Topological Hotspot Identification for Informative Path Planning with a Marine Robot

Seth McCammon and Geoffrey A. Hollinger

***Abstract*— In this work, we present a novel method for constructing a topological map of biological hotspots in an aquatic environment using a Fast Marching-based Voronoi segmentation. Using this topological map, we develop a closed form solution to the scheduling problem for any single path through the graph. Searching over the space of all paths allows us to compute a maximally informative path that traverses a subset of the hotspots, given some budget. Using a greedy-coverage algorithm we can then compute an informative path. We evaluate our method in a set of simulated trials, both with randomly generated environments and a real-world environment. In these trials, we show that our method produces a topological graph which more accurately captures features in the environment than standard thresholding techniques. Additionally, We show that our method can improve the performance of a greedy-coverage algorithm in the informative path planning problem by guiding it to different informative areas to help it escape from local maxima.**

I. Introduction

Monitoring biological activity in the ocean is one of the key challenges in oceanography. Scientists are interested in how ocean processes like hypoxia, changing ocean temperatures, and chemical spills affect marine life. Typically, the most informative places to monitor these effects are in areas of high concentration of biological activity, known as hotspots. These hotspots often occur in areas where there is a confluence of cold, nutrient-rich water drawn up from the ocean floor, and warm surface water, such as along a coastal upwelling. In order to gather data, scientists want to monitor these places for long periods of time, such as weeks or months. However, traditional methods for collecting data, such as chartering a research vessel, can be prohibitively expensive, with operating costs upwards of $30,000 per day [1]. Employing autonomous robotic systems capable of conducting these monitoring tasks can not only reduce costs, but also increase the amount and quality of data collected.

One popular type of system for ocean monitoring tasks is buoyancy-driven ocean gliders such as the Spray [2], and the Slocum Glider [3], which can be deployed for weeks without needing to be recharged due to their low energy consumption. These gliders have been successfully used in long-term monitoring tasks tracking upwelling fronts [4], and monitoring harmful algae blooms [5]. While gliders are less costly than long periods of research vessel time, they are still expensive systems, costing upwards of $100,000. In order to maximize the utility of the data that the gliders collect, scientists want to deploy them only in regions where they will collect large amounts of useful data, such as in hotspot regions. Fig. 1 shows a sample area off the coast of Monterey, CA. By identifying the hotspots in this area, we can more effectively use our limited glider deployment time to collect ocean data.

This work was funded in part by support from a W. M. Keck Foundation grant to J. Barth, K. Benoit-Bird, and G. Hollinger and NSF grant IIS-1723924.

S. McCammon and G. Hollinger are with the Collaborative Robotics and Intelligent Systems (CoRIS) Institute, Oregon State University, Corvallis, OR. (e-mail: {mccammos, geoff.hollinger}@oregonstate.edu)

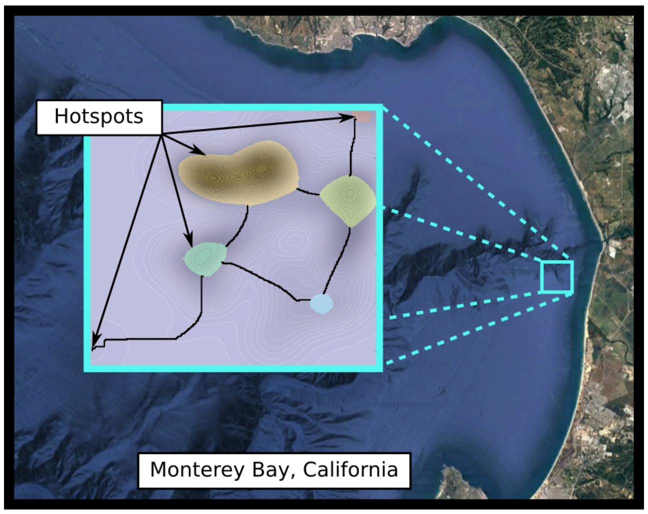


Fig. 1: A topological graph constructed using our method on a dataset collected in Monterey Bay, CA. Each colored region corresponds to a unique biological hotspot. The edges connecting them are shown in black. Figures in this paper are best viewed in color.

