Roadmap Learning for Probabilistic Occupancy Maps with Topology-Informed Growing Neural Gas

Manish Saroya[1,2], Graeme Best[1], and Geoffrey A. Hollinger[1]

***Abstract*—We address the problem of generating navigation roadmaps for uncertain and cluttered environments represented with probabilistic occupancy maps. A key challenge is to generate roadmaps that provide connectivity through tight passages and paths around uncertain obstacles. We propose the topology-informed growing neural gas algorithm that leverages estimates of probabilistic topological structures computed using persistent homology theory. These topological structure estimates inform the random sampling distribution to focus the roadmap learning on challenging regions of the environment that have not yet been learned correctly. We present experiments for three real-world indoor point-cloud datasets represented as Hilbert maps. Our method outperforms baseline methods in terms of graph connectivity, path solution quality, and search efficiency. Compared to a much denser PRM*, our method achieves similar performance while enabling a 27$\times$ faster query time for shortest-path searches.**

***Index Terms*—Motion and path planning, computational geometry, probability and statistical methods.**

I. Introduction

A fundamental requirement of all robotic systems is the ability to perform collision-free and cost-effective navigation between locations in an environment. Path planning for reliable navigation is a prerequisite for achieving success at high-level tasks including information gathering [1], package delivery [2], and manipulation [3]. Current path planning approaches typically assume perfect knowledge of the world, which is an unrealistic assumption for physical robot systems operating in challenging environments [4]. We propose algorithms for generating navigation roadmaps of uncertain environments by leveraging probabilistic topological information to inform sampling-based graph generation.

Imperfection and uncertainty are inherent to robotic mapping systems, which arise from a variety of sources, including sensor noise, localization error, line-of-sight occlusions, and dynamic environmental changes [5], [6]. Imperfect maps pose an inherent challenge to reliable navigation as the precise locations of obstacles are unknown to the robot. Probabilistic occupancy maps, such as the Hilbert map [5] illustrated in Fig. 1, model these imperfections with probability distributions. This information is useful to navigation, particularly in cluttered environments and tight corridors, as it provides uncertainty estimates for the location of obstacles and free space. Existing state-of-the-art path planning algorithms, such as RRT and PRM [7], are unable to exploit this information as they typically assume the map is deterministic, where mapping noise causes fallacious connectivity estimates.

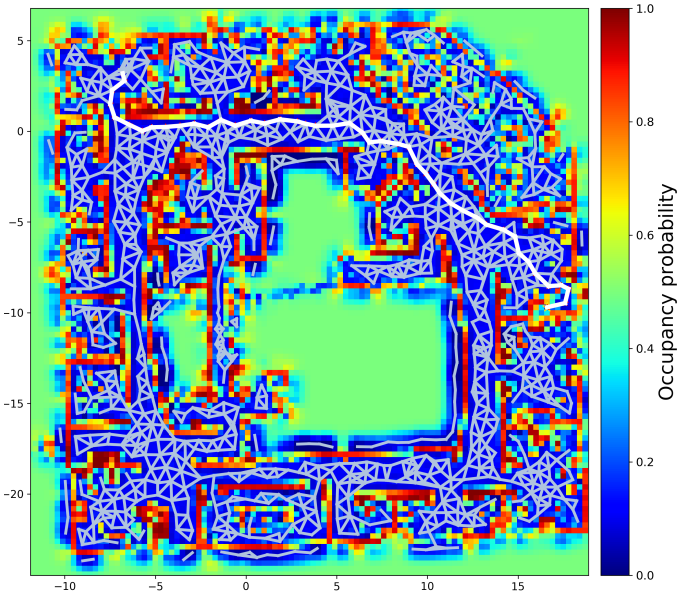


Fig. 1. A roadmap (grey graph) generated by our topology-informed growing neural gas algorithm. The probabilistic occupancy grid (colored image) is generated using a Hilbert map [5] from point-cloud observations of the Intel Research Lab [8]. The white path is a solution path planned over the roadmap through uncertain obstacles and narrow doorways.

We propose a new algorithm for generating navigation roadmaps over probabilistic occupancy maps. The key innovation of our approach is to inform roadmap generation with probabilistic estimates of topological structures of the environment. This topological feedback within the roadmap generation is employed to guide the learning towards critical and challenging regions of the environment, such as uncertain doorways and obstacles, and repair mistakes in the current graph. The topological feedback comes in the form of persistent homology features [9], which are compared between the currently-learned roadmap and the probabilistic world map. The roadmaps are generated with a growing neural gas (GNG) [10], which is a sampling-based graph learning algorithm that generalizes self-organizing maps [11] to have a dynamic topology. The topological feedback is used to

Manuscript received: October 15, 2020; Revised January 16, 2021; Accepted February 20, 2021.

This paper was recommended for publication by Editor Nancy Amato upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by NSF grant IIS-1845227 and Office of Naval Research grant N00014-17-1-2581.

[1]Manish Saroya, Graeme Best, and Geoffrey A. Hollinger are with the Collaborative Robotics and Intelligent Systems (CoRIS) Institute, Oregon State University, Corvallis OR, USA. {saroyam,bestg,geoff.hollinger}@oregonstate.edu

[2]Manish Saroya is also with the Honda Research Institute USA Inc., San Jose CA, USA. msaroya@honda-ri.com

Digital Object Identifier (DOI): see top of this page.

generate a dynamic sampling distribution that informs the growth of the GNG during the learning process.

We perform experiments with probabilistic occupancy maps generated from point-cloud data collected from real robots [8] and Hilbert mapping [5]. Our results demonstrate that roadmaps generated by our proposed algorithm exhibit several key benefits over standard PRMs [7] and GNG algorithms that do not explicitly consider probabilistic topological information. Specifically, we show that our method has superior success rate and average path length, while also producing relatively sparse roadmaps to accommodate efficient graph search. Similar performance is achieved to a dense PRM*, but requires only 4% of the computation time to compute shortest paths. The generated roadmaps, such as the one illustrated in Fig. 1, provide a useful tool for navigating uncertain environments, as well as enabling efficient multi-query path-cost estimates for graph search algorithms such as informative path planning solvers. We have released our code publicly on our repository[1].

