

Disk-Graph Probabilistic Roadmap: Biased Distance Sampling for Path Planning in a Partially Unknown Environment

Thibault Noël, Sala Kabbour, Antoine Lehuger, Eric Marchand, François Chaumette

▶ **To cite this version:**

Thibault Noël, Sala Kabbour, Antoine Lehuger, Eric Marchand, François Chaumette. Disk-Graph Probabilistic Roadmap: Biased Distance Sampling for Path Planning in a Partially Unknown Environment. IROS 2022 – IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct 2022, Kyoto, Japan. pp.1-8. hal-03752195

HAL Id: hal-03752195
https://inria.hal.science/hal-03752195

Submitted on 16 Aug 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

Disk-Graph Probabilistic Roadmap: Biased Distance Sampling for Path Planning in a Partially Unknown Environment

T. Noël[1,2], S. Kabbour[2], A. Lehuger[2], E. Marchand[3], F. Chaumette[1]

***Abstract*— In this paper, we propose a new sampling-based path planning approach, focusing on the challenges linked to autonomous exploration. Our method relies on the definition of a disk graph of free-space bubbles, from which we derive a biased sampling function that expands the graph towards known free space for maximal navigability and frontiers discovery. The proposed method demonstrates an exploratory behavior similar to Rapidly-exploring Random Trees, while retaining the connectivity and flexibility of a graph-based planner. We demonstrate the interest of our method by first comparing its path planning capabilities against state-of-the-art approaches, before discussing exploration-specific aspects, namely replanning capabilities and incremental construction of the graph. A simple frontiers-driven exploration controller derived from our planning method is also demonstrated using the Pioneer platform.**

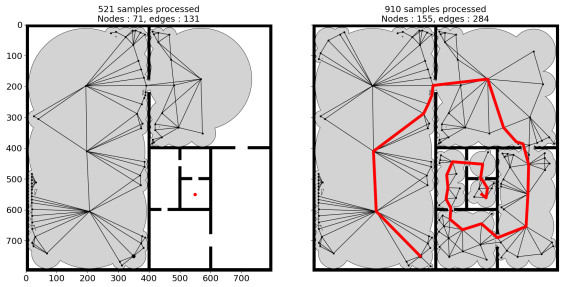


Fig. 1: Planning a path in a bugtrap-like environment with varied room sizes: state of the disk-graph roadmap after processing $n = \{521, 910\}$ samples, the effective roadmap size being $|\mathcal{V}| = \{71, 155\}$ vertices.

I. Introduction

The work presented in this paper is part of a project aiming at the autonomous exploration of an unknown indoors environment; we use a 2D planar LiDAR sensor, which has become a standard for mapping applications due to its high accuracy and decreasing price. Enabling autonomous exploration of an unknown environment for a robotic agent remains a challenging task, for which a generic control architecture can be described as follows:

- SLAM: the incrementally-built map serves as a memory for the agent;
- global planning: it provides high-level exploration targets and generates a feasible path towards them;
- local planning: it improves the feasible path provided by the global planner and returns an optimized path to be followed; and
- trajectory tracking: the robot tracks the trajectory generated from the local planner's path.

Pure reactive control for exploration is practically unfeasible due to the many local minima encountered when reaching dead-ends in the environment; efficient path planning is thus an essential component of exploration controllers, particularly if the higher-level exploration strategy uses a lot of replanning calls.

The classical path planning problem of finding the optimal path between a start and a goal configuration in a well-known environment is tackled by a variety of planning algorithms, usually categorized as search-based or sampling-based planners. However, they generally suffer significant drawbacks in autonomous exploration tasks, facing mapping uncertainties and partial information, which makes the environment model effectively dynamic. Those can be dealt with using a single-query approach, focusing on efficient sampling to rapidly discard areas of the workspace with a low probability of being part of the optimal path; other methods prefer an *anytime* approach, where the planner keeps updating to improve the path after an initial solution is found.

