Topology-Aware Self-Organizing Maps for Robotic Information Gathering

Seth McCammon, Dylan Jones, and Geoffrey A. Hollinger

***Abstract*— In this paper, we present a novel algorithm for constructing a maximally informative path for a robot in an information gathering task. We use a Self-Organizing Map (SOM) framework to discover important topological features in the information function. Using these features, we identify a set of distinct classes of trajectories, each of which has improved convexity compared with the original function. We then leverage a Stochastic Gradient Ascent (SGA) optimization algorithm within each of these classes to optimize promising representative paths. The increased convexity leads to an improved chance of SGA finding the globally optimal path across all homotopy classes. We demonstrate our approach in three different simulated experiments. First, we show that our SOM is able to correctly learn the topological features of a gyre environment with a well-defined topology. Then, in the second set of experiments, we compare the effectiveness of our algorithm in an information gathering task across the gyre world, a set of randomly generated worlds, and a set of worlds drawn from real-world ocean model data. In these experiments our algorithm performs competitively or better than a state-of-the-art Branch and Bound while requiring significantly less computation time. Lastly, the final set of experiments show that our method scales better than the comparison methods across different planning mission sizes in real-world environments.**

I. INTRODUCTION

In a wide variety of field applications, autonomous robots are an attractive alternative to traditional manual approaches to data collection. Compared to established data collection methods, deploying robots for remote sensing offers improved access to difficult environments, often at a lower cost, which enables wider coverage with less risk to humans. One domain where robots are already widely used to collect scientific data is in marine environments. Due to biological and geophysical processes, the regions of the ocean that provide valuable scientific data are relatively sparse, with interesting activity clustering around a discrete set of features, such as along upwelling and mixing fronts or in patches of wildlife activity known as hotspots [1]. Identifying high-quality paths in such environments presents a significant challenge to optimization-based information gathering algorithms. The sparse reward function leads to difficulties in calculating useful gradients for improving paths, as well as creating a significant risk of the optimizer becoming stuck in a local maxima, and failing to identify a globally optimal path [2].

In this paper, we propose using a topological representation of the environment to decompose the space of all possible paths a robot could take into a set of topologically distinct groups of paths, called homotopy classes. By using paths from different homotopy classes as the initial paths for stochastic optimization, we can ensure better coverage of the environment than using random restarts. An overview of our proposed method is shown in Fig. 1. Previous work has already shown that considering distinct topological classes can improve path optimization in obstacle-filled environments [3]. However, these methods rely on explicitly identifiable obstacles that induce topologically distinct trajectory classes. In exploration and information gathering, particularly in the marine domain, there are few or none of these obstacles. Instead, the different classes of trajectories a robot might wish to consider are defined by the information objective function, with high-value regions separated by low information voids.

