function. Pseudocode for the SGA algorithm is presented in Algorithm 4.

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:

$$\mathcal{D} = \mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}\sigma_x & 0\\ 0 & \sigma_y\end{bmatrix}\right) \tag{16}$$

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 defined the perturbations as independent but this is not required. 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, $\epsilon_i \in \boldsymbol{\epsilon}$ 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:

$$\Delta = \frac{1}{K}\sum_{k=1}^{|K|} w_k \times \epsilon_k, \tag{17}$$

where

$$w_k = e^{-h\left(\frac{s_k - \min \mathbf{s}}{\max \mathbf{s} - \min \mathbf{s}}\right)}, \tag{18}$$

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 TA-SOM training algorithm, a discount factor, λ, is used to facilitate convergence.

Algorithm 4 Stochastic Gradient Ascent (SGA)

```
function SGA(P, I(·))
    while ¬ stopping do
        for p ∈ P do
            ε ⟵ genPertubations(D, K)
            s ⟵ getScores(P, ε, p, I(·))
            Δ ⟵ calcGrad(s, ε)
            p ⟵ p + Δ × λ
    return P
```

5.4 Analysis

SGA is guaranteed to almost surely converge to a local maxima (Kiwiel 2001) 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 Sect. 6, we confirm this hypothesis empirically through comparisons with an SGA variant that is initialized using a heuristic that does not include topological information.

5.5 Extension to multi-robot planning

Topologically distinct trajectories offer an elegant way to distribute different members of a multi-robot team to explore an environment by assigning different robots to different homotopy classes, as described in Kim et al. (2013). 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 multi-robot informative path planning is the same as the single robot case. We use the TA-SOM algorithm defined in Algorithm 2 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 multi-robot 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 (Bhattacharya et al. 2012) to identify all possible homotopy classes of trajectories. Then, we produce a representative trajectory for each of these homotopy classes. The result of this is a set of topologically distinct representative trajectories for each member of the robot team.

To produce a single plan for the entire team of robots, we need to assign each robot to a single class and correspond-

ing representative trajectory, which we will then be able to optimize. The first step in this process is to evaluate each combination of the five most promising representative trajectories for each robot and select the five best joint plans. While this 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. The second step is to use the SGA optimizer described in Sect. 5.3 to optimize the joint plans. In the single-robot case, while generating perturbations (Algorithm 4, Line 4), we perturb each waypoint in $\mathcal{P}$ in a random order. To avoid biases in the multi-robot 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 Results

To demonstrate the benefits of considering topological features while solving the IPPP, we performed several experiments both in simulation, using real-world ocean modelling datasets, as well as in hardware with an autonomous boat on a small 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.

6.1 Evaluation datasets

Our primary simulated dataset consists of 20 worlds built using real-world data taken from the Regional Ocean Modelling System (ROMS)[1] (Shchepetkin and McWilliams 2005). Each ROMS world is created from a 35 by 35 km section in the center of Monterey Bay, California and is resolved at a grid resolution of 700m. 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 localization of upwelling fronts.

In addition, we use two other datasets to produce worlds for illustrative purposes. The gyre world, shown in Fig. 3b, is a hand-constructed environment that contain a quadruple-gyre system, similar to the worlds used in Kularatne et al. (2018) 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 (Benoit-Bird et al. 2018). This dataset can be seen in Fig. 1a.

[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.

6.2 Topological feature detection

Both of our proposed methods must be able to to correctly identify topological features in the environment. In the HHIG planner, the hotspot regions should cover only areas of the environment with high levels of information, while simultaneously ensuring that these hotspots cover all points of interest in the environment. In the TA-SOM planner, the trained TA-SOM should have high connectivity in high information areas, and it should have holes in the SOM triangulation that correspond with the low information voids in the environment.

Our first set of experiments assess the effectiveness of the hotspot segmentation within the HHIG planner. We compare our Fast Marching segmentation method with thresholding, where each point with information value greater than a threshold is considered part of a hotspot. We use two different thresholding variants: a static threshold of 0.5, as well as an adaptive threshold, which is set to capture all PoIs. This is done by setting the threshold value equal to the information value of the lowest PoI. In Fig. 5, we show each thresholding method on a dataset bioacoustic activity collected by Slocum gliders in Monterey Bay, California. A qualitative examination of the three segmentation techniques highlights the drawbacks of the two standard thresholding approaches to segmentation. Using a constant threshold set at 0.5, as seen in Fig. 5b, fails to capture a number of the PoIs. Adjusting this threshold to capture the lowest-valued PoI with an information value of 0.05 collects nearly all of the the environment into one giant hotspot, as can be seen in Fig. 5c. In contrast, our proposed Fast Marching hotspot expansion, shown in Fig. 5a both captures each Point of Interest, as well as produces a set of discrete, compact, and high-value hotspots.