II. Related Work

Path planning is the problem of finding collision-free and cost-effective trajectories from one location to another. This capability is critical to robot navigation and therefore is one of the most studied problems in robotics research. Many approaches form single-query solutions, such as RRT [7], where the focus is on finding solutions from the current location to a goal location. Multi-query solutions involve a precomputation step, such as generating a roadmap graph [7], which enables efficiently finding paths for multiple start and goal locations. This is particularly advantageous for planning algorithms that optimize sequences of goals [1], [2].

Most existing path planning methods assume the availability of perfect knowledge of the world [7]; in practice, this is an unrealistic assumption. Existing techniques that attempt to overcome the challenge of map uncertainty involve weighting edge costs by collision probabilities during the query phase [12], [13], but the preprocessing step to generate roadmaps is still challenging. Missiuro and Roy [13] propose biasing the sampling such that the roadmap has greater coverage in highly-certain free-space regions. We propose similar ideas here, in conjunction with topological cues that bias the sampling towards critical and challenging regions.

Biased sampling techniques for roadmap generation in *deterministic* maps have been proposed to increase the chance of making connections through challenging tight passages [14]. Hsu et al. [15] propose successive dilation of obstacles to increase the probability of sampling in critical regions. Deep learning [16] and geometric [17] methods have been proposed to learn the location of critical regions. For single-query planning, techniques have been proposed for sampling only in regions that are likely to improve the current solution [18], [19], [20]. Our algorithm also utilizes a biased sampling approach, but unlike the above methods, it also accounts for uncertainty by leveraging probabilistically-defined topological features. Other work also focuses on increasing the sparsity of roadmaps [21], [22], which is a property that inherently arises from our GNG algorithm.

Our roadmap generation algorithm is a generalization of the GNG algorithm [10]. GNG is an unsupervised learning method that learns a graph-based representation of an underlying distribution. GNGs are closely related to self-organizing maps [11], except that GNGs do not enforce a strict topology, but rather it evolves during the learning process. To make GNGs effective at generating navigation roadmaps, we propose using biased sampling that focuses on growing the graph in critical regions.

Our algorithm borrows ideas from persistent homology theory [9], which provides estimates of probabilistic topological features, to guide the growth of the GNG. These ideas have not previously been used for building navigation roadmaps from probabilistic occupancy maps, but have been leveraged for other robot planning tasks, including single-query path planning [23], trajectory clustering [24], map prediction [25], and informative path planning [26].

III. Problem Formulation

We address the problem of generating a navigation roadmap over a probabilistic occupancy map that facilitates finding cost-effective and collision-free paths between all pairs of locations. We are particularly interested in achieving this for environments that contain topological structures such as rooms, tight corridors, and narrow passages. The roadmap should be well connected, contain cost-effective paths, and be relatively sparse. We define this problem formally as follows.

The environment is represented by a 2-dimensional probabilistic occupancy map that defines a probability of occupancy $P(\zeta)$ at every location $\zeta \in R^2$ in the environment. One example of how to generate these probabilistic occupancy maps from point-cloud observations is to use the Hilbert mapping technique [5], as illustrated earlier in Fig. 1. The uncertainty encoded by $P(\zeta)$ may arise from a variety of sources, such as sensor noise and localization error.

Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a roadmap graph with vertices $\mathcal{V}$ and edges $\mathcal{E}$. Each vertex $v \in \mathcal{V}$ refers to a location $w_v \in R^2$ in the occupancy map. Each edge $e = (v_i, v_j) \in \mathcal{E}$ connects two vertices $v_i, v_j \in \mathcal{V}$ where a straight-line collision-free path exists between v_i and v_j. Each edge e has a cost c_e defined as the Euclidean distance between the two vertices.

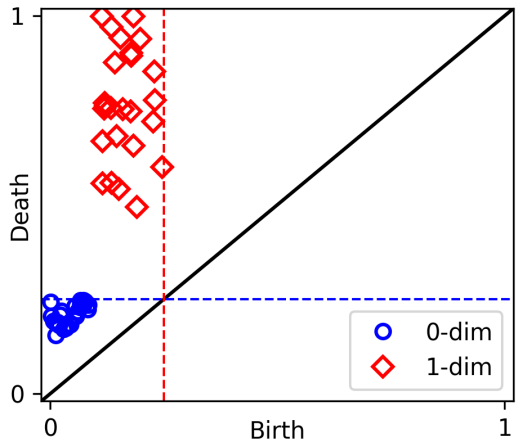
Given a start and goal location represented by $v_s, v_g \in \mathcal{V}$, a path X corresponds to a sequence of vertices $X = (v_1, v_2, ..., v_n)$ such that $(v_i, v_{i+1}) \in \mathcal{E}, \forall i$, and also that $v_1 = v_s$ and $v_n = v_g$. The overall path cost $C(X)$ is given by the sum of the edge costs, i.e.,

$$C(X) = \sum_{i=1}^{|X|-1} c_{(v_i, v_{i+1})}. \tag{1}$$

We address the problem of finding collision-free and cost-effective paths for multiple start ζ_s and goal ζ_s positions. In order to solve this multi-query shortest-path problem on a probabilistic occupancy map, we aim to learn a sparse, topologically-accurate roadmap of the environment. More formally, we aim to generate a roadmap that satisfies the

[1]https://github.com/manishsaroya/GNG

(a) The death locations of the most-persistent topological structures. 0-dim: blue circles; 1-dim: red diamonds. The three cutouts show topologically-challenging regions: (top) a narrow passage connects two parts of a room; (middle) an obstacle lies in the middle of a room; (bottom) a cluttered region consists of several 0-dim and 1-dim structures.