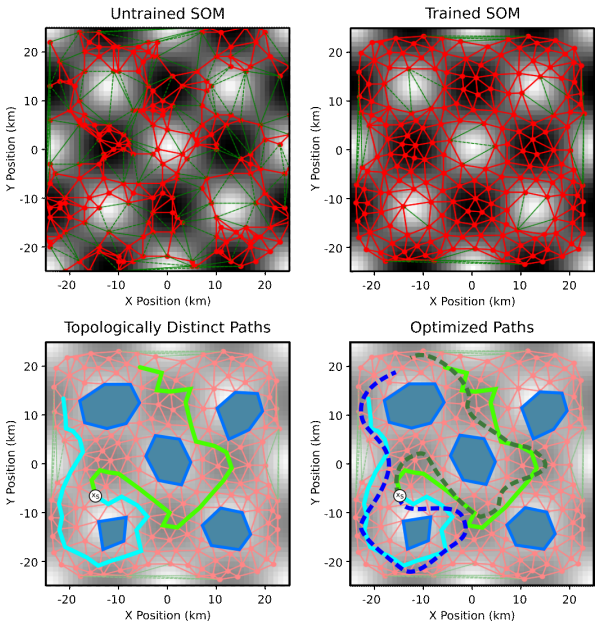


Fig. 1: Training of an SOM on a Gyre world. Darker regions contain more information. Before training (top-left), the topological features identified by the SOM (areas with green edges) are driven by the sampling noise and indistinguishable from the true topological features, (areas in white). Once trained (top-right) the identified regions align with the true features. The topology of the SOM (bottom-left) is used to identify a set of reference trajectories in unique homotopy classes. These trajectories are then optimized using Stochastic Gradient Ascent (bottom-right).

The main challenge in applying topological techniques to information gathering tasks then becomes identifying the shape, size, and connectivity of the high information regions [4]. The core of our approach is to construct a discrete graphical model to allow the computation of a set of homotopy classes. These homotopy classes then can be used to constrain a set of locally near-convex regions which paths

This work has been funded in part by the following grants: NSF IIS-1845227, NSF IIS-1723924, and ONR N00014-17-1-2581.

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

can be optimized more effectively. The primary contribution of this paper is a topology-aware Self-Organizing Map (SOM) which is able to learn the topology of an anisotropic information field. We accomplish this through the addition of a topology modification step in the SOM training algorithm. We then utilize the learned topology in conjunction with a local optimizer to search for a globally optimal path from among the learned homotopy classes.

The remainder of the paper is structured as follows. Section II covers existing work in the areas of informative path planning, gradient-based optimization, and topological path planning. Section III outlines our problem formulation and notes key assumptions made by our algorithm. Section IV provides a detailed description of our proposed algorithm. Section V contains our experimental results, including a set of experiments on randomly-generated worlds, as well as on real-world data extracted from a Regional Ocean Modelling System (ROMS) model. Finally, we conclude the paper in Section VI and discuss potential avenues for future research.

II. BACKGROUND AND RELATED WORK

A. Autonomous Information Gathering

The Informative Path Planning problem 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. These facts result in the informative path planning problem being classified as NP-Hard [5].

There have been a large number of algorithms proposed to solve the Informative Path Planning problem. Algorithms such as Greedy, Recursive Greedy [6], and Branch and Bound [7] 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 the only new information being added to the search process is that which is local to the action being made. Since the information gathering reward function is full-path dependent, the submodularity property of information gathering necessitates the re-evaluation of the entire path reward at each evaluation, making the sequential path construction approach fairly inefficient in evaluating paths in a global context. Sampling-based methods, such as Rapidly exploring Information Gathering (RIG) [5], 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 [8], 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 [9] 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 [10], and EESTO [11] 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 [12], 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 for information gathering by more rapidly incorporating global information about the shape of the distribution of information throughout the world in the planning process. In doing so we will enable more efficient calculation of high-quality paths.

B. Planning with Topological Features

Topological features provide a high-level method to describe the way that a path moves through an environment. In doing so, they provide a language that places a trajectory in a global context. Topological features have been used to plan for information gathering [4]. However, there the authors build a topological representation of the the world as a set of discrete, connected regions, rather than considering the full space of possible trajectories a robot may wish to take. This capability requires the ability to identify topologically distinct classes of trajectories. Early work in this area was 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 [13]. In their work, Bhattacharya et al. developed the H-signature, a homotopy invariant which uniquely describes a trajectory's homotopy class [14]. They used this to construct a homotopy-augmented graph, a topology-aware structure which 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.