To improve the capability of these autonomous systems to explore and monitor their environment, we are interested in leveraging topological techniques to create a compact representation of the environment. This is accomplished by identifying elements within the environment that share key features and segmenting the environment based on these features. A topological representation can then be used to make high-level decisions about useful regions to gather data more efficiently than an exhaustive search of the space, since a search over a topological graph scales with the complexity of the graph rather than with the size and dimensionality of the space.

In this paper, we present a novel topological hotspot monitoring method, which leverages a modified Fast Marching (FM) algorithm. We developed this method to identify biological hotspot regions where useful observations are likely to be made and to construct a compact topological graph of these regions, which captures the underlying structure of the information likelihood field. The vertices of the graph correspond to the hotspot regions, and the edges are show their connectivity. In order to find the optimal allocation of time to the edges and vertices of this graph, we developed a closed-form solution, which uses Lagrange Multipliers [6] to solve the time allocation problem for a given set of vertices and edges. By searching over the space of all possible paths through the graph, we can identify the optimal schedule for the robot, which is then combined with a coverage algorithm to produce the robot's final path.

The remainder of the paper is organized as follows. Section II discusses prior work in the areas of hotspot identification and informative path planning. In Section III, we outline the three steps of our method to construct a topological representation of the environment and use it to solve the informative path planning problem. Then, in Section IV, we present the results from a set of tests using both real-world ocean data collected in Monterey Bay and simulated environments. In these tests, we show that by identifying hotspots that are likely to provide useful observations, we can improve the performance of a naïve greedy coverage algorithm.

II. Related Work

A. Topological Segmentation Techniques

Topological mapping techniques have been used to derive abstract representations of environments from metric maps. A significant body of work in this area focuses on segmenting indoor environments into regions which correspond with rooms and hallways [7], [8]. These methods leverage Voronoi partitioning [9], which segments an area by creating a set of regions based on a set of seed points, then assigning each location in space to one of these regions based on which seed point the location is closest to. In indoor environments, walls and other obstacles provide clear boundaries and seed points for the expansion of Voronoi regions. In environments with less structure, such as marine environments, it becomes more difficult to divide the space into meaningful regions. To address this, we propose leveraging Fast Marching (FM) techniques [10] to incorporate the the features of the marine environment in the Voronoi expansion process. Since they approximate the propagation of a wavefront through a space, FM techniques have been successfully employed in identify the time of first arrival of a monotonically advancing front over a cost field, as well as in planning continuous, minimum-cost paths for underwater vehicles [11].

One widely used method to create a topological representation of such a space is using some global metric to identify coherent regions within the space that share key features. There are a variety of segmentation techniques which can be used to do this. In the area of data visualization and interpretation, hotspot identification and tracking approaches use thresholding to isolate areas of interest. All regions with value greater than some threshold are included in hotspot regions, creating areas defined by isobars in $\mathbb{R}^2$ and isosurfaces in $\mathbb{R}^3$ and higher dimensional spaces [12], [13]. However, these thresholding approaches require hand-tuning the threshold parameter which, in turn, requires a significant amount of domain knowledge in order to select the correct value. We seek to improve upon these methods by substituting information contained within the environment for the advanced domain knowledge to develop a partitioning that captures the hotspots in a region, while also identifying locally relevant features that would go unnoticed by a naïve thresholding.

B. Informative Path Planning

Often, a sensing robot will need to plan a path which maximizes an information objective function, subject to budgeting constraints such as battery life or travel time. This problem is known as the informative path planning problem, and it is a well-studied problem in robotics. The informative path planning problem can be summarized as computing an optimal trajectory $\mathcal{P}^*$, which satisfies the following:

$$\mathcal{P}^* = \underset{\mathcal{P} \in \Psi}{\operatorname{argmax}} \{I(\mathcal{P})\} \; s.t. \; c(\mathcal{P}) \leq B, \tag{1}$$

where Ψ is the set of all possible trajectories, $I(\mathcal{P})$ is the information value of a trajectory $\mathcal{P}$, $c(\mathcal{P})$ is the cost of the trajectory, and B is a budget, such as one imposed by time or energy. Computing P^* is NP-Hard [14] for nearly all relevant objective functions, since the cost of searching over ψ typically scales combinatorially, both in B and in the size of the environment, and quickly becomes intractable.