(b) Persistent homology diagram. Shown are the $birth$ and $death$ intensities for the 50 most persistent structures. We only use structures that are less than the two dotted lines since these structures correspond to free-space.

Fig. 2. Persistent homology structures in the Intel environment probabilistic map. (a) shows the death location of the most persistent structures, where 0-dim structures (blue circles) are connected components that correspond to difficult narrow passages, and 1-dim structures (red diamonds) are loops around uncertain obstacles. (b) shows the $birth$ and $death$ probability intensity values for these structures. The sampling for our GNG algorithm is biased towards topological structures that have not been learned correctly in the navigation roadmap.

following three properties for all (ζ_s, ζ_g) pairs. Define X' as the shortest path over the roadmap from ζ_s to ζ_g. Similarly, define X^* as the shortest path over a ground-truth map from ζ_s to ζ_g. Given these definitions, the desirable properties are:

1) **Connectivity:** If the path X^* exists, then the path X' should also exist.
2) **Path quality:** The cost $C(X')$ should be approximately equal to $C(X^*)$.
3) **Search efficiency:** The roadmap is sparse, such that Dijkstra's graph search finds X' by expanding a small number of vertices.

IV. Topology-Informed Growing Neural Gas

We propose the topology-informed growing neural gas algorithm for generating roadmaps over probabilistic occupancy maps. Our method leverages persistent homology features to inform the sampling process of a GNG graph algorithm. This topology-informed sampling guides the growth of the GNG towards challenging regions of the environment in order to ensure connectivity of the roadmap through all noisy narrow passages.

This section begins by introducing the persistent homology features of the probabilistic map and how they correspond to features in the currently-learned roadmap. Then we define the dynamic sampling distribution, followed by a description of the GNG algorithm that combines all of these components.

A. Topology of Probabilistic Occupancy Map

We begin by computing topological structures of the environment, such as rooms, corridors, and narrow passages, via persistent homology theory [9]. Identifying these features allows us to bias the learning of navigation roadmaps (discussed later in Sec. IV-D) towards regions containing these challenging structures.

Persistent homology theory is a branch of computational topology that robustly extracts topological features from noisy representations. The 0-dimensional topological structures of an environment correspond to connected components, such as free space connections between rooms. The 1-dimensional topological structures correspond to free space loops around obstacles. Persistent homology involves the computation of these topological structures and tracks their existence over a spectrum of intensity thresholds.

We generate *persistent homology diagrams* [9] such as the one shown in Fig. 2. This is achieved by converting the probabilistic map into a binary map by thresholding all probability values. This threshold is varied over the spectrum of 0 to 1. The existence of topological structures are tracked as this threshold varies. Every topological structure has associated $birth$ and $death$ thresholds. The $birth$ of a structure is defined as the lowest threshold where this structure exists in the thresholded binary map. Similarly, the $death$ is defined as the highest threshold where this structure exists. A structure exists for all thresholds between its $birth$ and $death$ probabilities. Structures with longer lifespans are said to be more persistent.

Every feature that is tracked in the persistent homology diagram has an associated location that causes the birth event, and another location that causes the death event. These two locations cause these events because their associated probability values are equal to the $birth$ or $death$ threshold. In this work, we focus only on the death locations because

they represent regions that are likely to be challenging for a roadmap generation algorithm to learn correctly. For the 0-dim features, these locations contain noise that hinder the ability to connect two regions of free space. For the 1-dim features, these locations contain noise that hinder the ability to find paths around an obstacle.

In Fig. 2a we show $death$ locations associated with the death of structures given in the persistence diagram in Fig. 2b. For the 0-dim features, we only include features where the $death$ is below a threshold. For the 1-dim features, we only include features where the $birth$ is below a threshold. This filtering ensures we focus on features that are most relevant to possible free space. Out of the remaining features, for each dimension we use the 25 most persistent features, i.e. have longest lifespan. We use the death locations to bias the sampling distribution for our roadmap generation algorithm, which is introduced in Sec. IV-C.

B. Topological Features Within Learned Roadmap

Our topology-informed sampling distribution, defined further below, favors sampling in regions around the death locations of topological features. However, once a topological feature has been correctly learned by our roadmap, further sampling in this region is unnecessary. For this to occur, we need a way of determining when a topological feature has been correctly learned in the current roadmap. We do this by searching for nearby features in the roadmap that correspond to 0-dim and 1-dim features.

For 0-dim features, we identify if they exist in the learned roadmap by looking at a region of radius ρ_0 around the death location. Within the subgraph of the roadmap in this region, we count how many connected components there are, while ignoring isolated single vertices. If there is only 1 connected component, then we determine that this topological feature has been correctly learned in the roadmap.

For 1-dim features, which correspond to loops, we identify if they exist in the learned roadmap by searching for the existence of loops that surround the death location. We first select the closest vertex in the roadmap to this location. A breadth-first graph traversal is performed from this vertex without exceeding a distance of ρ_1 from the death location. In this traversal, loops are detected, and checked to see if they enclose the death location. If the death location is enclosed then we determine that this topological feature has been correctly learned in the roadmap.

C. Topology-Informed Sampling Distribution

We introduce two sampling distributions that are combined to aid the learning of topologically-accurate roadmaps. We leverage topological structures present in the environment to define these sampling distributions. Firstly, we create a map probability distribution and, secondly, we create a Gaussian mixture distribution using structure locations from the persistent homology features defined above.

The map probability distribution $\mathcal{M}(\zeta)$ is obtained directly from the probabilistic occupancy map. First, all locations ζ that have occupancy probability above a threshold are set to $\mathcal{M}(\zeta) = 0$. All other locations are assigned a sampling probability of $\mathcal{M}(\zeta) = 1-\mathcal{P}(\zeta)$, where $P(\zeta)$ is the occupancy probability (see Sec. III). The distribution $\mathcal{M}(\zeta)$ is then normalized to sum to 1.