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

Persistent homology enables identification of major topological features within an environment [15], [16]. This is

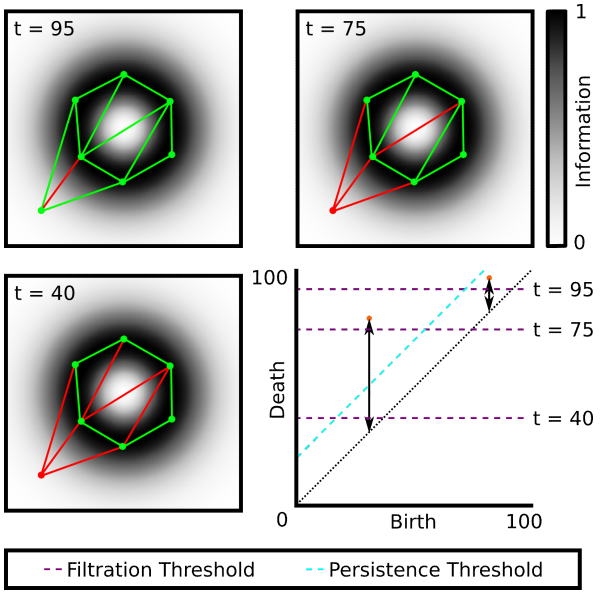


Fig. 2: An example of a persistence diagram for a simplicial complex. Using the persistence of 1^{st} order topological features (orange dots), we identify a persistence threshold (blue), classify features as 'persistent' and 'ephemeral', and set a corresponding filtration threshold (purple) which is used to alter the SOM topology. Three different filtration thresholds and their corresponding simplicial complexes are shown.

accomplished through a persistence diagram, which documents the birth and deaths of topological features on a graph. These features include connected components and holes in the graph, which are created and destroyed relative to the value of a filtration parameter. An example of a persistence diagram with two topological features is shown in Fig 2. As the filtration parameter increases along the black dashed line, connected components merge into progressively larger components, causing the death of a smaller component as it merges with a larger one. As components connect they can birth holes which will eventually die as they are covered over by the increasing connectivity of the graph. For a more in-depth discussion of persistent homology, we refer the interested reader to [16].

Existing methods that use persistent homology to identify topological features focus on identifying features which correspond with physical obstacles in the environment. We expand on this capability by using them to identify prominent features in an information field in conjunction with an SOM.

III. PROBLEM FORMULATION

A. Problem Formulation

In this paper, we consider a typical formulation for the Informative Path Planning problem [6]. We are interested in finding an optimal path for a robot, which satisfies the following equation:

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

where $\mathcal{P}$ is a path defined as a series of waypoints $(x_0, x_1, ..., x_n) \in \mathbb{R}^2$. The optimal path, $\mathcal{P}^*$, is the path from the space of all possible paths Φ which 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.

IV. METHOD

Our proposed algorithm is comprised of two main steps. The first is to identify the salient topological features in the environment, and use them to partition Φ into different homotopy classes. The second step is to select a representative path from each homotopy class and use Stochastic Gradient Ascent to optimize a path within each homotopy class.

In many previous domains where topological techniques have been utilized (e.g. [14], [16]), 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 which 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 persistent local minima as features which 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.

A. 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, $\Psi(\cdot)$ [17]. In robotics applications, SOMs have been used as a method to solve the Travelling Salesperson Problem [18], [19], as well as the TSP's common extensions, such as the Orienteering Problem [20], and other information gathering tasks [21]. 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 [22]. 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

Algorithm 1 Topology-Augmented Self-Organizing Map

function TOPOLOGYSOM($I(\cdot), N$)
  $\mathcal{V} \longleftarrow$ DrawSamples($N, I(\cdot)$)
  $\mathcal{E} \longleftarrow$ DelaunayTriangulation($\mathcal{V}$)
  $\mathcal{G} \longleftarrow (\mathcal{V}, \mathcal{E})$
  while $\neg$ stopping **do**
    $\hat{\mathcal{G}} \longleftarrow$ TrainSOM($\mathcal{G}, I(\cdot)$)
    $\mathcal{G} \longleftarrow$ Filtration($\hat{\mathcal{G}}, I(\cdot)$)
  return $\mathcal{G}$

Algorithm 2 Self-Organizing Map

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

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. Psuedocode is given in Algorithm 1. 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 moved toward v^* is based on both the Euclidean distance between v_i and v^*, as well as on a neighborhood function that maps based the graph distance along $\mathcal{G}$ (i.e. the number of edges between v_i and v^*) to the range $[0, 1]$. We used a common form for the neighborhood function:

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

where γ is a hand-tuned weighting parameter which 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

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

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 2.