Early work in this area uses Recursive-Greedy algorithms, which leverage the submodularity of information to achieve performance bounds [15]. More recent work used Branch and Bound techniques [16] to compute optimal solutions. Mixed Integer Programming formulations have been used to compute maximally informative paths over a finite set of discrete sampling locations, adapting the informative path planning problem into the orienteering problem [17]. The orienteering problem has been extended to incorporate planning paths with time-varying amounts of reward at each vertex [18], as well as leveraging correlations between nearby nodes. [19]. Computing exact solutions to the informative path planning and orienteering problems can be expensive, since they scale exponentially in the number of sensing locations. To address this, sampling-based information gathering algorithms have been developed [14]. Closely intertwined with the Informative Path Planning Problem is the problem of Adaptive Sampling, where a robot needs to identify locations in the world which will provide it with the best observations. Recent work in this area has utilized Fast Marching and Gaussian Processes [20] to efficiently select good observation locations. However many of these information gathering algorithms are limited in the size of the environments and robot budgets that they can consider. By leveraging topological planning techniques, we can identify hotspot regions within an environment that are likely to contain areas of high information, and use these to reduce the overall search space to a more tractable one.

III. Method

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

1) Identify hotspots in the environment and construct a compact topological representation of these hotspots and their connectivity as a graph.
2) Plan a maximally informative path through these hotspots, deciding which hotspots are worth visiting, scheduling an amount of time to spend at each of them, and deciding which edges to use to travel between the chosen hotspots.
3) Transform this high-level plan over the graph into one which can be executed on a vehicle by creating subplans within each hotspot.

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

A. 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 which preserves their underlying topology. Using the robot's estimate of the likelihood of making a positive observation of a phenomenon at a particular location in the environment, $I_{global} : \mathbb{R}^2 \to [0, 1]$, we will construct a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ that captures the underlying topology of I_{global}.

Each $v_i \in \mathcal{V} = \{v_1, v_2, ..., v_n\}$ is a region in space containing one or more points of interest, which we define as local maxima of $I_{global}(x, y)$. In an ocean monitoring task, a relative increase in the occurrence of a phenomena, such as phytoplankton can provide valuable data about the causes of larger oceanic trends. Each v_i has a corresponding estimate of its local reward function, $\hat{I}_i(t_i)$, where t_i is the amount of time spent at v_i. This estimate can be any nondecreasing differentiable function, and in this work we choose to model it on the exponential reward function defined in [18] 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 given by b_i and is a function of both the size of the hotspot, A_i, and the sensing radius of the robot, obs_r:

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

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

The vertices of $\mathcal{G}$ are connected by a set of edges, $\mathcal{E} = \{e_1, e_2, ..., e_m\}$, where each $e_i \in \mathcal{E}$ consists of a pair of opposite directed edges $\{\overrightarrow{e_i}, \overleftarrow{e_i}\}$. It is possible for a pair of vertices to be connected by more than one edge. A robot can only observe the information associated with a given edge, e_i once, by traversing it in either direction (by traversing either $\overrightarrow{e_i}$ or $\overleftarrow{e_i}$). The 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 section III-B.

To construct this graph, we begin by creating a discrete approximation of the global information function by sampling it in a regular grid pattern. Using this discrete approximation of I_{global}, we collect the local maxima and minima into two sets $\mathcal{S}_{max}$, and $\mathcal{S}_{min}$, respectively. $\mathcal{S} = \mathcal{S}_{max} \cap \mathcal{S}_{min}$. The elements of $\mathcal{S}_{max}$ are Points of Interest (PoI)s, as they represent locations where there is a relative increase in the global utility function. Conversely the elements of $\mathcal{S}_{min}$ are locations where there is a relative lack of the desired phenomena, and therefore should be avoided. This is shown in Fig 2a.

Once $\mathcal{S}$ is constructed, it is used as the seed points for our modified Fast Marching expansion method. As outlined in [10] and [11], the standard FM algorithm approximates a solution to the Eikonal Equation:

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

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

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

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

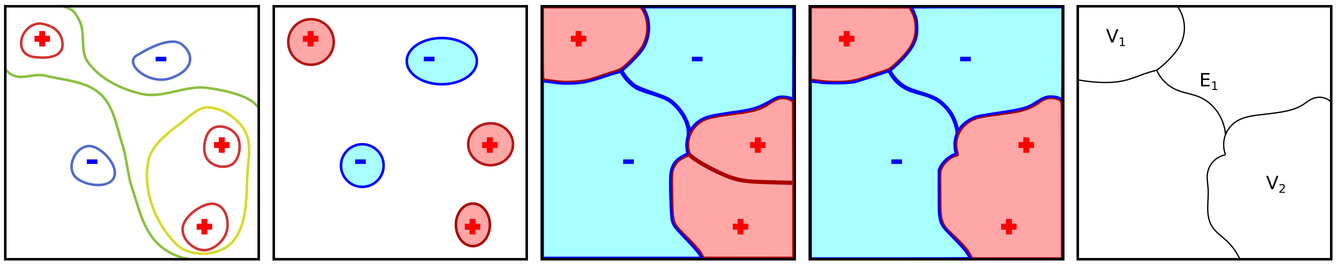
$$D_{i,j}^{+x} = \frac{u_{i+1,j} - u_{i,j}}{h},\ D_{i,j}^{-x} = \frac{u_{i,j} - u_{i-1,j}}{h},$$

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

The upwind scheme uses a breadth-first update method to iteratively select a trial point which 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 necessary.

If a neighbor, $x_{i,j}$ is adjacent to one point or one pair of opposite points in *accepted*, termed P_1, then the time-of-first-arrival at $x_{i,j}$, $u_{i,j}$ is updated according to:

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



(a) (b) (c) (d) (e)

Fig. 2: Hotspot Identification Process. Fig 2a shows the selected maxima and minima. Figs 2b and 2c show the growth of the labelled regions around each s_{max} and $s_{min} \in \mathcal{S}$ Fig 2d shows the merging of adjacent maxima regions. Finally, Fig 2e shows the labelling of each maxima region as a vertex, and each interface between minima regions as edges.

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

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

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

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

We adapt this standard FM formulation by varying $\tau_{i,j}$ based on whether $x_{i,j}$ is a descendant of a member of $\mathcal{S}_{max}$ or $\mathcal{S}_{min}$. We 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 $1 - I_{global}(i,j)$. For an $x_{i,j}$ which descends from a $s_{min} \in \mathcal{S}_{min}$, $\tau_{i,j} = I_{global}(i,j)$. 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 Figs. 2b and 2c.

To construct a graph from the labelled regions, we merge adjacent regions grown from maxima, as depicted in Fig. 2d, 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 I_{global}, and therefore correspond to relatively information-rich paths between two vertices. The resultant graph for a sample environment is shown in Fig. 2e.

It is possible for environments to exist where the topological graph construction fails, such as when a hotspot is completely enclosed by a single region grown from a local minima, or a local minima is enclosed by a hotspot. In the first case, we simply use Fast Marching to extend an edge from the isolated hotspot to the nearest edge or hotspot, connecting it to the graph. We do not address the situation where a hotspot encloses one or more local minima, as it has no effect on the topological structure of the resulting graph; we simply end up with a hotspot that bounds one or more areas that are not included in the hotspot. If a particular domain requires that hotspots be solid, a simplex-based method such as the one employed in [21] 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.

B. 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 [19], [18], the problem that we seek to solve is to identify an informative path, $\mathcal{P} = (\mathcal{V}_\mathcal{P}, \mathcal{E}_\mathcal{P}, \mathcal{T})$, where $\mathcal{V}_\mathcal{P} \subset \mathcal{V}$ is the set of unique vertices visited along the path, $\mathcal{E}_\mathcal{P} \subset \mathcal{E}$ is the set of edges that the robot traverses, and $\mathcal{T}$ is the set of times, t_i, spent at each $v_i \in \mathcal{V}_\mathcal{P}$. However, in our approach, we do not restrict the path that the robot follows to be a tour. Instead, the robot can begin and end its path at any vertex.

We begin the scheduling process by constructing a tree with its root at the vertex of $\mathcal{G}$ corresponding to the robot's initial position. We 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,

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

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 the graphs we develop are relatively uncomplicated, rarely consisting of more than 10 vertices. However, graphs with a particularly high branching factor can lead to an intractable number of paths. Additionally we are able to prune paths which 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

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

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 $t_1^+, t_2^+, ..., t_{|\mathcal{V}_\mathcal{P}|}^+$ which maximize

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

using a Lagrange Multiplier Method [6] to solve the resource-constraint problem inherent in allocating R among $\{t_1^+, t_2^+, ..., t_{|\mathcal{V}_\mathcal{P}|}^+\}$. We construct our Lagrange Function, $\mathcal{L}$, using the Lagrange Multiplier variable, λ,

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

We then take the partial derivative with respect to each $t_i^+ \in \{t_1^+, t_2^+, ..., t_{|\mathcal{V}_\mathcal{P}|}^+\}$, as well as λ. Setting these equal to 0 yields the following system of equations:

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

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

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 Equation 9 equal to each other:

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

Plugging this back in to Eq. 10, we are left with

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

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

Taken together, Equations 11 and 12 can be used to compute the optimal values for all $\{t_1^+, t_2^+, ..., t_{|\mathcal{V}_\mathcal{P}|}^+\}$ along a given path.