The topological sampling distribution $\mathcal{T}(\zeta)$ is defined by a masked Gaussian mixture model. The Gaussian mixture is created with mixture components with means defined at the death locations of topological features that have not yet been satisfied by the current roadmap. This satisfaction is determined as described in Sec. IV-B. 0-dim feature components have a fixed standard deviation of σ_0, and 1-dim features have a smaller standard deviation of σ_1. Periodically, we randomly select two of these features to be defined as mixture components, so that the learning spends several iterations at a small number of features before moving on. The distribution $\mathcal{T}(\zeta)$ is masked by multiplying the Gaussian mixture by $\mathcal{M}(\zeta)$ so that occupied locations are not sampled.

The combined sampling distribution is defined such that at every iteration of the GNG algorithm, we sample from $\mathcal{M}(\zeta)$ with some probability or otherwise sample from $\mathcal{T}(\zeta)$. In our implementation, we sample from the topological sampling distribution $\mathcal{T}(\zeta)$ with 0.25 probability.

D. Growing Neural Gas

We propose using the growing neural gas learning procedure [10] for generating navigation roadmaps. However, the standard GNG formulation fails to produce a topologically-accurate roadmap, which we show empirically in Sec. V. Therefore, we propose a generalization of the standard GNG to incorporate the topologically-informed sampling distribution introduced above.

The GNG algorithm incrementally grows a graph that represents a solution roadmap. The algorithm consists of an initialization step followed by a series of epochs, where in each epoch we draw samples and grow the graph towards these samples. Pseudocode is provided in Alg. 1.

The initialization step (lines 1-7) first samples two locations from the map distribution $\mathcal{M}(\zeta)$. The graph vertex set is initialized with these two locations.

Within each epoch (lines 9-47), first the topological sampling distribution $\mathcal{T}$ is updated by receiving feedback regarding the presence of topological features in the current roadmap, as described earlier in Sec. IV-C.

Then, at each iteration of the inner loop, a sample location ζ drawn from the combined sampling distribution is used to grow the graph. Each sample ζ is used to select two nearby vertices, update the corresponding edge, and move neighbouring vertices towards ζ. The age_e of edge e measures how many times one of its neighboring edges have been selected as the winner since the last time edge e was a winner. The $error_v$ of vertex v is defined as the accumulated distance from ζ for all occasions that v was the winner vertex.

At the end of each iteration (lines 34-35), any edges that have an age beyond a threshold a_{max} are removed. Any disconnected vertices are also removed. This edge removal process helps remove erroneous connections and maintain sparsity. Both of these benefits are important in the context of navigation roadmap generation.

Algorithm 1 Topology-Informed Growing Neural Gas.

Input: probabilistic occupancy map $\mathcal{P}(\zeta)$
Parameters: loop limits λ_1, λ_2, λ_3,
step and decay rates ϵ_b, ϵ_n, α, β,
edge age limit a_{max}
Output: roadmap graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$

1: ▷ Generate map sampling distribution (see Sec. IV-C)
2: $\mathcal{M}(\zeta) \leftarrow$ distribution derived from $\mathcal{P}(\zeta)$
3: ▷ Initialize roadmap with two sampled points
4: Sample (ζ_1, ζ_2) from $\mathcal{M}(\zeta)$
5: Create vertex v_1 with location $w_{v_1} = \zeta_1$
6: Create vertex v_2 with location $w_{v_2} = \zeta_2$
7: $\mathcal{V} \leftarrow \{v_1, v_2\}$, $\mathcal{E} \leftarrow \{\}$
8: **for** λ_1 epochs **do**
9: ▷ Update the topological sampling distribution
10: $f \leftarrow$ topological features not satisfied by $\mathcal{G}$
11: $\mathcal{T}(\zeta) \leftarrow$ sum of Gaussians around f
12: **for** λ_2 iterations **do**
13: ▷ Grow graph towards λ_3 samples
14: **for** λ_3 iterations **do**
15: ▷ Draw from combined sampling distribution
16: Sample ζ from $\{\mathcal{M}(\zeta), \mathcal{T}(\zeta)\}$
17: ▷ Find closest vertices to ζ
18: $v_1 \leftarrow$ closest $v \in \mathcal{V}$ to ζ
19: $v_2 \leftarrow$ 2nd closest $v \in \mathcal{V}$ to ζ
20: ▷ Update edge
21: **if** $(v_1, v_2) \notin \mathcal{E}$ **then**
22: $\mathcal{E} \leftarrow \mathcal{E} \cup \{(v_1, v_2)\}$
23: $age_{(v_1,v_2)} \leftarrow 0$
24: ▷ Increment the error of winner vertex
25: $error_{v_1} \leftarrow error_{v_1} + \|w_{v_1} - \zeta\|$
26: ▷ Move winner and neighbors towards ζ
27: $V_{v_1} \leftarrow$ neighboring vertices of v_1
28: $w_{v_1} \leftarrow w_{v_1} + \epsilon_b(w_{v_1} - \zeta)$
29: $w_n \leftarrow w_n + \epsilon_n(w_n - \zeta), \forall n \in V_{v_1}$
30: ▷ Update ages and errors
31: $age_{(v_1,n)} \leftarrow age_{(v_1,n)} + 1, \forall n \in V_{v_1}$
32: $error_v \leftarrow \beta \times error_v, \forall v \in \mathcal{V}$
33: ▷ Remove old edges and isolated vertices
34: $\mathcal{E} \leftarrow \mathcal{E} \setminus \{e\}, \forall e \in \mathcal{E} : age_e > a_{max}$
35: $\mathcal{V} \leftarrow \mathcal{V} \setminus \{v\}, \forall v \in \mathcal{V}$: neighbors of $v = \{\}$
36: ▷ Add new vertex
37: $v_1 \leftarrow \text{argmax}_{v \in \mathcal{V}}[error_v]$
38: $V_{v_1} \leftarrow$ neighboring vertices of v_1
39: $v_2 \leftarrow \text{argmax}_{v \in V_{v_1}}[error_v]$
40: $error_{v_1} \leftarrow \alpha \times error_{v_1}$
41: $error_{v_2} \leftarrow \alpha \times error_{v_2}$
42: create v_{new} with $w_{v_{new}} \leftarrow (w_{v_1} + w_{v_2})/2$
43: $\mathcal{V} = \mathcal{V} \cup \{v_{new}\}$
44: ▷ Update edges
45: $e_1 \leftarrow (v_1, v_{new})$ with $age_{e_1} \leftarrow 0$
46: $e_2 \leftarrow (v_{new}, v_2)$ with $age_{e_2} \leftarrow 0$
47: $\mathcal{E} \leftarrow \mathcal{E} \cup \{e_1, e_2\} \setminus \{(v_1, v_2)\}$