We first briefly discuss the choice of the map representation for the environment and how it can be updated along exploration; we also give a brief overview of the necessity of path planning for exploration strategies before focusing more specifically on sampling-based path planning algorithms.

A. Related work

We first discuss the mapping aspect of autonomous exploration; we consider localization as a separate problem here. A widely adopted method for mapping an unknown environment is **occupancy mapping**, where each spatial configuration is mapped to its probability of being occupied. Occupancy maps can be implemented as Occupancy Grid Maps (OGM) as described in [25]. OGMs have the advantage of being very efficient to compute, but the cell independence hypothesis produces maps with many discontinuities; they also impose a trade-off between map size and resolution, which is restrictive for exploration tasks where the exact size of the environment to be mapped is not known in advance. To address these issues, **continuous occupancy mapping** methods have been proposed [15], [19], [22], as well as **Signed Distance Field** (SDF) representations, where each configuration is mapped to its minimal distance to an invalid configuration [16].

[1] Inria, Univ Rennes, CNRS, IRISA, Rennes, France. Firstname.Name@inria.fr
[2] Créative Ingénierie, Rennes, France.
[3] Univ Rennes, Inria, CNRS, IRISA, Rennes, France.

To derive an exploration strategy from the environment map, two main approaches exist in the literature: frontiers- or information-driven. The hypothesis for frontiers-driven methods, first proposed in [26], is that frontiers between the free and the unknown space in an OGM are the areas where most information about the environment can be retrieved. Information-driven approaches take a different, information-theoretic point of view: by computing the mutual information between the current map and the sensor, they plan towards areas that minimize expected map entropy [4], [7], [21].

Finally, we describe the main path-planning algorithms used to navigate towards the goals provided by the exploration strategy. We focus on sampling-based planners, which do not require a complete model of the environment as optimal search-based planners do. Sampling-based planners can be further divided into tree-based planners, with many methods deriving from the Rapidly-exploring Random Tree (RRT) [14], and graph-based planners among which the Probabilistic Roadmap (PRM) [11] is a base for many planners. Tree-based approaches are generally used as fast planning methods, called every time a path is needed but without any memory of the previous planning calls. Although effective in this regard, one of the main issues of the RRT is the sub-optimality of the paths it returns. To solve this issue, an anytime, asymptotically optimal version of the RRT was proposed in [10], namely the RRT*: at the expense of rewiring the tree when new samples improve the cost between existing nodes and the root, the tree is guaranteed to converge to optimal paths to all nodes. In [9], a Fast-Marching Tree (FMT*) is expanded from a set of initial samples in cost-to-arrive space, improving the computational performance with respect to the RRT* while providing better paths than the RRT. Informed sampling methods are introduced in [5], [6], using an admissible ellipsoid heuristic to avoid sampling the whole environment and improving the convergence of a RRT*-derived planner. In an exploration context, tree-based approaches have the advantage of being expanded iteratively from a root configuration, ensuring connectivity of the tree. However they suffer the drawback of being easily disconnected if the map model is updated; another issue is their lack of flexibility, only providing a unique path for each pair of start and goal nodes in the tree.

On the other hand, graph-based methods such as the PRM offer more choices for navigation in the sense that a given vertex can be part of multiple paths. These planners generally provide longer-term planning capabilities, in the form of a roadmap computed only once and reused for graph-search algorithms such as Dijkstra [2]. However, with the initial version of the PRM, high sampling densities are needed to ensure the connectivity of the roadmap. In [8], a free-space dilatation approach is proposed to better discover narrow passages in PRM by allowing initial sampling inside obstacles before reconstraining the obtained samples in free space. As for the RRT, [10] proposes the PRM*, an asymptotically optimal modification of the PRM, by allowing all valid edges to be added to the graph, without any limit on the number of neighbors. This makes the graph much denser and allows the planner to provide near-optimal paths for all start-goal pairs in the roadmap; however, the initial construction step of the graph and the subsequent graph-search calls can quickly become computationally intractable. To avoid the computational burden of querying the PRM*, non-uniform sampling and graph sparsification methods have also been proposed to help obtain a sparser final roadmap [3], [23], [24]. More recently, [20] proposes a learned graph-based approach to specifically tackle the construction of a roadmap in a probabilistic occupancy map, using a topology-informed Growing Neural Gas (GNG) to sparsely cover the environment while achieving similar performance to the PRM* in terms of paths cost. However, sparse roadmaps have the disadvantage of being easily disconnected when new obstacles are discovered in the environment; this aspect can represent an important additional cost in the context of exploration depending on the roadmap construction method.