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 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 [13]. 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 [23], 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 which might arise as artifacts of the triangulation process. To determine which features are ephemeral and which are persistent, we fit a Weibull distribution to the first-order persistence values. We define features with a persistence value higher than one standard deviation of the fitted Weibull as persistent, while each feature with persistence less than one standard deviation is considered ephemeral. This results in a diagonal persistence threshold, as seen in Fig. 2. 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 which maximizes the number of persistent features in existence. Then, from these, we select the value which minimizes the number of ephemeral features which exist simultaneously. Once the filtration set, we remove edges with a value greater than the filtration threshold. This process is shown in Fig. 2. Applying the filtration alters topology of the SOM to be closer to that of the underlying function, allowing it to fit the function better during subsequent training.

Once the Topology-Aware SOM is trained, it can be used to enumerate the possible homotopy classes of trajectories. To accomplish this, we use the homotopy augmented graph proposed in [14]. 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 which 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 which maximizes the objective function, $I(\cdot)$.

B. Stochastic Gradient Ascent

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

We use Stochastic Gradient Ascent (SGA) algorithm as the local optimization function, since the gradient of the information gathering objective function is difficult to calculate analytically due to the path dependence of the reward [12]. At a high level, SGA operates by estimating the gradient by sampling perturbations and recombining them using a weighting based upon the objective function. Psuedocode for the SGA algorithm is presented in Algorithm 3.

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

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

with zero mean and covariance matrix defined by σ_x and σ_y, which are the variation in the x and y directions respectively. Note that here we have 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, ϵ_i in $\boldsymbol{\epsilon}$ is independently applied to $\mathcal{P}$ at the given index to generate perturbed path $\hat{\mathcal{P}}_k$. Each of these $\hat{\mathcal{P}}_k$ is then scored using $I(\cdot)$ to generate a score vector $\mathbf{s}$. This score vector, $\mathbf{s}$, is then used in conjunction with $\boldsymbol{\epsilon}$ to calculate the update to that waypoint as:

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

where

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

Algorithm 3 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
```

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

C. Analysis

SGA is guaranteed to almost surely converge to a local maxima [24] 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. We confirm this hypothesis empirically in our comparisons with a naïvely initialized SGA method (RRT-OPT).

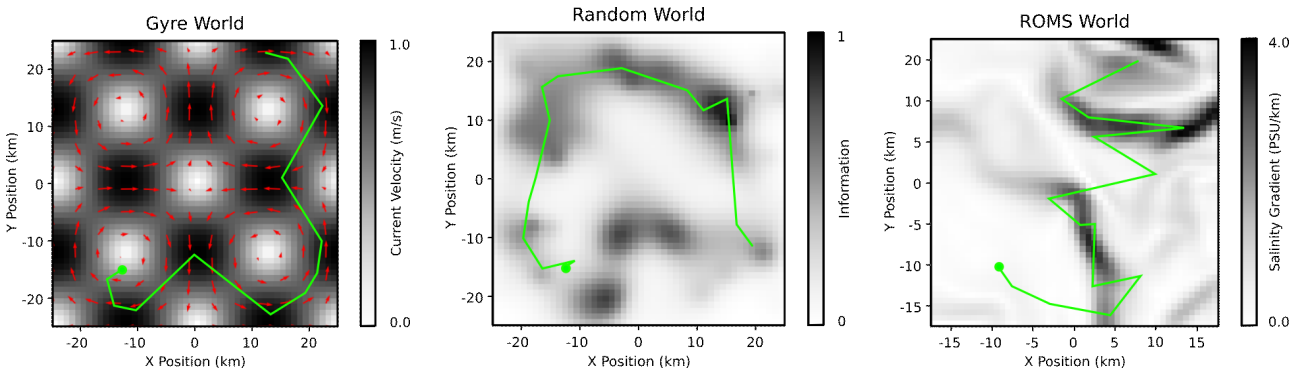
V. RESULTS

To determine the effectiveness of our proposed approach, we performed a three different sets of experiments. The first of these was to evaluate how well our topology-aware Self-Organizing Map was able to learn the underlying topological structure of a field. The second of these was to evaluate how well using topological classes improves the performance of a robot in the Informative Path Planning problem. The last was to evaluate how our approach scales across different budgets.

In our experiments, we consider the information gathering problem in the context of a marine science gathering task. We perform experiments in three different types of worlds: Gyre, Random, and a real-world environment based on Regional Ocean Modelling System (ROMS) model output which simulates ocean conditions in Monterey Bay, California[1]. [25].

The Gyre world, shown in Fig. 3a, is hand-constructed to contain a quadruple-gyre system, similar to the worlds used in [26] for planning in flows. The information function in this world is 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. The Gyre world is a 50 km by 50 km environment on a 1 km grid.

[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.



(a) Gyre World (b) Random World (c) ROMS World

Fig. 3: Examples of the three different types of worlds used for experiments with the path planned by our method. The robot's path starts at the green dot.

The Random worlds is a set of randomly generated worlds using a sum-of-Gaussians method to randomly distribute information hotspots throughout a given world. Similar to the Gyre world, the Random worlds are each 50 km by 50 km with a 1 km grid resolution.

The ROMS worlds consists of a 35 km by 35 km section in the center of Monterey Bay at 20 randomly chosen times throughout 2017. The information function was defined as the magnitude of the surface salinity gradient, a key identification marker for the localization of upwelling fronts. An example ROMS world is shown in Fig. 3c.

A. *Identifying Topology*

The first set of experiments we performed was to evaluate the ability of the SOM to learn the topology of an information field. We used the Gyre world to compare the number of features that the SOM found to the true number: five, as seen in Fig. 3a.

We constructed 20 SOMs on the Gyre world training the 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 4. At 100 vertices the SOM struggles to consistently find all of the features present in the environment. In the remainder, the SOM is able to smoothly converge to the correct number of features. Additionally, the amount of time required to train each of these SOMs is shown in Fig 5. As expected, as the number of vertices in the SOM increases the amount of time required to train the SOM increases. Based on these results, we chose to use 150 vertices and three training epochs for a balance of quality-of-fit and computation time.

B. *Planning Informative Paths*

After testing that the SOM is able to properly identify the relevant topological features in the environment, we tested the performance of the entire system against three comparison methods. **Greedy**: The greedy method first builds a Probabilistic Road Map (PRM) over the environment to build a planning graph. Then, starting from x_0, the algorithm selects the edge which maximizes $I(\cdot)$. **Branch and Bound**: We also compare to Branch and Bound for information gathering [7]. We evaluated its performance planning over a PRM. However, we found that it failed to converge to