We compared the three segmentation methods: Fast Marching, Static Thresholding, and Adaptive Thresholding across a set of 20 simulated environments drawn from the ROMS model. 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 environment as a whole. The results of these trials are shown in Fig 6a, b, respectively. Our method shows an increase in hotspot density over the adaptive thresholding method, while maintaining 100% of Points of Interest captured in hotspot

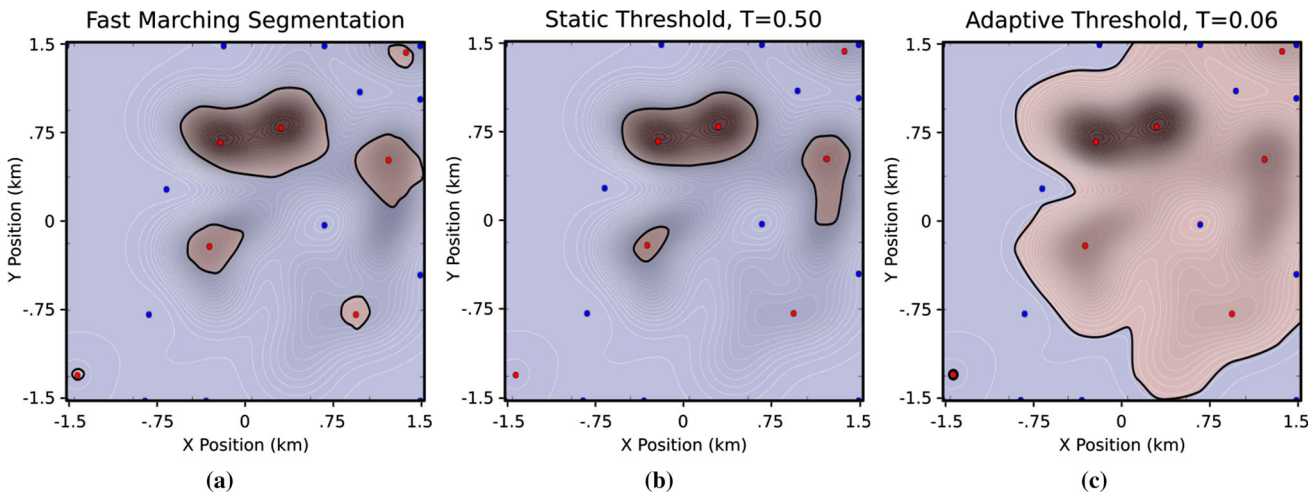


Fig. 5 Comparison of segmentation methods. Areas labeled as hotspots are shown in red. Segmentation is performed on the map of bioacoustic data shown in **a**, normalized to the range [0,1]. **a** Our fast marching based method for identifying hotspots. **b** The hotspots identified using static thresholding (activity > 0.5) and **c** the hotspots identified using an adaptive threshold set to capture each point of interest (Color figure online)

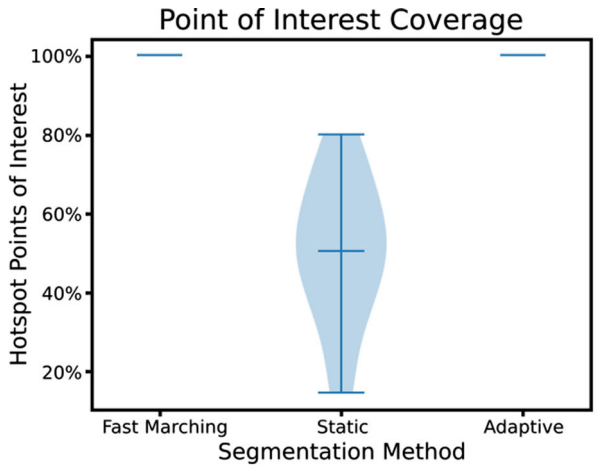
regions. While our segmentation method does produce less-dense hotspots when compared to a static threshold, the static thresholding misses a significant portion of the points of interest. This is problematic from an ocean science perspective, since many of the phenomena that we are interested in monitoring 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.

To evaluate the ability of our Topology-Aware Self Organizing Map to learn the topological features, 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. 7. 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 maps is shown in Fig. 8. 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 Pokorny et al. (2016). These results clearly show that by using the training process of a Self Organizing Map 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.