B. Contribution

This paper introduces the Disk-Graph PRM (DGPRM), an *anytime* navigation roadmap which naturally expands towards free-space to always provide paths towards unexplored areas of the environment. We rely on a standard occupancy grid-map and on the notion of **free-space bubbles** (initially described in [18] as base for the Elastic-Bands path optimization method) to construct a sparse roadmap, using the state of the disk-graph to bias the sampling step of the roadmap construction. We also describe how the roadmap can be updated on map changes and propose a minimal frontiers-driven exploration controller, demonstrated in a real-world scenario. We show that our method is able to produce sparse roadmaps while maintaining on-par path length performance and better path safety performance when compared to state-of-the-art planners. We also provide an open-source implementation of the method[1]. Section II formalizes the definitions of the environment and of the disk-graph of free-space bubbles which constitutes the roadmap. The biased expansion step and reconnection step are described in Section III. Our minimal frontiers-driven exploration controller is presented in Section III-D. In Section IV, we present the results obtained on a single-query path planning task against the PRM, PRM* and FMT planners. We also present a comparison with the topology-informed GNG on a roadmap construction task; in particular, we demonstrate better results both in terms of final roadmap size and environment coverage, without decreasing performance. We finally describe the qualitative results obtained in the real-world exploration scenario.

II. Environment definitions

We consider the task of constructing a path-planning roadmap in a partially unknown 2D space; we first define the workspace $\mathcal{W} \subset \mathbb{R}^2$, also called "environment" in this paper. The same ideas could apply to higher-dimensional spaces.

[1] https://github.com/thibnoel/disk_graph_prm

A. Occupancy Map and Free-space Bubbles

We rely on a standard OGM to represent the environment: the map $M_{occ}(q_0, \Delta l, m, n)$ associates an occupancy probability $p(q_{ij})$ to all discretized configurations $\{q_{ij} = q_0 + i\Delta l\, e_x + j\Delta l\, e_y | (i,j) \in [0,m] \times [0,n]\}$ where q_0 is the map origin, Δl the map resolution, $[m,n]$ the map dimensions, and e_x, e_y are base units vectors defining the orientation of the map. By defining a discretization function $q \mapsto q_{ij}$ mapping q to the closest discretized configuration in the map, we extend the definition of the occupancy map to the whole workspace. The occupancy probability is set to 0.5 for configurations outside the bounds of the occupancy map.

We also use the thresholded occupancy map to extract the obstacles space, denoted by $\mathcal{C}$, for distance computations in the following; this naive implementation has a significant impact on the performance of the algorithm, nonetheless as our focus is on demonstrating the interest of our approach, more advanced distance checks remain out of the scope of this paper. Note however that our method only requires access to the distance function for sparse collision checks and does not require a complete distance map. The signed distance to the obstacles is classically defined as follows, based on the underlying Euclidean distance in Cartesian space:

$$d_{\mathcal{C}}(q) = \begin{cases} \min_{q_c \in \mathcal{C}} ||q - q_c|| & \text{if } q \in \mathcal{W} \backslash \mathcal{C} \\ -\min_{q_{free} \in \mathcal{W} \backslash \mathcal{C}} ||q - q_{free}|| & \text{otherwise} \end{cases} \quad (1)$$