Periodically, a new vertex is added (lines 36-47) in areas that are frequently sampled but have high error. The existing vertex with the highest error is selected and a neighboring vertex. A new vertex is inserted halfway between these two vertices. The original edge is replaced with two new edges. This vertex insertion process helps grow the graph in free-space regions that have not yet been learned correctly.

This learning process continues for a predefined number of epochs. The generated graph directly corresponds to a solution navigation roadmap.

V. Experiments

We present results for experiments with a real-world dataset of indoor environments that demonstrate the behavior and performance of our algorithm. The following results demonstrate that our method achieves similar solution quality and success rate as dense roadmaps while enabling much more efficient graph search. We also highlight the performance benefits of using the topological feedback within the GNG to repair challenging and uncertain regions of the roadmap.

A. Experimental Setup

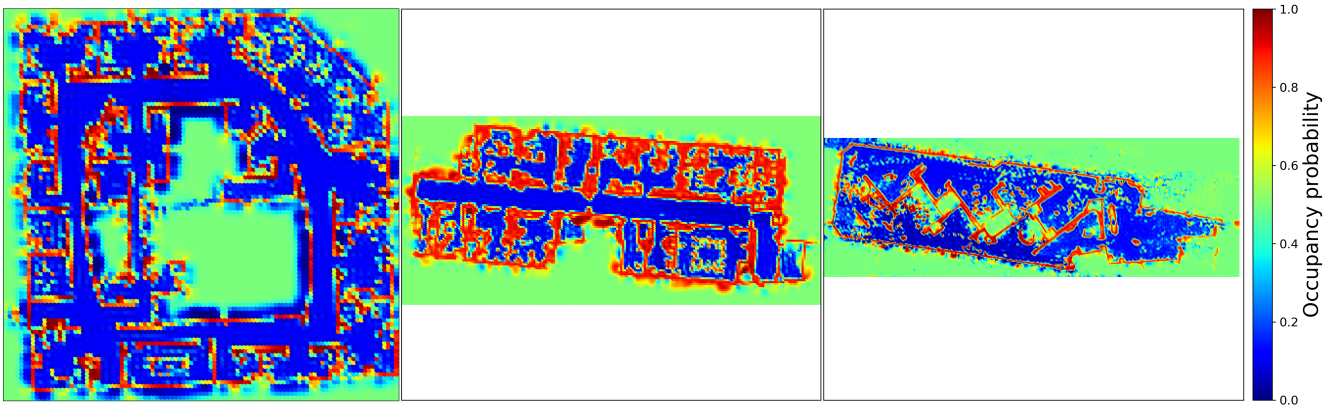
1) Probabilistic occupancy maps: We perform experiments using three real-world 2D point-cloud datasets named Intel, Freiburg, and FHW [8], which are illustrated in Fig. 3. These datasets are from indoor office and laboratory environments, and their topology consists of various structures such as rooms, corridors, narrow passages and obstacles. As the data was recorded from onboard real robots, there is inherent noise in the generated maps, such as occluded regions and noisy obstacles. We generate probabilistic occupancy maps from the point cloud observations using Hilbert maps [5], and our method generates navigation roadmaps with respect to these probabilistic maps.

2) Comparison methods: We compare the following roadmap generation methods:

(a) *dense PRM**: Probabilistic roadmap using the PRM* algorithm [7] with 4000 vertices is used as baseline high-density roadmap.
(b) *topological GNG*: Proposed topology-informed GNG.
(c) *no feedback*: Same as our method except the topological features remain constant such that the feedback defined in Sec. IV-B is not used.
(d) *standard GNG*: A baseline GNG algorithm [10] that does not use the topological sampling distribution $\mathcal{T}$.
(e) *PRM*-2000*: A PRM* with 2000 vertices.
(f) *PRM*-1000*: A sparser PRM* with 1000 vertices, which is the same number of vertices as gets generated by the GNG algorithms.
(g) *PRM*-500*: A sparser PRM* with 500 vertices.