The process used to calculate the schedule for the robot is outlined in Algorithm 1. Since each node in the tree corresponds to a unique path through the graph, for each node in the tree we can recover this candidate path by re-tracing the path through the tree from the node to the root. By identifying the set of vertices and edges visited along this path, we can form an instance of the scheduling problem. By solving this problem for each unique path, we can select the $\mathcal{P}$ which maximizes our expected reward, thus resulting in the optimal path for the robot.

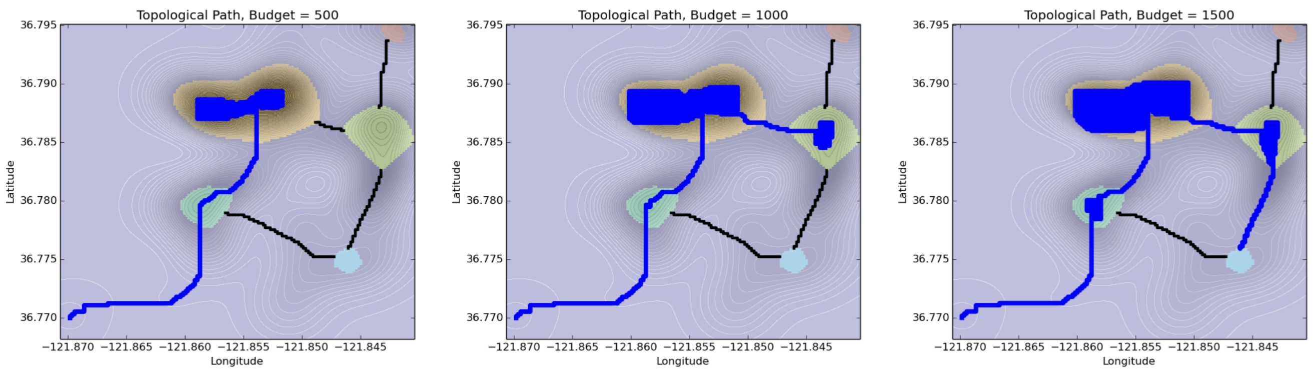
Algorithm 1 Hotspot Scheduling

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

C. Path Planning

Once the optimal schedule for the graph is computed, and the budget for each node is assigned, the robot needs to plan a path that uses the assigned budget within each node. We use a greedy-coverage algorithm to quickly compute a coverage path for the vertex. This works well, since the topological hotspot identification and segmentation component of our approach identifies areas which are filled with only high-information areas, making a simple coverage approach more effective than it would otherwise be. However, more sophisticated planners such as Branch and Bound [16] or stochastic trajectory optimizers such as STOMP [22] may be considered to compute more optimal coverage paths at the expense of 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.



(a) (b) (c)

Fig. 3: Informative paths planned using budgets of 3a 500, 3b 1000, and 3c 1500. 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.

By combining the paths along the selected edges with the greedy-coverage paths within the vertices, we obtain the final path for the robot. Some sample paths created on our Monterey Bay dataset are shown in Fig. 3.

IV. Results

We evaluated our method both on a set of simulated environments and on a dataset of ocean bioacoustic activity collected using a Slocum glider [3] in Monterey Bay, CA. The simulated environments were created by taking the sum of 50 Gaussian functions randomly distributed in a 4-connected 100x100 world and normalizing the value in each cell to be between 0 and 1. The information value in each cell corresponds to the likelihood of making a useful observation, such as observing the presence of marine life, in that cell. Making an observation decreases the likelihood of making useful observations within each cell inside the vehicle's sensor footprint. The robot is able to move through this 4-connected world at a rate of 1 cell per unit time.