Using the distance function (1), we can define the free-space bubble $B(q_0)$ associated to a given free configuration $q_0 \in \mathcal{W} \backslash \mathcal{C}$, similarly to [18]:

$$B(q_0) = \{q \in \mathcal{W} \mid d_{\mathcal{C}}(q_0) > ||q_0 - q||\} \quad (2)$$

Note that with our definition of $d_{\mathcal{C}}$, the free-space bubbles are actually 2D disks centered around free configurations and tangent to the obstacles space $\mathcal{C}$. Other distance definitions, for example accounting for a robot's specific geometry, can lead to other shapes for the bubbles as demonstrated in [17]. In our case, a minimal R_0 radius is enforced for the free-space disks (see Section II-B), which is equivalent to modeling the robot as circular with a R_0 radius.

B. Disk-Graph of Free Space

The main idea behind our approach is to extend the definition of free-space bubbles to a roadmap graph, imposing additional validity conditions related to the free-space bubbles associated with the graph vertices, which help constructing a sparse but flexible roadmap. Indeed, for a 2D area of a given shape, the number of free-space disks needed to cover it above a certain coverage rate mostly depends on the **geometrical shape** of the area rather than on its scale, providing a natural way of achieving varying vertices densities in environments with features of varied sizes; this allows for a sparser roadmap far from obstacles, with more densely packed vertices in cluttered areas.

We define the disk-graph as $G(\mathcal{V}, \mathcal{E})$: the set of vertices $\mathcal{V}$ is indexed by $i \in [1, N]$; each vertex v_i is defined as a pair $v_i := (q_i, d_{\mathcal{C}}(q_i))$ representing the free-space disk centered around configuration q_i. To achieve the desired connectivity property and also take into account the physical size of the robot, characterized by radius R_0, we impose the following validity conditions during the construction of the graph:

Vertex validity condition: a candidate vertex v_{cand} is considered invalid if $d_{\mathcal{C}}(q_{cand}) < R_0$ and can not be added to the graph. This condition is also enforced on current graph vertices on map updates to discard vertices becoming invalid on new information.

Edge validity condition: an edge $e_{ij} = (v_i, v_j)$ can only be included in the edges set $\mathcal{E}$ if the free-space disks associated to v_i and v_j overlap, *i.e.,* $B(q_i) \cap B(q_j) \neq \emptyset$. This condition can be equivalently rewritten as follows:

$$(v_i, v_j) \in \mathcal{E} \implies d_{\mathcal{C}}(q_i) + d_{\mathcal{C}}(q_j) > ||q_i - q_j|| \quad (3)$$

Similarly to the vertex validity condition, this edge validity condition is also used to update the roadmap on new map information (see Section III-E). The subsqequent section describes in more details the construction process of the disk-graph roadmap.

III. Disk-Graph Roadmap

In this section, we start by describing the expansion step of the disk-graph, relying on the vertex and edge validity conditions defined just above to derive a more general update rule for a given samples set; we then detail how the current state of the disk-graph is used to inform the sampling procedure, and propose a biased sampling map that ensures the graph is drawn towards free uncovered space. Finally, we propose a procedure to update the disk-graph and maintain its desirable properties on map updates, making it suitable for incrementally discovered environments.

A. Samples Processing: Expanding For Maximal Coverage

We first quickly describe the initialization of the disk-graph: if it is used for roadmap construction in an exploration context, it is initialized from the agent's configuration q_{start} knowing that $d_{\mathcal{C}}(q_{start}) > R_0$, and we have $\mathcal{V}_{init} = \{v_{start}\}$, $\mathcal{E}_{init} = \emptyset$. If used for path planning towards a specific goal, the initialization can be improved by using a *connect* approach [13], simultaneously expanding the roadmap from the start and goal configurations by setting $\mathcal{V}_{init} = \{v_{start}, v_{goal}\}$. Let us now consider the case of checking a unique candidate sample $v_{cand} = (q_{cand}, d_{\mathcal{C}}(q_{cand}))$ and expanding the graph towards it.