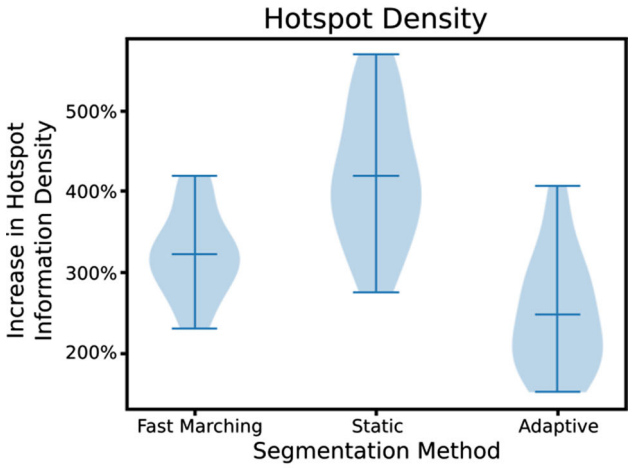
6.3 Single robot information gathering

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 Sect. 4.
- **TA-SOM**—Our Topology-Aware Self Organizing Map planner described in Sect. 5, 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 TA-SOM framework. Instead of using a Topology-Aware Self Organizing Map to identify topological features in the environment, this method uses the algorithm in Bhattacharya et al. (2015) as an alternative method of proposing topologically distinct trajectory classes within the TA-SOM framework. The method uses the persistence of homotopy classes identified by creating multiple homotopy augmented graphs across multiple thresholds of the information function to identify a set of persistent homotopy classes of trajectories. Once the homotopy classes have been identified, the Path Persistence method is identical to TA-SOM, with representative trajectories from the five most persistent homotopy classes optimized using our SGA framework. Comparisons to



(a)



(b)

Fig. 6 Results comparing our fast marching hotspot segmentation algorithm to static thresholding (activity$>$ 0.5) and adaptive thresholding on the simulated ROMS worlds. 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 (Color figure online)

this algorithm are meant to demonstrate the computational benefits of the TA-SOM and HHIG approaches in identifying topological features of information fields for planning.

– **RRT-OPT**—A method that uses a Rapidly exploring Random Tree (RRT) to quickly build a large number of paths through the environment (LaValle 1998). 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.

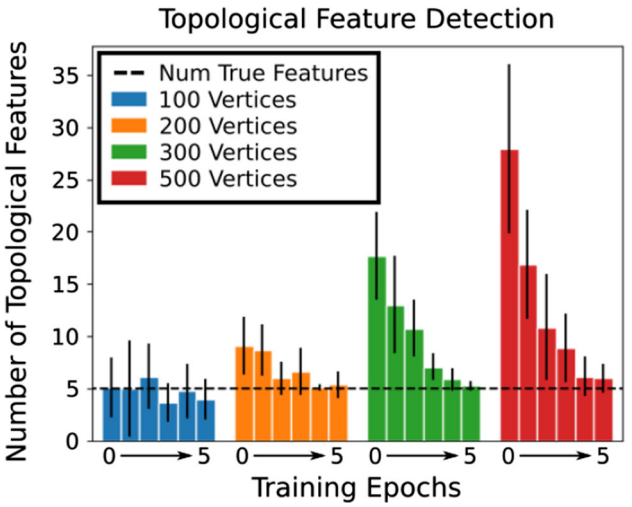


Fig. 7 Number of topological features found by the TA-SOM in the gyre world for different numbers of vertices across up to five training epochs. The black dashed line at five is the true number of topological features in this environment (Color figure online)

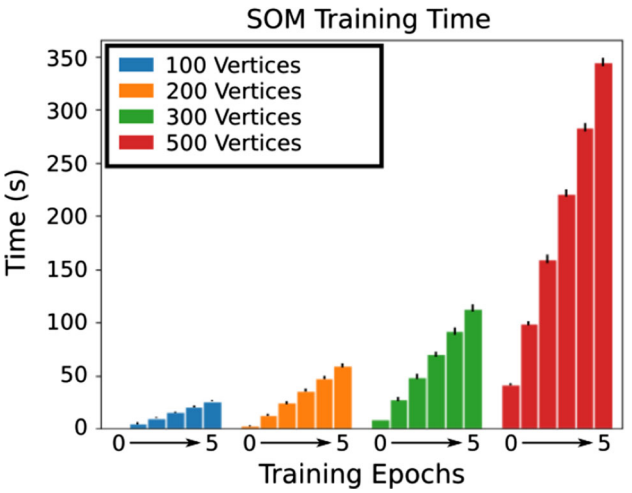


Fig. 8 Amount of time to train the TA-SOM for different numbers of vertices and training epochs in the gyre world (Color figure online)

– **BnB**—The Branch and Bound informative path planner (Binney and Sukhatme 2012). 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 sam-

pling 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. 9. The first trend that we notice is, predictably, as the budget increases from 35 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. Using a planning budget of 35 km, all six methods perform competitively. As the budget increases to 70 km, the TA-SOM and Path Persistence methods significantly outperforms all the non-topological methods, while HHIG only significantly outperforms BnB and Greedy ($p < 0.05$). Finally, with a budget of 105 km, the topological methods on average, all collect more information than the non-topological planners. HHIG, TA-SOM, and Path Persistence were able to collect a respective average of 63.04%, 64.39%, and 65.78% of the total information across the 20 environments. In comparison, the non-topological methods: RRT-OPT, Branch and Bound, and Greedy attained an average of 48.01%, 54.50%, and 58.02%, respectively. With this budget, HHIG, TA-SOM, and Path Persistance all perform significantly better than Branch and Bound and RRT-OPT ($p < 0.05$). While the topological methods do not have statistically significant performance gains over Greedy at this budget, they do perform better on average, and have considerably lower variance in performance, suggesting that they more reliably find better paths with fewer exceptionally poor outliers.