A. *Hotspot Segmentation*

To evaluate the effectiveness of our hotspot segmentation approach, we compared our method of segmentation to standard thresholding, which labels as hotspots each point in space greater than a threshold. We compare to a static threshold of 0.5, as well as an adaptive threshold, set to capture all PoIs. This is done by setting the threshold value equal to the value of the lowest PoI. Each thresholding method is demonstrated on the Monterey, CA environment in Fig. 4. A visual examination of the three segmentation techniques highlights the drawbacks of the two thresholding approaches to segmentation. Using a constant threshold, as in [12], and [13], set at 0.5 is shown in Fig. 4c fails to capture a number of the PoIs, while adjusting this threshold to capture the lowest-valued PoI (at .05) collects nearly all of the the environment into one giant hotspot.

We compared the three segmentation methods: Fast Marching, Static Thresholding, and Adaptive Thresholding across a set of 20 simulated environments. In each of these environments, we compared the methods on both the number of Points of Interest captured and the hotspot density, which is the percent increase in average information between the hotspots and the whole environment. The results of these trials are shown in Figs 5a and 5b, respectively. Our method shows an increase in hotspot density over the adaptive thresholding method, while maintaining 100% of Points of Interest captured in hotspot regions. While our method does produce less-dense hotspots when compared to a static threshold, the static thresholding misses a significant portion of the points of interest. This 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.

B. *Informative Path Planning*

In a set of 20 simulated trials, we compared our algorithm with a greedy-n-lookahead algorithm, where n is between 1 and 5. The greedy lookahead algorithm searches over all paths of length n, and selects the one that will allow the robot to collect the most information as it moves through the world.

We compare the results across these trials in both the amount of useful observations received by the algorithms and the computational runtime. These are shown in Figs. 6 and 7, respectively. Our method outperforms the greedy-n-lookahead algorithm for all values of n that we tested. Since our method identifies all locations which are informative and computes the information-maximizing tradeoff of time between each, it is significantly less prone to being stuck in a local maxima. Contrastingly, since the greedy algorithm is myopic, it is easily caught, and does not explore additional hotspots.

It is worth noting that the greedy-1-lookahead algorithm is identical to the greedy-coverage algorithm used to plan the sub-path within each hotspot in our topological planning method. Thus, these results show that by utilizing our method

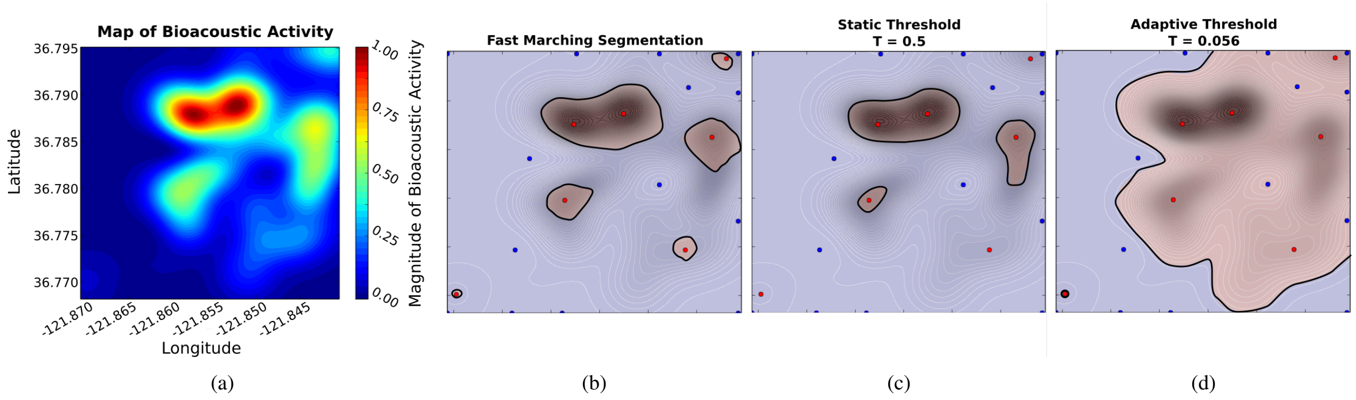


Fig. 4: Comparison of segmentation methods. Areas labeled as hotspots are shown in red. A map of bioacoustic activity is shown in Fig 4a. Fig. 4b shows our Fast Marching based method for identifying hotspots. Fig. 4c shows the hotspots identified using static thresholding (Activity > 0.5) and Fig 4d shows the hotspots identified using an adaptive threshold set to capture each Point of Interest