1) Per-sample Validity Rule: To determine if a candidate sample should be added to the graph, we could check its validity according to the vertex and edge conditions defined in Section II-B: more specifically, it should have a valid disk radius and a valid edge towards at least one of the vertices currently in $\mathcal{V}$. However, enforcing these conditions imposes strict constraints on the admissible region of the sampling space, especially in the near proximity of obstacles where the small radius of candidate vertices prevent them from reaching existing vertices in the disk-graph. To circumvent this issue while maintaining the connectivity of the roadmap,

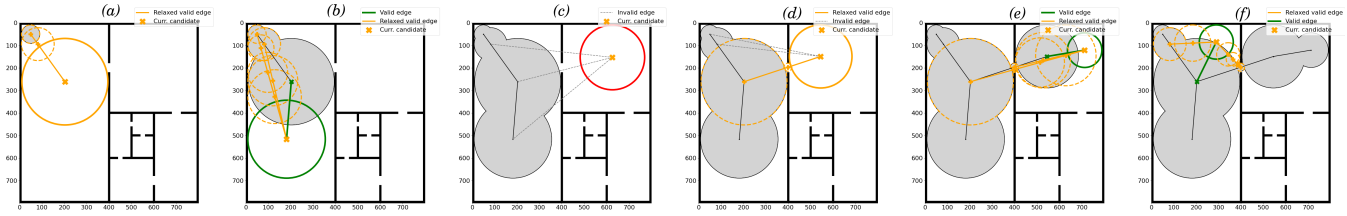


Fig. 2: Expanding the graph towards the open set, querying $k = 3$ nearest neighbors for 6 successive candidate samples (excluding those with invalid radii): *(a)* and *(d)* - *Relaxed edge validity*: a candidate is added with edges to its valid neighbors, and intermediary nodes are added, *(b)*, *(e)* and *(f)* - *Strict edge validity*: the candidate sample has a valid neighbor and gets added without intermediary nodes, *(c)* the candidate sample has no valid neighbors.

we introduce a **relaxed edge validity condition** for a pair of vertices (v_G, v_{cand}), which expresses the intuitive idea that if the straight line between q_G and q_{cand} can be covered with free-space disks centered on the (q_k), then v_{cand} is indeed valid as well as the associated (v_k). It can be expressed as :

$$\begin{aligned} &\exists n_c \in \mathbb{N}, (s_k)_{k\in[1,n_c]} \in [0,1],\ 0 = s_1 < \cdots < s_{n_c} = 1, \\ &\qquad q_k = q_G + s_k(q_{cand} - q_G) \\ &\text{s.t. } \forall k \in [1,n_c]\ d_C(q_k) > R_0 \\ &\qquad \text{and } (v_k, v_{k+1}) \text{ verifies edge validity (3)} \end{aligned} \tag{4}$$

In practice, the (s_k) are constructed recursively by considering the radius-weighted barycenter between q_G and q_{cand}: $s_1 = \frac{1}{2}(1 + \frac{R_G - R_{cand}}{||q_G - q_{cand}||})$, $s_2 = \frac{1}{2}(1 + \frac{R_G - R_{s_1}}{||q_G - q_{s_1}||})$, etc. We simply reorder them if they exist to satisfy $0 = s_1 < \cdots < s_{n_c} = 1$.

In the following, we define the method **TryAddToGraph**(M_{occ}, q_{cand}) which checks the relaxed edge validity condition for the k-nearest graph neighbors of a given candidate vertex and adds the relevant vertices and edges to the graph; an additional check is performed on the intermediary candidates (q_k) when they exist to check if they are inside an existing free-space disk, in which case they are not added to the graph. Fig. 2 illustrates the possible outcomes of a candidate vertex check using the 6 first samples of a candidates set.