The planning horizon also affects the importance of non-myopic decision making. At lower budgets, (i.e. with shorter paths), in a given environment there are fewer topological decisions to make. This reduces the impact that considering topological features can have relative to the non-topological methods, equalizing their performance. On the other hand, 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 can be seen comparing Fig. 9a with c. The non-topological, non-myopic planners struggle with the largest planning horizon, reflecting how the expanded decision space impacts their ability to converge to the global optimal path. 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 s 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. Taken together, these results support our hypothesis that the additional global context about the structure of the information distribution provided by ISTP enables more effective and more efficient non-myopic planning.

Comparing the three planners that leverage ISTP, we can see that TA-SOM, and Path Persistence, all have competitive performance with each other across all three planning budgets. Instead, they are differentiated by the amount of computation time 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 topological planning paradigm than the other two topological methods. HHIG constructs a sparse topological graph, which 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 a manageable search. In contrast, both the TA-SOM and Path Persistence algorithms rely on homotopy augmented graphs (Bhattacharya et al. 2012) 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. 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.

6.4 Multi-robot information gathering

The final set of experiments examined the ability of the TA-SOM and Path Persistence methods to scale in multi-robot 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 multi-robot planning problem. We compare these two methods with a baseline Greedy algorithm. We do not extend our HHIG planner to the multi-robot case, since the inclusion of multiple robots into the topological graph scheduling problem is less of a straightfor-

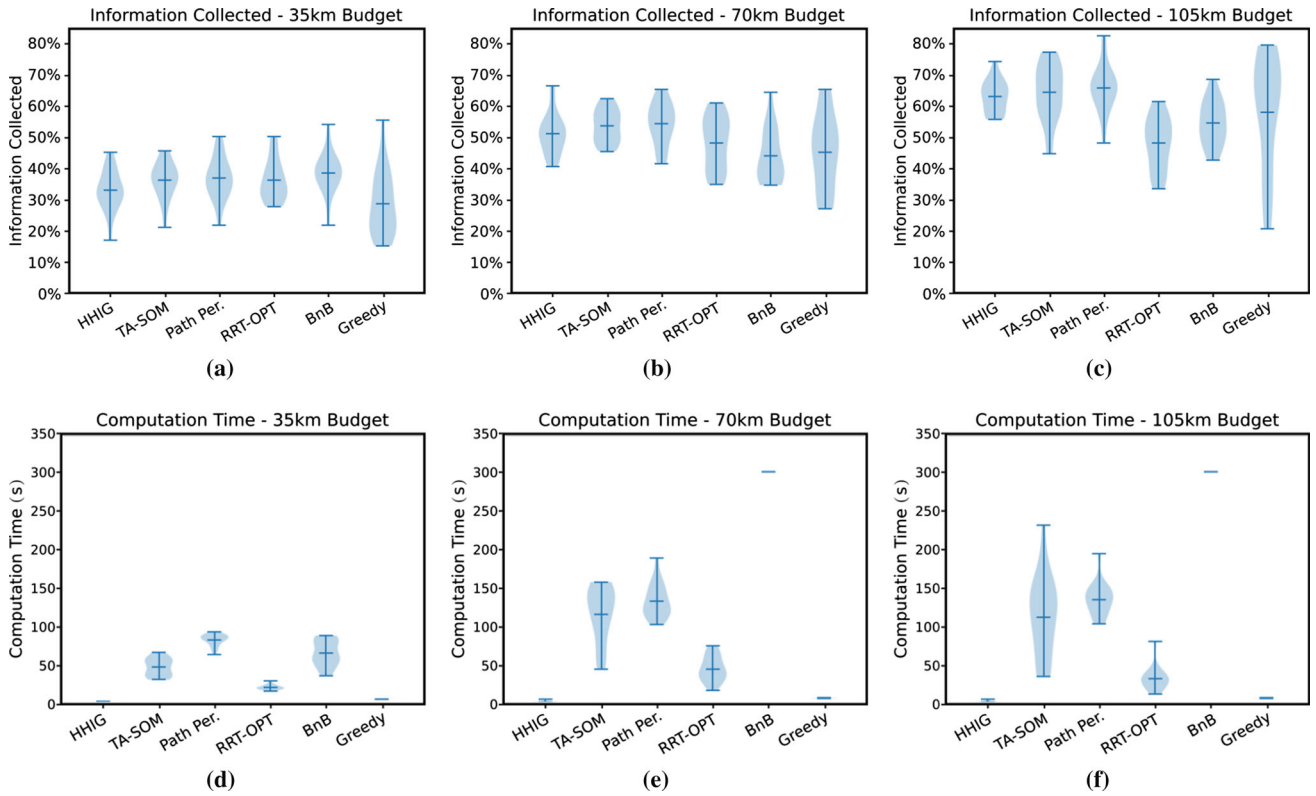


Fig. 9 Violin plots showing the percentage of information collected by a single robot (**a**–**c**) and computation time required for planning (**d**–**f**) for mission budgets of 35 km, 70 km, and 105 km (Color figure online)

ward 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 Figs. 10 and 11, respectively.