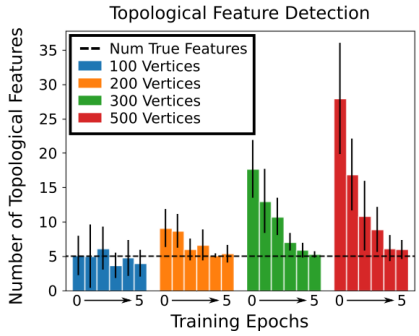


Fig. 4: Number of topological features found by the SOM in the Gyre world for different numbers of vertices across up to five training epochs. The black dashed line at five is the true number of topological features in this environment.

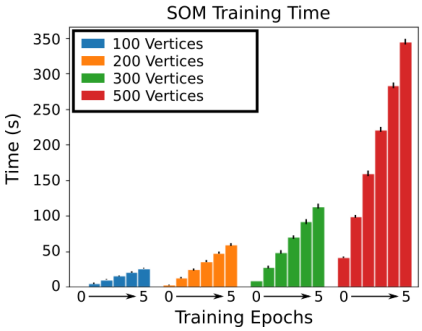


Fig. 5: Amount of time to train the SOM for different numbers of vertices and training epochs in the Gyre world.

an optimal solution for even the shortest paths, and so we also tested it using it on a uniform 4-connected grid with a resolution of 10 km. **RRT-Opt**: Lastly, we compare to a method which uses Rapidly Exploring Random Trees (RRT) to quickly build a large number of paths through the environment. Then, the top five scoring paths to leaf nodes are optimized by SGA. RRT provides a set of random restarts for SGA which do not use the topological information identified by our approach.

To compute statistical results, we ran each algorithm on each world type 20 times using randomized starting locations. We also restricted the maximum computation time of each to 600 seconds during these trials using the best solution found by each algorithm before time expired. The

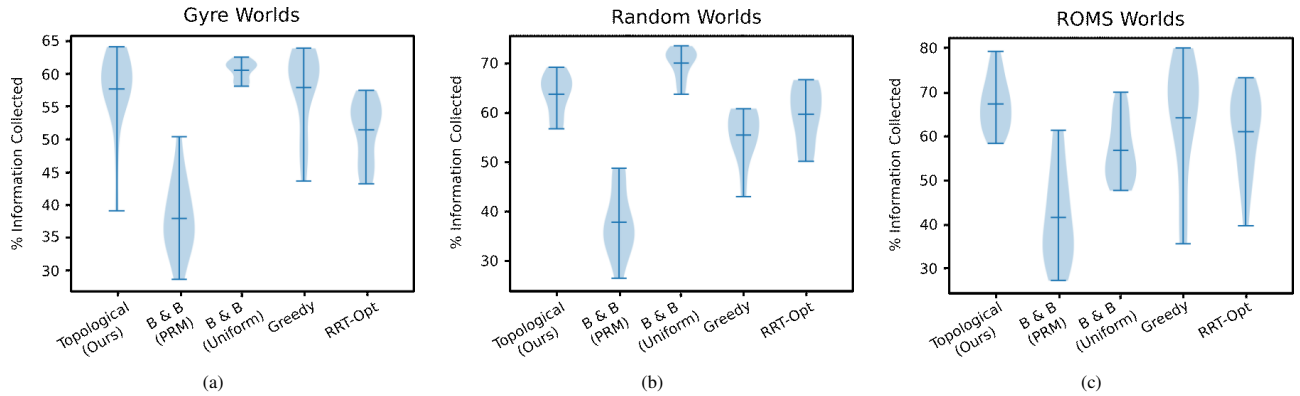


Fig. 6: Percentage of total environmental information collected by a robot using each of the compared methods. The exploration budget used across the Gyre and Random worlds was 150 km. For the ROMS world, the ratio of budget to world size was kept the same, and a budget of 105 km was used.

exploration budget used across the Gyre and Random worlds was 150 km. For the ROMS worlds, the ratio of budget to world size was kept the same, and therefore a budget of 105 km was used. The simulated robot has a sensor radius of 5 km. Similar to the budget, this radius was scaled to 3.5 km for ROMS trials to maintain the ratio to world size.

We first examined the percent of total information gathered by our algorithm and the comparison methods on each of the different world types. In the Gyre world, shown in Fig 3a, our topological method performs comparably with Greedy, and slightly worse than Branch and Bound (Uniform) in terms of the total percentage of information gathered: 57.5%, 57.7%, and 60.4% respectively. A violin plot for these trials with random starting locations and a budget of 150 km is shown in Fig. 6a. In the Gyre world, there is a large amount of information distributed throughout the world, and the world exhibits rotational symmetry. The result is that many topological classes contain the same amount of information, reducing the amount of information provided by each individual class.

In the random worlds, the relative sparsity of information significantly diminishes Greedy's effectiveness, while our method continues to perform well. We achieve an average information collection percentage of 63.8%, compared to an average of only 55.52% for Greedy. Branch and Bound (Uniform) continues to exhibit the best performance, collecting an average of 70.1% of the total information in the world.

Finally, when tested on the ROMS worlds derived from real-world data, our method outperforms its competitors, collecting an average of 67.2% of the information in the environment, compared to 64.4% for Greedy and only 60.8% for RRT-Opt and 56.56% for Branch and Bound (Uniform). This result can be attributed to the magnitude and importance of the topological features in the ROMS data. By reasoning first over these features, our approach is able to place its initial representative path in a good location, which is further improved by optimization. Branch and Bound's poor performance can be attributed to the computational time limitations (Branch and Bound had an average computation time of 600s) combined with the shape of the frontal features. When the fronts align poorly with the uniform grid used by Branch and Bound the algorithm will struggle to find quality paths. This can be counteracted by increasing the resolution of the grid or increasing its connectivity, however these will come at additional computational cost.