2) Generic Graph Update: We now describe a more generic procedure to expand the graph towards a given set of samples, denoted by $\mathcal{S}$. Similarly to the FMT* [9], we use a fast-marching inspired method to hierarchically expand the graph towards most rewarding samples. However, rather than considering cost-to-arrive, we choose to prioritize expansion towards samples with maximal free-space disk radius. Intuitively, this helps quickly eliminating uninformative samples in open areas, and ensures the graph first covers the "easiest" parts of the environment. Formally, we define an unvisited (U) and open (O) sets of samples from the initial samples set $\mathcal{S}$. At initialization, we fix $U = \mathcal{S}$, $O = \emptyset$. Unvisited samples and open samples are updated when a valid candidate sample q_{valid} is added to the graph, getting discarded if they are now inside or close to the graph, or being added to the open set if they are empirically "in range" for expansion. To this end, we define the *opening range* (resp. *closing range*) of q_{valid}, $R_{open}(q) = \rho_{open} \cdot d_C(q)$ and $R_{close}(q) = \rho_{close} \cdot d_C(q)$ where $\rho_{open} > \rho_{close}$ are hyperparameters of the planner. A sample $q_s \in U$ is simply moved to the open set O (resp. discarded) if $R_{close}(q_{valid}) < ||q_{valid} - q_s|| < R_{open}(q_{valid})$ (resp. $||q_{valid} - q_s|| < R_{close}(q_{valid})$). The graph expansion procedure is summed up in Algorithm 1.

Algorithm 1: ProcessSamples(M_{occ}, $\mathcal{S}$, G)

Input: occ. map M_{occ}, samples set $\mathcal{S} = (q_i)_{i\in[1,n]}$, current graph G
Output: unvisited samples U, updated graph G

$U \leftarrow \mathcal{S}$ `/* Unvisited set */`
$O \leftarrow \{\}$ `/* Open set: priority queue (PQ) */`
$q_{curr} \leftarrow$ FindValidInitSample(U)
O.AddToPQ(q_{curr}, BubbleRadius(M_{occ},q_{curr}))
while *not* O.*IsEmpty()* **do**
  q_{curr} = O.PopFromPQ()
  added = G.TryAddToGraph(M_{occ},q_{curr})
  `/* Handles collision and k-NN checks */`
  if *added* **then**
    for $q \in$ *SamplesInOpeningRange*(q_{curr}, U) **do**
      O.AddToPQ(q, BubbleRadius(q))
    end
  end
end
return U,G

Remark: the **FindValidInitSample** method used in Algorithm 1 finds a first valid sample when the open set is still empty; in our case, we implement it by considering U as an initial open set, returning the first valid sample in disk-radius descending order; other search heuristics could be used, for example the Euclidean distance to a specific goal.

B. Biased Sampling

Having defined how a given sample set is processed, we now describe how the samples are chosen to heuristically maximize the probability of expanding the current graph towards free space. Most path planning methods use uniform sampling over the workspace (or a bounded region of the workspace for tasks such as navigation) to generate random configurations; it has the advantage of being computationally very cheap and for workspaces with a low density of invalid configurations, few samples will be rejected. However for navigation tasks specifically, where the workspace can generally be related to a Cartesian physical space, sampling the workspace uniformly also has the disadvantage of favoring a specific scale, depending on the chosen sampling density: spatial features smaller than this typical scale will often

be missed while those bigger will be unnecessarily "over-covered" by many samples.