In terms of performance in the information gathering task, the multi-robot results mirror the single-robot results. Generally, both ISTP 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 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 s time limit for all team sizes, the Path Persistence quickly scales to the maximum computation time. This highlights a key advantage of our TA-SOM algorithm over the Path Persistence method. 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 TA-SOM the computational expense of the first step is largely independent of the number of robots, since all robots can use the same trained self organizing map. Homotopy information, then, needs to only be computed once for each robot. In contrast the Path Persistence

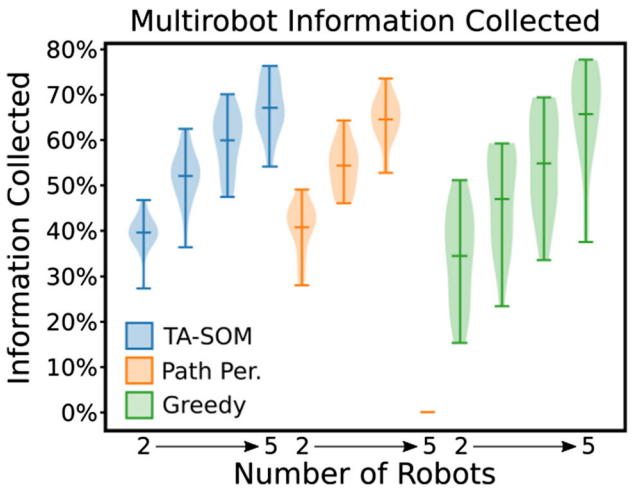


Fig. 10 Multi-robot Score. With a team size of 5 robots, the Path Persistence method failed to produce a plan in all 20 of the trials, resulting in an average score of 0 (Color figure online)

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, and therefore achieves a score of zero.

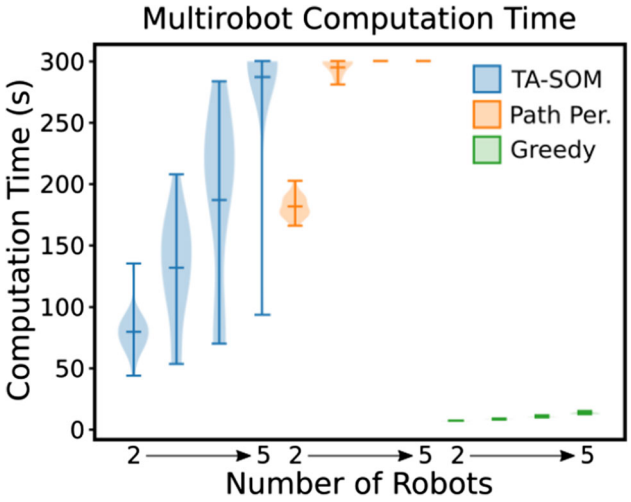


Fig. 11 Multi-robot computation time. As the number of robots increases, TA-SOM scales to the maximum computation time more efficiently (Color figure online)

6.5 Field testing

To validate the performance of our topological planning algorithms, we ran experiments using a Platypus Lutra autonomous boat (Platypus, LLC 2014), shown in Fig. 12, at Ireland Lane Pond, near Corvallis, Oregon. We tested four different algorithms, our HHIG algorithm, our TA-SOM algorithm, along with the Path Persistence variant and Greedy algorithm, 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, 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 1, and the best-performing trajectories, those from the northern deployment location are shown in Fig. 15. 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 Figs. 13 and 14. Qualitatively, both representations effectively capture the underlying structure of the information 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. 15. The behaviors shown 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.

Fig. 12 Platypus Lutra autonomous boat with lowrance depth sonar used in field trial experiments (Platypus, LLC 2014) (Color figure online)

Table 1 Percentage of total information collected by the robot at each of three starting locations

	Central (%)	Southeast (%)	North (%)
HHIG	19.88	18.56	20.21
TA-SOM	**22.56**	19.62	**22.51**
Path Per.	22.23	**21.85**	21.23
Greedy	16.97	15.43	17.63

Bold numbers indicate the best performance at each starting location

7 Conclusion

In this paper, we have introduced the idea of Information Space Topological Planning, and presented two algorithms that use different topological paradigms to incorporate information about the global structure of an information field to improve the performance of a robot in the Informative Path Planning Problem. In our Hierarchical Hotspot Information Gathering algorithm, we showed that, we can produce a high level discrete model of the information distribution by iden-