The last set of experiments that we performed examined how each of these methods scaled across different planning budgets in ROMS worlds. Results from these trials can be seen in Table I. Results annotated with a star indicate where our method performs significantly better than the comparison methods with 95% confidence (N = 20). At the low budgets, many of the methods perform comparably to ours. With the relatively low budget there are few homotopy classes to search over, and so there is little benefit to considering the topological information. As we scale to higher budgets, the benefits of our proposed approach become more apparent. At a budget of 70 km, both Branch and Bound methods reach the computation time limit of 10 minutes while still performing significantly worse than our method. Increasing the budget again to 105 km, our method performs significantly better than all the comparison algorithms apart from Greedy, which jumps up to be the second-highest performing algorithm. We attribute this change to our implementation of Greedy, which prevents it from taking the same action multiple times. As a result, eventually Greedy will be able to work its way out of local minima. However, from an examination of the distribution of the path scores in Fig. 6c, we can see that our method is able to perform consistently well, while Greedy's performance varies wildly across worlds.

Our topological approach consistently outperforms the naïvely initialized RRT-OPT, and as the budget increases we see a trend of our topological method performing increasingly well when compared to RRT-OPT. This evidence supports our hypothesis that using homotopy information to seed a local optimizer results in higher-quality plans than simply performing random restarts. With longer budgets, the initialization of the optimizer is more important, as it is easier for a path to become trapped in a local minima. By rapidly incorporating global information about the environment through topology, our method is able to better avoid these pitfalls, leading to better performance.

VI. CONCLUSION

In this paper we have presented a novel method for identifying the topology of an information field using Self-

TABLE I: Average results for methods ROMS Worlds across different budgets. Standard deviations in parentheses. Results annotated with a star indicate where our method performs significantly better than the comparison methods with 95% confidence.

	Budget: 35 km		Budget: 70 km		Budget: 105 km	
Method	Computation (Seconds)	Information Collected (% of Total)	Computation (Seconds)	Information Collected (% of Total)	Computation (Seconds)	Information Collected (% of Total)
Greedy	7.21 (0.09)	29.14 (8.96)*	8.35 (0.38)	45.70 (10.73)*	9.71 (0.14)	64.02 (13.06)
Branch and Bound (Uniform)	73.18 (17.97)	**39.26 (3.92)**	600.11 (0.01)	46.64 (7.52)*	600.12 (0.01)	56.56 (6.83)*
Branch and Bound (PRM)	600.17 (0.01)	28.03 (5.36)*	600.22 (0.02)	36.55 (11.30)*	600.29 (0.03)	41.42 (10.42)*
RRT-Opt	52.65 (6.57)	38.45 (4.65)	107.06 (36.55)	54.60 (7.89)	118.81 (29.14)	60.81 (9.17)*
Topological (ours)	58.88 (22.55)	36.00 (5.42)	133.88 (63.82)	**54.87 (6.25)**	259.88 (156.65)	**67.17 (5.86)**

Organizing Maps. Once extracted, a set of topologically distinct trajectory classes can be utilized to generate a set of reference trajectories that span the prominent local maxima of the path space in an 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 algorithm is able to outperform a greedy algorithm as well as a sampling and optimization-based algorithm. We also achieved comparable performance to Branch and Bound, a globally optimal algorithm, in a fraction of the computation time across both randomized worlds and real-world ocean model datasets.