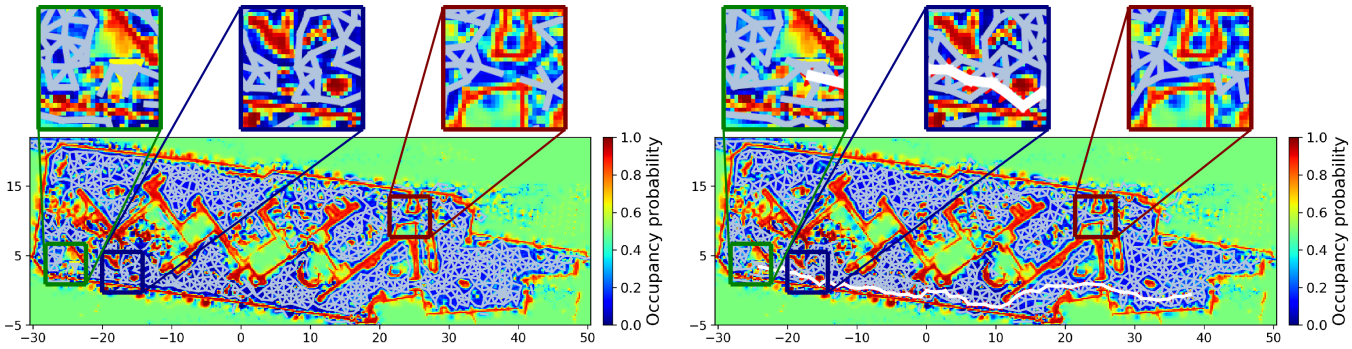
The GNG algorithms learn with respect to the probabilistic Hilbert map, while the PRM* methods use a deterministic map generated by thresholding the probabilistic map.

3) Evaluation metrics: We evaluate the quality of the generated roadmaps with three metrics for 500 random start-goal pairs per map:



(a) Intel Research Lab (b) Freiburg-Building 079 (c) FHW

Fig. 3. An illustration of the probabilistic occupancy maps used in the experiments, generated from the point-cloud datasets [8] recorded by robots in three indoor environments and the Hilbert mapping technique [5]. The environments have varying configurations of rooms, doorways, narrow passages, and noisy obstacles, and are between 20 m and 80 m across.



(a) Roadmap generated with a standard GNG.

(b) Roadmap generated with our topology-informed GNG.

Fig. 4. Two roadmaps generated for the FHW environment. The three cutouts show uncertain narrow passages where the standard GNG in Fig. (a) failed to make a connection and therefore cannot be used for finding paths. Our topology-informed GNG in Fig. (b) biases the sampling towards these challenging regions and therefore can be used to successfully generate a path from one side of the environment to the other (white path).

(i) *Success rate* is a measure of the *connectivity* of the roadmap, computed as the number of start-goal pairs where a feasible solution is found divided by the total number of start-goal pairs.

(ii) *Success-weighted path cost (SPC)* measures the *quality* of the paths generated from a roadmap [27]. SPC is a measure of normalized average path cost, such that any unsuccessful start-goal pairs get penalized with a score of 0, and successful and short paths get scores near 1. SPC is formed by averaging these scores over all start-goal pairs. Formally, SPC is defined as

$$SPC = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{C(X_i^*)}{C(X_i)} \tag{2}$$

where N denotes the total number of start-goal pairs, $C(X_i)$ denotes the path cost, $C(X_i^*)$ denotes the path cost achieved by the baseline *dense PRM**, and S_i is a binary indicator for path success. Higher SPC values correspond to a high quality roadmap.

(iii) *Success-weighted vertices explored (SVE)* measures the amount of *computation* required to search for a path. SVE is a measure of how many vertices get explored by a Dijkstra graph search while searching for shortest paths over the roadmap. SVE is computed in the same way as SPC except with number of vertices explored replacing the path costs $C(X_i)$. Higher SVE values correspond to roadmaps that can be efficiently searched over, while SVE values near 0 indicate the roadmap is dense and therefore slow to search over. The numerator of the ratio uses *dense PRM** again, which has a relatively high number of vertices explored.

4) Implementation details: Parameters used for the GNG methods are: $\epsilon_b = 0.2$, $\epsilon_n = 0.005$, $\lambda_2 = 6$, $\lambda_3 = 100$, $\alpha = 0.5$, $\beta = 0.995$, and the maximum vertices allowed is 1000. The maximum edge a_{max} varied between maps as 70, 56, and 56. The number of GNG epochs λ_1 varied between maps as 200, 300, and 400, due to the different sizes. For the PRM* methods: the constant γ_{PRM} used to determine the variable connection radius (see [7]) varied between maps as 30, 25, 45, and the probability threshold as 0.25, 0.5, and 0.45. All GNG parameters were selected using the default values from the NeuPy library [28] and tuned empirically.

The Hilbert map implementation is from [29]. The Hilbert map is converted into a simplicial complex using Freudenthal triangulation [30] and the GUDHI library [31] computes persistent homology. We use the NeuPy GNG library [28]. Experiments were performed with an Intel i7 8th-gen CPU.

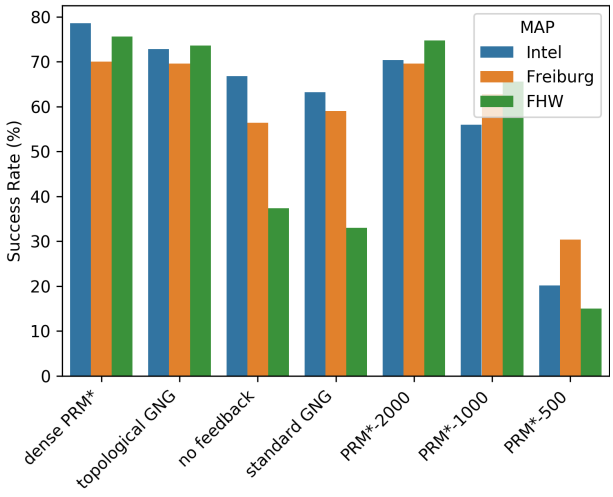


Fig. 5. The success rate achieved by the comparison methods for 500 start-goal pairs in the three maps of Fig. 3. Higher success rates indicate the roadmap has good connectivity across the environment. Our *topological GNG* achieves a similar success rate to *dense PRM** and enables much more efficient search (see Fig. 7).