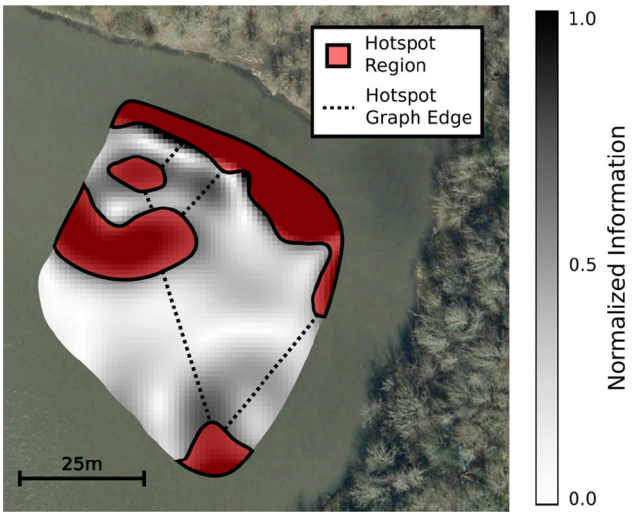


Fig. 13 Hotspots on Ireland Ln Pond (Color figure online)

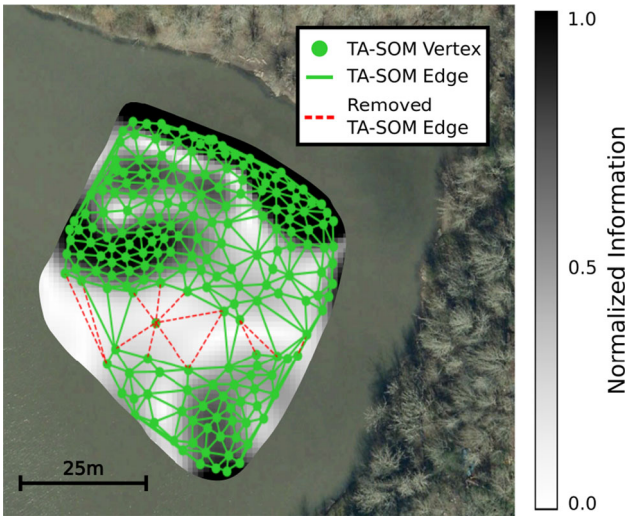


Fig. 14 Trained TA-SOM on Ireland Ln Pond (Color figure online)

tifying information hotspots in an environment. Using this graph, we can schedule a robot's time among the hotspots, enabling it to efficiently plan non-myopically. Our second algorithm, Topology Aware Self-Organizing Maps, method extends self-organizing maps to allow them to adapt their topology around prominent features in the environment. Using this, we 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 simulated trials, we showed that our topological methods were able to outperform methods that do not consider the topological information, since they are able to reason non-myopically about the global structure of the information field. Additionally, we compared our proposed methods for identifying topological features, and showed that when adapted to the information gathering problem, our methods maintained competitive performance, while incorporating the topological information more efficiently, requiring significantly less computation time to do so, and providing better scaling in an extension to multi-robot information gathering.

The main limitation of this work is that it assumes that the robot has prior knowledge about the information field in order to build a model of its topological structure. While in some applications, such as those where the robot can benefit from satellite data or prior surveys, this assumption is valid, it limits the applicability of the proposed methods in unknown environments. In future work, 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. Another avenue for future research is to apply these techniques in time-varying environments. Since topological

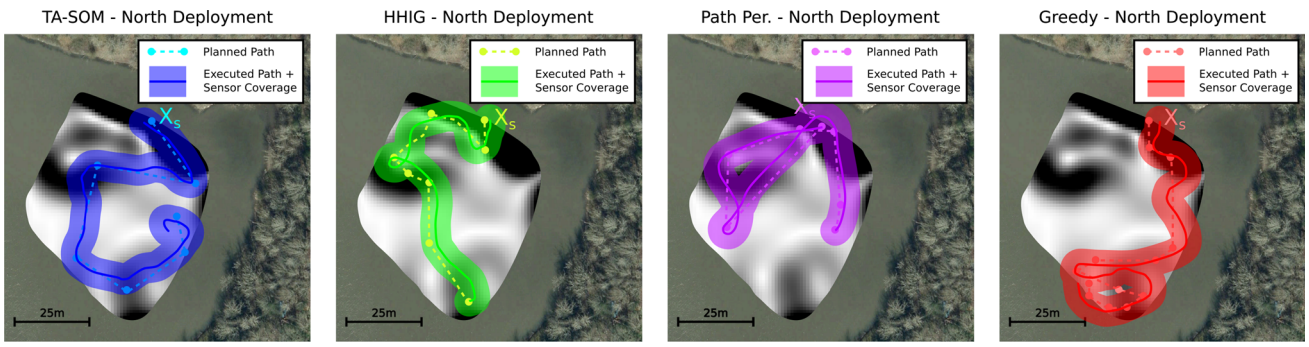


Fig. 15 Sample paths from the northern starting location. The shaded regions show the area covered by the Lutra's 5 m sensing radius as it traveled along its path. While the Greedy path (**d**) gets stuck in the southern corner after travelling down the east side of the experiment region, the TA-SOM (**a**) and HHIG (**b**) planners both visit the high information area in the west (Color figure online)

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.

References

Aurenhammer, F. (1991). Voronoi diagrams—A survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, *23*(3), 345–405.