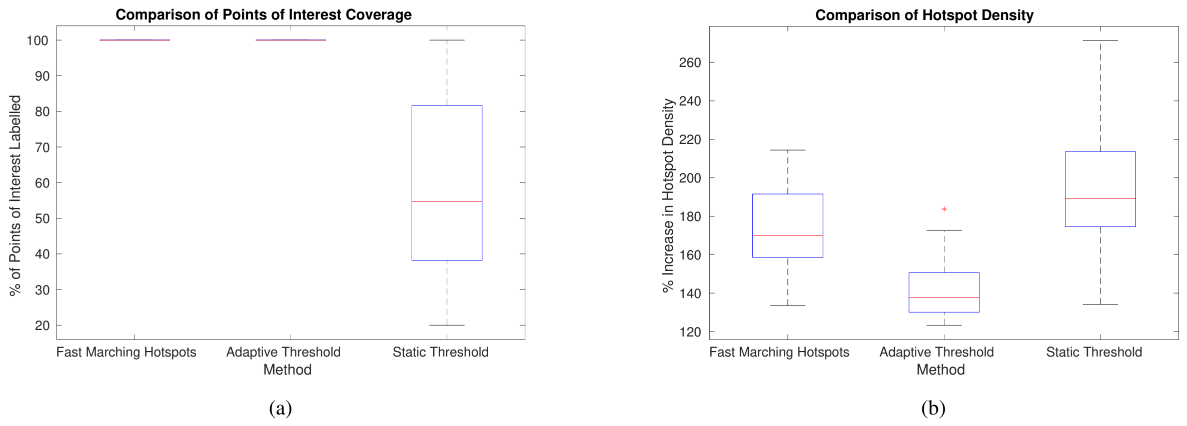


Fig. 5: Results comparing our Fast Marching Hotspot Segmentation algorithm to static thresholding (Activity$>$ 0.5) and adaptive thresholding. Both our method and adaptive thresholding capture 100% of points of interest within regions labelled as hotspots. However, our approach outperforms the adaptive thresholding in terms of hotspot density. The static thresholding method does produce denser hotspots; however it fails to capture a significant portion of the points of interest in the environment

with a naïve planner, we can vastly improve the performance of the planner by forcing it to spend its budget across many different hotspots.

As expected, our algorithm requires more computational time than the greedy-n-lookahead algorithm. Even though the greedy-5-lookahead planner examines 1024 paths at each iteration, our method has to examine far more, since it is exhaustively searching over the full topological graph for the best path.

Additionally, we attempted to compare our approach with Branch and Bound over the same information field. However, Branch and Bound was unable to terminate in under an hour on any of the 5 environments that it was attempted on, while our approach to informative path planning was able to plan an informative path on each of them. The budget that we used for planning (500 timesteps) are well beyond the ability of branch and bound techniques on modern computers.

V. Conclusion

In this paper we presented a novel approach to the Informative Path Planning problem. By leveraging the un-

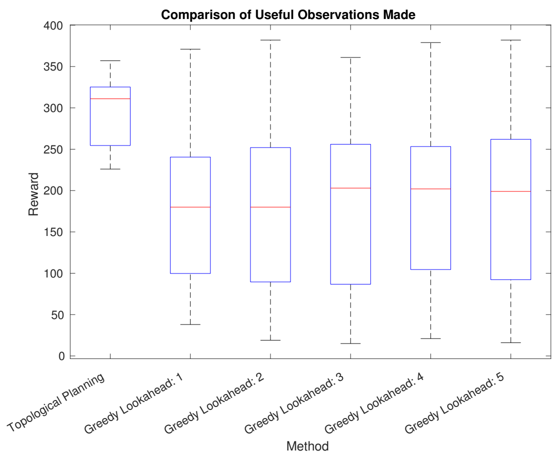


Fig. 6: Comparison of the number of useful observations made by the robot in simulated trials between our algorithm and a 1, 2, 3, 4, and 5-step lookahead greedy algorithm for a budget of 500 steps. We additionally compared our approach to a branch and bound method, however, the branch and bound method failed to produce a path within a reasonable timeframe, and so the results are not included here.