Following the idea of informed sampling, we use the state of the current disk-graph to bias the sampling of candidate configurations. To this end, we define a discrete sampling map M_S matching the resolution and dimensions of the occupancy map M_{occ}, and a random variable $X \in \mathcal{W}$ so that, on a given sampling event:

$$p(X = q_{ij}) \propto M_S[i,j] \tag{5}$$

To construct M_S, we first define the *inflated* distance to the graph surface, denoted by d_G. The inflated distance to the surface of the free-space disk of a given vertex v_0 is defined as $d_{v_0}(q,\rho) = ||q - q_0|| - \rho d_C(q_0)$, where ρ is the inflation factor. The inflated distance to the surface of the graph can be obtained by taking the minimum on all vertices:

$$d_G(q,\rho) = \min_{v \in \mathcal{V}} d_v(q,\rho) \tag{6}$$

The idea is then to bias the sampling map to be zero for discrete configurations verifying $d_G(q_{ij},\rho) < 0$, with the parameter ρ controlling more precisely the admissible sampling area around the graph free-space disks. However, simply sampling the environment uniformly outside of the graph-masked space still conducts to drawing a high rate of invalid samples; to further bias the sampling map, we propose to apply an exponential distribution depending on the graph distance d_G to the sampling map, so that

$$M_S[i,j] \propto \lambda e^{-\lambda d_G(q_{ij},\rho)} \tag{7}$$

where the parameter λ (also referred to further as $d_{samp} := \frac{1}{\lambda}$) controls the probability decay when considering configurations further and further away from the disk-graph surface. To finalize the construction of the sampling map, we also exclude the invalid and unknown configurations extracted from the OGM, *i.e.* $M_{occ}[i,j] \geq 0.5 \implies M_S[i,j] = 0$. In Fig. 3, we demonstrate the combined effect of the tuning parameters ρ and d_{samp} on the final biased sampling map.

C. Path Planning

To actually plan paths in the environment, the disk-graph roadmap is expanded using successive batches of n_S new samples, where n_S depends linearly on the number of current vertices in the graph up to a maximum value n_{Smax}. At any time, we can define the reachable space $\mathcal{R}$ as:

$$\mathcal{R} = \{q \in \mathcal{W} \backslash \mathcal{C} \mid d_G(q, \rho = 1) < 0\} \tag{8}$$

For any pair of configurations $(q_{start}, q_{goal}) \in \mathcal{R} \times \mathcal{R}$, assuming connectivity of the disk-graph roadmap, a path $\Gamma : [0,1] \to \mathcal{R}$, verifying $\Gamma(0) = q_{start}, \Gamma(1) = q_{goal}$, exists and can be constructed as follows: we first define v_{init} (resp. v_{final}) as the closest graph vertex to q_{start} (resp. q_{goal}). The roadmap connectivity ensures the Dijkstra algorithm [2] provides a graph path $(v_i)_{i \in [0, n_\Gamma]}$ with $v_0 = v_{init}, v_{n_\Gamma} = v_{final}$. The path Γ can finally be linearly interpolated from the list of the (v_i) configurations, extended with the start and goal