Basener, W. F. (2006). *Topology and its applications*. New York: Wiley.

Benoit-Bird, K. J., Welch, T. P., Waluk, C. M., Barth, J. A., Wangen, I., McGill, P., et al. (2018). Equipping an underwater glider with a new echosounder to explore ocean ecosystems. *Limnology and Oceanography: Methods*, *16*(11), 734–749.

Best, G. (2019). Planning algorithms for multi-robot active perception. PhD thesis, University of Sydney.

Bhattacharya, S., Ghrist, R., & Kumar, V. (2015). Persistent homology for path planning in uncertain environments. *IEEE Transactions on Robotics*, *31*(3), 578–590.

Bhattacharya, S., Kumar, V., & Likhachev, M. (2010). Search-based path planning with homotopy class constraints. In *Proceedings of the AAAI conference on artificial intelligence*, Atlanta, Georgia, pp. 1230–1237.

Bhattacharya, S., Likhachev, M., & Kumar, V. (2012). Topological constraints in search-based robot path planning. *Autonomous Robots*, *33*(3), 273–290.

Binney, J., & Sukhatme, G. S. (2012). Branch and bound for informative path planning. In *Proceedings of the IEEE international conference on robotics and automation*, Minneapolis, Minnesota, pp. 2147–2154.

Bormann, R., Jordan, F., Li, W., Hampp, J., & Hägele, M. (2016). Room segmentation: Survey, implementation, and analysis. In *Proceedings of the IEEE international conference on robotics and automation*, Stockholm, Sweden, pp. 1019–1026.

Brunskill, E., Kollar, T., & Roy, N. (2007). 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.

Charrow, B., Kahn, G., Patil, S., Liu, S., Goldberg, K., Abbeel, P., Michael, N., & Kumar, V. (2015). Information-theoretic planning with trajectory optimization for dense 3d mapping. In *Proceedings of robotics: Science and systems*, Rome, Italy, Vol. 11.

Edelsbrunner, H., Letscher, D., & Zomorodian, A. (2000). Topological persistence and simplification. In *Proceedings 41st annual symposium on foundations of computer science*, IEEE, pp. 454–463.

Edelsbrunner, H., Morozov, D., & Pascucci, V. (2006). Persistence-sensitive simplification functions on 2-manifolds. In *Proceedings of the twenty-second annual symposium on computational geometry,* pp. 127–134.

Everett, H, I. I. I. (1963). Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, *11*(3), 399–417.

Faigl, J., & Hollinger, G. A. (2017). Autonomous data collection using a self-organizing map. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(5), 1703–1715.

Faigl, J., Kulich, M., Vonásek, V., & Přeučil, L. (2011). An application of the self-organizing map in the non-Euclidean traveling salesman problem. *Neurocomputing*, *74*(5), 671–679.

Faigl, J., Pěnička, R., & Best, G. (2016). 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.

Friedman, S., Pasula, H., & Fox, D. (2007). Voronoi random fields: Extracting topological structure of indoor environments via place labeling. In *Proceedings of the international joint conference on artificial intelligence*, Vol. 7, Hyderabad, India, pp. 2109–2114.

Hollinger, G. A., & Sukhatme, G. S. (2014). Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research*, *33*(9), 1271–1287.

Huyer, A. (1983). Coastal upwelling in the California current system. *Progress in Oceanography*, *12*(3), 259–284.

Ji, G., Shen, H.-W., & Wenger, R. (2003). Volume tracking using higher dimensional iso surfacing. In *Proceedings of the IEEE computer society visualization conference*, Seattle, Washington, pp. 28–36.

Jones, D., & Hollinger, G. A. (2017). Planning energy-efficient trajectories in strong disturbances. *IEEE Robotics and Automation Letters*, *2*(4), 2080–2087.

Jones, D., Kuhlman, M. J., Sofge, D. A., Gupta, S. K., & Hollinger, G. A. (2018). Stochastic optimization for autonomous vehicles with limited control authority. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*, Madrid, Spain, pp. 2395–2401.

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., & Schaal, S. (2011). STOMP: Stochastic trajectory optimization for motion planning. In *Proceedings of the IEEE international conference on robotics and automation*, Shanghai, China, pp. 4569–4574.

Kim, S., Bhattacharya, S., Ghrist, R., & Kumar, V. (2013). Topological exploration of unknown and partially known environments. In *2013 IEEE/RSJ international conference on intelligent robots and systems*, Tokyo, Japan, pp. 3851–3858.

Kiwiel, K. C. (2001). Convergence and efficiency of subgradient methods for quasiconvex minimization. *Mathematical Programming*, *90*(1), 1–25.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, *78*(9), 1464–1480.

Kostavelis, I., Charalampous, K., Gasteratos, A., & Tsotsos, J. K. (2016). Robot navigation via spatial and temporal coherent semantic maps. *Engineering Applications of Artificial Intelligence*, *48*, 173–187.

Kularatne, D., Bhattacharya, S., & Hsieh, M. A. (2018). Going with the flow: A graph based approach to optimal path planning in general flows. *Autonomous Robots*, *42*(7), 1369–1387.