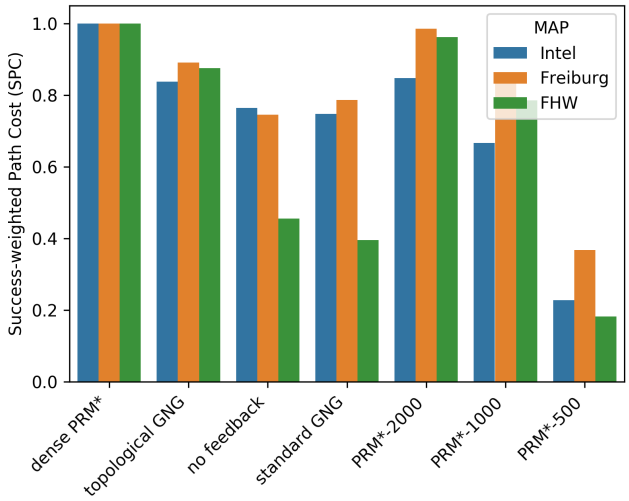


Fig. 6. The success-weighted path cost (defined in Sec. V-A3) achieved by the methods. Higher SPC values indicate that the roadmap generates better quality paths and has a high success rate. Our *topological GNG* finds similar quality paths to *dense PRM** and outperforms the sparser PRMs.

B. Results

1) Illustrative example: Fig. 4b presents an illustrative example of roadmaps generated by the GNG methods. The standard GNG algorithm generates the roadmap in (a), which achieves sufficient coverage and connectivity of the open and certain areas of the environment. However, it fails to find connections through narrow and uncertain areas, as shown in the three cut-outs. These disconnects mean that a graph search will not find a path through these regions. Our topology-informed GNG improves on this by dynamically biasing the sampling toward these regions to repair errors in the roadmap. This results in the improved roadmap shown in (b), which successfully connects the narrow and uncertain corridors, and therefore can be used to find a path across the map. Another topology-informed GNG roadmap is presented earlier in Fig. 1, where we can see that the roadmap achieves good coverage and connectivity by being guided by the topological structures illustrated in Fig. 2a.

2) Numerical results: Fig. 5 compares the achieved success rates. The baseline *dense PRM** had the highest success rate (average 75%) since it has a very high density of vertices and edges. We note that the success rate is below 100% since some start-goal pairs are disconnected in the environment. Our *topological GNG* performed almost as well (72%). The two comparison GNG algorithms performed worse (54%, 52%), particularly in the FHW environment where there is a lot of noisy narrow passages as they could not make use of the topological feedback to repair mistakes in these regions (see Fig. 4b). We note that in Freiburg, *standard GNG* (59%) performed better than *no feedback* (56%), which indicates that *no feedback* spent too many iterations sampling near features that had already been correctly learned. In contrast, our proposed algorithm does not continue to sample a feature once it has been learned correctly. The *PRM*-2000* achieved similar success rate (72%) to our method, but this comes at a cost of higher computation time (discussed below). The sparser PRMs performed worse (61%, 22%) as these roadmaps were disconnected.

Fig. 6 compares the success-weighted path cost. Similar trends can be observed and conclusions drawn as for the success rate metric discussed above. The results here demonstrate that not only does *topological GNG* achieve a similar success rate to *dense PRM**, but also that the quality of these paths are similar in terms of path cost (average 0.87 SPC).

Fig. 7 compares the success-weighted vertices explored. *Dense PRM** performs relatively poorly (1.0 SVE) since Dijkstra's algorithm requires a larger number of expansions to search over this denser graph. Our *topological GNG* generated roadmaps were much more efficient to search over in terms of vertex expansions (9.2). The two comparison GNG algorithms had a similar number of vertices explored as *topological GNG*, but a lower SVE (7.2, 6.8) is achieved due to the lower success rate. The comparison PRMs have a higher SVE (2.0, 3.9, 3.0) than *dense PRM** due to generating sparser roadmaps, but are outperformed by our method.

The computation time for each query (to find a path or report a failure) was consistently faster for the GNG methods due to the improved sparsity compared to the PRM* methods. The query times in milliseconds over the 500 queries and 3 maps had mean μ and standard error σ_μ: *dense PRM**: $\mu = 647$, $\sigma_\mu = 4.0$; *topological GNG*: $\mu = 28$, $\sigma_\mu = 0.5$; *no feedback*: $\mu = 32$, $\sigma_\mu = 0.5$; *standard GNG*: $\mu = 27$, $\sigma_\mu = 0.5$; *PRM*-2000*: $\mu = 252$, $\sigma_\mu = 1.8$; *PRM*-1000*: $\mu = 111$, $\sigma_\mu = 1.0$; and *PRM*-500*: $\mu = 60$, $\sigma_\mu = 0.6$. The offline learning time for GNG methods with topology averaged 260 s, *standard GNG* averaged 100 s, and the PRM* methods were 4 to 15 s.

VI. Discussion and Future Work

Overall, our results demonstrate that our proposed topology-informed GNG achieves a similar success rate and path quality to a dense PRM* while providing a 27× speedup when performing Dijkstra graph searches. In comparison to baseline GNG and sparse PRM* approaches, our method

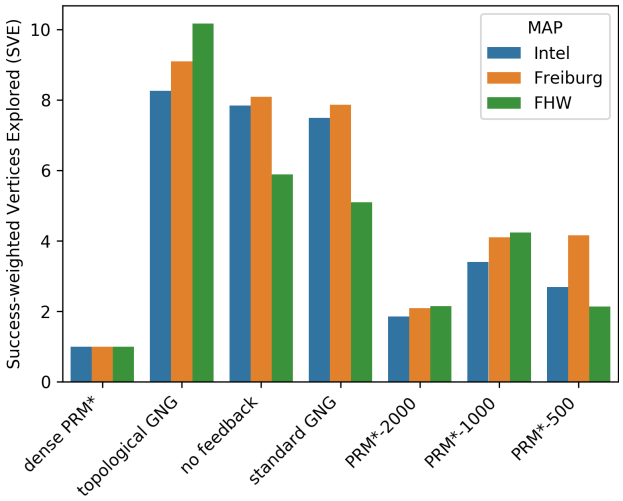


Fig. 7. The success-weighted vertices explored (defined in Sec. V-A3) achieved by the methods. Higher SVE values indicate that the roadmaps are relatively sparse and therefore can be efficiently searched over by Dijkstra's graph search. The baseline *dense PRM** performed relatively poorly here due to the increased number of vertices.

achieved higher performance in all three evaluation metrics. This improvement in performance is due to how our method successfully uses topological feedback to focus the learning in regions of the map that are challenging due to high clutter and uncertainty. These results show that our method is an effective algorithm for generating high-quality, sparse roadmaps of cluttered and uncertain environments.