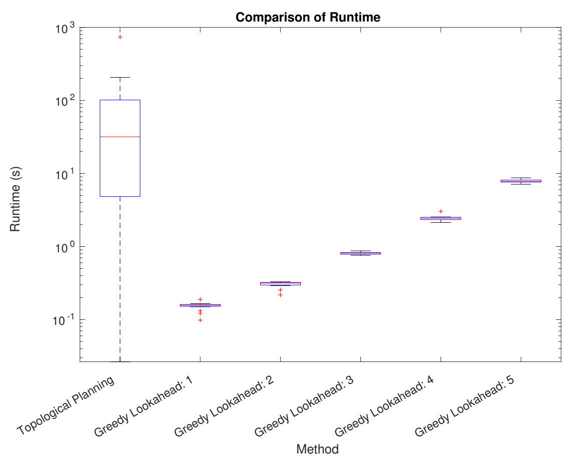


Fig. 7: Comparison of runtimes between our algorithm and a 1, 2, 3, 4, and 5-step lookahead greedy algorithm for a budget of 500 steps. Note the log scale on the time axis. We additionally compared our approach to a branch and bound method, however, the branch and bound method failed to produce a path within a reasonable timeframe, and so the results are not included here.

derlying topological structure of an information field, we are able to construct a graph which captures the structure of hotspots in an information field. Additionally, we developed a closed-form analytic solution for optimally distributing time amongst the vertices visited along a path. Using this schedule, we can improve the information collected by a greedy-coverage algorithm by guiding it to hotspots where it can be most effective.

There are many interesting research directions that remain in this area. We would like to continue to investigate the utility of our approach as a divide-and-conquer method for informative path planning. Since we segment out regions of the space which are information-sparse, we greatly reduce the search space for a planner such as branch-and-bound, and may be able to reduce it to a tractable problem. Additionally, we would like to examine more pruning techniques that we can apply to our tree of paths to allow us to search it more efficiently for the best path. Finally, we would like to investigate how the topological graph changes in a time-varying information field, and use this for planning for long-term hotspot monitoring tasks.

Acknowledgment

The authors would like to thank John A. Barth, Anatoli Erofeev, Stephen D. Pierce, and Pat Welch from Oregon State University as well as Kelly J. Benoit-Bird and Chad Waluk from the Monterey Bay Aquarium Research Institute for their contributions in designing the glider bioacoustics system used in acquiring the dataset used in this paper. Additionally, we would like to thank Jared Figurski, captain of the R/V Paragon.

References

[1] National Research Council *et al.*, *Science at Sea: Meeting Future Oceanographic Goals with a Robust Academic Research Fleet*. National Academy Press, Washington, DC, 2009.

[2] J. Sherman, R. E. Davis, W. Owens, and J. Valdes, "The autonomous underwater glider "Spray"," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 4, pp. 437–446, 2001.

[3] C. Jones, E. Creed, S. Glenn, J. Kerfoot, J. Kohut, C. Mudgal, and O. Schofield, "Slocum gliders - a component of operational oceanography," in *Proceedings of the International Symposium on Unmanned Untethered Submersible Technology, Portsmouth, NH*, 2005.

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

[5] R. N. Smith, M. Schwager, S. L. Smith, B. H. Jones, D. Rus, and G. S. Sukhatme, "Persistent ocean monitoring with underwater gliders: Adapting sampling resolution," *Journal of Field Robotics*, vol. 28, no. 5, pp. 714–741, 2011.

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

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

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

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

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

[11] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, and D. Lane, "Path planning for autonomous underwater vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 331–341, 2007.

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

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

[14] G. A. Hollinger and G. S. Sukhatme, "Sampling-based motion planning for robotic information gathering." in *Proceedings of Robotics: Science and Systems, Berlin, Germany*, vol. 3, no. 5, 2013.

[15] A. Singh, A. Krause, C. Guestrin, W. J. Kaiser, and M. A. Batalin, "Efficient planning of informative paths for multiple robots." in *Proceedings of the International Joint Conference on Artificial Intelligence*, vol. 7, 2007, pp. 2204–2211.

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

[17] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.

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

[19] J. Yu, M. Schwager, and D. Rus, "Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 342–349.

[20] N. R. Lawrance, J. J. Chung, and G. A. Hollinger, "Fast marching adaptive sampling," *Robotics and Automation Letters*, vol. 2, no. 2, pp. 696–703, 2017.

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

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