LaValle, S. M. (1998). *Rapidly-exploring random trees: A new tool for path planning*. Princeton: Citeseer.

Lukasczyk, J., Maciejewski, R., Garth, C., & Hagen, H. (2015). Understanding hotspots: A topological visual analytics approach. In *Proceedings of the SIGSPATIAL international conference on advances in geographic information systems*, ACM, Seattle, Washington, pp. 36–46.

Luperto, M., & Amigoni, F. (2019). Predicting the global structure of indoor environments: A constructive machine learning approach. *Autonomous Robots*, *43*(4), 813–835.

McCammon, S., & Hollinger, G. A. (2017). 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.

McCammon, S., & Hollinger, G. A. (2018). 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.

McCammon, S., Jones, D., & Hollinger, G. A. (2020). 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.

McNamara, T. P. (1986). Mental representations of spatial relations. *Cognitive Psychology*, *18*(1), 87–121.

Michini, M., Hsieh, M. A., Forgoston, E., & Schwartz, I. B. (2014). Robotic tracking of coherent structures in flows. *IEEE Transactions on Robotics*, *30*(3), 593–603.
Oßwald, S., Bennewitz, M., Burgard, W., & Stachniss, C. (2016). Speeding-up robot exploration by exploiting background information. *IEEE Robotics and Automation Letters*, *1*(2), 716–723.
Petres, C., et al. (2007). Path planning for autonomous underwater vehicles. *IEEE Transactions on Robotics*, *23*(2), 331–341.
Platypus, L. L. C. (2014). *The Lutra Prop*.
Pokorny, F. T., Hawasly, M., & Ramamoorthy, S. (2016). Topological trajectory classification with filtrations of simplicial complexes and persistent homology. *The International Journal of Robotics Research*, *35*(1–3), 204–223.
Popović, M., Hitz, G., Nieto, J., Sa, I., Siegwart, R., & Galceran, E. (2017). Online informative path planning for active classification using UAVS. In *Proceedings of the IEEE international conference on robotics and automation*, Singapore, pp. 5753–5758.
Rinne, H. (2008). *The Weibull distribution: A handbook*. Boca Raton: CRC Press.
Saroya, M., Best, G., & Hollinger, G. A. (2020). Online exploration of tunnel networks leveraging topological cnn-based world predictions. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*, Las Vegas, Nevada (Virtual).
Sethian, J. A. (1999). *Level set methods and fast marching methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science* (Vol. 3). Cambridge: Cambridge University Press.
Shchepetkin, A. F., & McWilliams, J. C. (2005). The regional oceanic modeling system (ROMS): A split-explicit, free-surface, topography-following-coordinate oceanic model. *Ocean Modelling*, *9*(4), 347–404.
Singh, A., Krause, A., Guestrin, C., Kaiser, W. J., & Batalin, M. A. (2007). Efficient planning of informative paths for multiple robots. In *Proceedings of the international joint conference on artificial intelligence*, Vol. 7, pp. 2204–2211.
Somhom, S., Modares, A., & Enkawa, T. (1997). A self-organising model for the travelling salesman problem. *Journal of the Operational Research Society*, *48*(9), 919–928.
The GUDHI Project. (2014). *GUDHI user and reference manual*.
Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, *99*(1), 21–71.
Topp, E. A., & Christensen, H. I. (2006). Topological modelling for human augmented mapping. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*, Beijing, China, pp. 2257–2263.
Yu, J., Aslam, J., Karaman, S., & Rus, D. (2015). 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.
Yu, J., Schwager, M., & Rus, D. (2016). Correlated orienteering problem and its application to persistent monitoring tasks. *IEEE Transactions on Robotics*, *32*(5), 1106–1118.
Zivkovic, Z., Bakker, B., & Krose, B. (2006). 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.
Zomorodian, A., & Carlsson, G. (2005). Computing persistent homology. *Discrete & Computational Geometry*, *33*(2), 249–274.

Seth McCammon is a Postdoctoral Scholar in the Applied Ocean Physics and Engineering (AOPE) Department at Woods Hole Oceanographic Institution. He completed his Ph.D. in Robotics at Oregon State University (2021). He received his B.S. in Computer Science with an emphasis in AI and Machine Learning at Northwestern Universiy (2015). His research interests include using topological techniques to enable robots to reason about the environments they operate within.

Geoffrey Hollinger is an Associate Professor in the Collaborative Robotics and Intelligent Systems (CoRIS) Institute at Oregon State University. He has previously held research positions at the University of Southern California, Intel Research Pittsburgh, University of Pennsylvania's GRASP Laboratory, and NASA's Marshall Space Flight Center. He received his Ph.D. (2010) and M.S. (2007) in Robotics from Carnegie Mellon University and his B.S. in General Engineering along with his B.A. in Philosophy from Swarthmore College (2005). He is a recipient of the ONR YIP award (2017) and the NSF CAREER award (2019).