Our results motivate several avenues of future work. We would like to extend our algorithm for online scenarios where the map is discovered as the robot moves through the world; GNG and related algorithms are well suited for refining prior graphs [32]. It would also be interesting to extend our approach to 3D, which may involve receiving feedback regarding higher-dimensional persistent homology structures. Also, the topology-informed sampling may be useful in other contexts that require processing probabilistic maps, such as single-query path planning and risk-aware planning. Finally, we would like to use the roadmaps generated by our approach for robot planning tasks that depend on reliable and efficient multi-query graph searches, such as for information gathering [1] and package delivery [2].

REFERENCES

[1] G. Best, O. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized planning for multi-robot active perception," *Int. J. Robotics Research*, vol. 38, no. 2-3, pp. 316–337, 2019.

[2] N. Mathew, S. L. Smith, and S. L. Waslander, "Planning paths for package delivery in heterogeneous multirobot teams," *IEEE Trans. Automation Science and Engineering*, vol. 12, no. 4, pp. 1298–1308, 2015.

[3] K. Hauser, "The minimum constraint removal problem with three robotics applications," *Int. J. Robotics Research*, vol. 33, no. 1, pp. 5–17, 2014.

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

[5] F. Ramos and L. Ott, "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent," *Int. J. Robotics Research*, vol. 35, no. 14, pp. 1717–1730, 2016.

[6] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

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

[8] "Robotics 2D-laser datasets," 2014. [Online]. Available: http://www.ipb.uni-bonn.de/datasets/

[9] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington, "A roadmap for the computation of persistent homology," *EPJ Data Science*, vol. 6, no. 1, p. 17, 2017.

[10] B. Fritzke, "A growing neural gas network learns topologies," in *Adv. in neural information processing systems*, 1995, pp. 625–632.

[11] T. Kohonen, "Essentials of the self-organizing map," *Neural Networks*, vol. 37, pp. 52–65, 2013.

[12] L. J. Guibas, D. Hsu, H. Kurniawati, and E. Rehman, "Bounded uncertainty roadmaps for path planning," in *Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2009, pp. 199–215.

[13] P. E. Missiuro and N. Roy, "Adapting probabilistic roadmaps to handle uncertain maps," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2006, pp. 1261–1267.

[14] D. Hsu, J.-C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *Int. J. Robotics Research*, vol. 25, no. 7, pp. 627–643, 2006.

[15] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, S. Sorkin, *et al.*, "On finding narrow passages with probabilistic roadmap planners," in *Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 1998, pp. 141–154.

[16] B. Ichter, E. Schmerling, T. W. E. Lee, and A. Faust, "Learned critical probabilistic roadmaps for robotic motion planning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2020, pp. 9535–9541.

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

[18] M. Rickert, A. Sieverling, and O. Brock, "Balancing exploration and exploitation in sampling-based motion planning," *IEEE Trans. Robotics*, vol. 30, no. 6, pp. 1305–1317, 2014.

[19] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2015, pp. 3067–3074.

[20] W. Reid, R. Fitch, A. H. Göktoğan, and S. Sukkarieh, "Sampling-based hierarchical motion planning for a reconfigurable wheel-on-leg planetary analogue exploration rover," *J. Field Robotics*, vol. 37, no. 5, pp. 786–811, 2020.

[21] W. Wang, D. Balkcom, and A. Chakrabarti, "A fast online spanner for roadmap construction," *Int. J. Robotics Research*, vol. 34, no. 11, pp. 1418–1432, 2015.

[22] O. Salzman, D. Shaharabani, P. K. Agarwal, and D. Halperin, "Sparsification of motion-planning roadmaps by edge contraction," *Int. J. Robotics Research*, vol. 33, no. 14, pp. 1711–1725, 2014.

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

[24] F. T. Pokorny, K. Goldberg, and D. Kragic, "Topological trajectory clustering with relative persistent homology," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2016, pp. 16–23.

[25] M. Saroya, G. Best, and G. A. Hollinger, "Online exploration of tunnel networks leveraging topological CNN-based world predictions," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2020, pp. 6038–6045.

[26] S. McCammon, D. Jones, and G. A. Hollinger, "Topology-aware self-organizing maps for robotic information gathering," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2020.

[27] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[28] Y. Shevchuk, "NeuPy growing neural gas (GNG) algorithm," 2019. [Online]. Available: http://neupy.com/

[29] R. Senanayake and F. Ramos, "Bayesian Hilbert maps for dynamic continuous occupancy mapping," in *Proc. Conf. Robot Learning*, 2017, pp. 458–471.

[30] J. Bey, "Simplicial grid refinement: On Freudenthal's algorithm and the optimal number of congruence classes," *Numerische Mathematik*, vol. 85, no. 1, pp. 1–29, 2000.

[31] The GUDHI Project, *GUDHI User and Reference Manual*. GUDHI Editorial Board, 2020. [Online]. Available: https://gudhi.inria.fr/doc/3.3.0/

[32] G. Best, J. Faigl, and R. Fitch, "Online planning for multi-robot active perception with self-organising maps," *Autonomous Robots*, vol. 42, no. 4, pp. 715–738, 2018.