There are many interesting avenues for further research. We demonstrated the capabilities of our approach in a one-shot information gathering task, we would like to expand this algorithm into a multirobot planning framework, enabling a team of robots to collaboratively distribute themselves among likely homotopy classes of trajectories. Additionally, we see areas of improvement for our topology-aware Self-Organizing Map. Through weighting the edges based on the information function during the filtration step, we may be able to improve the efficiency with which we can discover the true underlying topology of an information function.

REFERENCES

[1] D. B. Olson, G. L. Hitchcock, A. J. Mariano, C. J. Ashjian, G. Peng, R. W. Nero, and G. P. Podestá, "Life on the edge: marine life and fronts," *Oceanography*, vol. 7, no. 2, pp. 52–60, 1994.

[2] S. Choudhury, J. D. Gammell, T. D. Barfoot, S. S. Srinivasa, and S. Scherer, "Regionally accelerated batch informed trees (RABIT*): A framework to integrate local information into optimal path planning," in *Proc. of IEEE ICRA*, 2016, pp. 4207–4214.

[3] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.

[4] S. McCammon and G. A. Hollinger, "Topological hotspot identification for informative path planning with a marine robot," in *Proc. of IEEE ICRA*, 2018, pp. 4865–4872.

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

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

[7] J. Binney and G. S. Sukhatme, "Branch and bound for informative path planning," in *Proc. of IEEE ICRA*, 2012, pp. 2147–2154.

[8] B. Charrow, G. Kahn, S. Patil, S. Liu, K. Goldberg, P. Abbeel, N. Michael, and V. Kumar, "Information-theoretic planning with trajectory optimization for dense 3d mapping." in *Robotics: Science and Systems*, vol. 11, 2015.

[9] M. Popović, G. Hitz, J. Nieto, I. Sa, R. Siegwart, and E. Galceran, "Online informative path planning for active classification using uavs," in *Proc. of IEEE ICRA*, 2017, pp. 5753–5758.

[10] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. of IEEE ICRA*, 2011, pp. 4569–4574.

[11] D. Jones and G. A. Hollinger, "Planning energy-efficient trajectories in strong disturbances," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2080–2087, 2017.

[12] D. Jones, M. J. Kuhlman, D. A. Sofge, S. K. Gupta, and G. A. Hollinger, "Stochastic Optimization for Autonomous Vehicles with Limited Control Authority," *Proc. IEEE/RSJ IROS*, Madrid. Oct, 2018.

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

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

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

[16] 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.

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

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

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

[20] J. Faigl, R. Pěnička, and G. Best, "Self-organizing map-based solution for the orienteering problem with neighborhoods," in *IEEE Int. Conf. on Systems, Man, and Cybernetics*, 2016, pp. 1315–1321.

[21] J. Faigl and G. A. Hollinger, "Autonomous data collection using a self-organizing map," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1703–1715, 2017.

[22] G. Best, "Planning algorithms for multi-robot active perception," Ph.D. dissertation, University of Sydney, 2019, Chapter 4.

[23] The GUDHI Project, *GUDHI User and Reference Manual*, 3.0.0 ed. GUDHI Editorial Board, 2019.

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

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

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