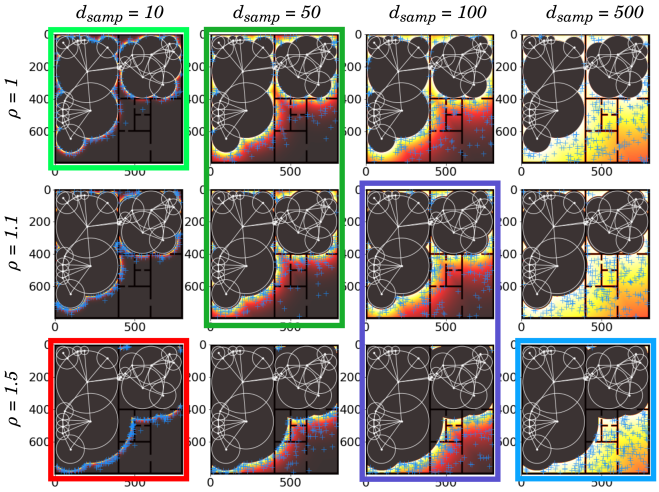


Fig. 3: Sampling hyperparameters influence : from the same initial graph, we draw 200 samples in the displayed probability maps. We can see how ρ controls the minimal distance between the new samples and the current graph, acting as an additional inflation to ensure larger subsequent bubbles; d_{samp} controls the probability spread, the distribution becoming more uniform when d_{samp} grows. With a fixed number of samples, using relatively low values for ρ and d_{samp} ensures good discovery of small spatial features in the environment (light green and green), while higher values allow a faster expansion towards uncovered space (dark blue and blue). A high ρ, low d_{samp} combination (red) generally leads to poor performance. In the experiments, we usually choose ρ close to 1 and $d_{samp} \approx n \cdot l$ where $n \in [3,8]$ and l is the characteristic size of the smallest spatial features.

positions: $q_\Gamma = \{q_{start}, q_0^*, q_1, \ldots, q_{n_\Gamma - 1}, q_{n_\Gamma}^*, q_{goal}\}$. Here we define q_0^* and $q_{n_\Gamma}^*$ as

$$q_0^* = q_0 + d_C(q_0) \frac{q_1 - q_0}{||q_1 - q_0||} \tag{9}$$

$$q_{n_\Gamma}^* = q_{n_\Gamma} + d_C(q_{n_\Gamma}) \frac{q_{n_\Gamma - 1} - q_{n_\Gamma}}{||q_{n_\Gamma - 1} - q_{n_\Gamma}||} \tag{10}$$

to link the start and goal configurations to the perimeter of their neighboring disk-graph vertex rather than to its center, avoiding sharp turns at the path ends.

A notable advantage of using a graph over a tree is the ability to vary the weight of the graph edges to bias the Dijkstra algorithm and select among multiple paths when available; this allows to easily prioritize between shortest and safest path, which is a common trade-off to consider when planning in partially unknown environments.

D. Frontiers Characterization For Exploration

To also demonstrate the interest of the approach as the base planner for a minimal exploration strategy, we introduce a simple frontiers characterization. It relies on the standard definition of frontier cells in an occupancy grid map, i.e. the set of free space configurations with unknown neighbors. We define the set of **frontier vertices** in the disk-graph as vertices whose free-space disks overlap unknown cells of the underlying occupancy map. From there, we derive a simple strategy consisting in navigating towards the closest frontier vertex in terms of arc-length cost. Additionally, to ensure a choice always exists between multiple targets while limiting sampling calls, the biased sampling procedure is periodically called as long as the current frontiers set does not contain a number of nodes comprised in a range $[n_{min}, n_{max}]$.

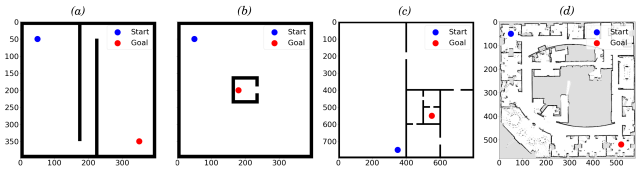


Fig. 4: The 2D environments used for the single-query path planning task: *(a)* Narrow passage, *(b)* Bugtrap, *(c)* Successive bugtraps, *(d)* Intel Research Lab

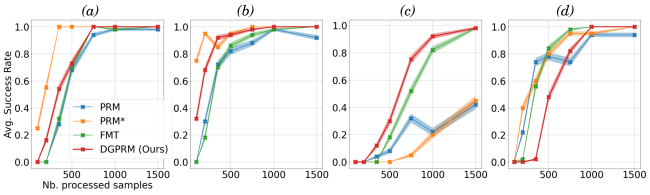


Fig. 5: Average planning success rates for the single-query task; the *(a)* to *(d)* graphs correspond with the environments of Fig. 4. Transparent outlines show standard deviations.

We keep the strategy minimal by choice to demonstrate the exploratory behavior of the planner without additional high-level strategy considerations. Indeed, the planner is not searching for a specific goal in this case, we only always plan towards frontier nodes in reachable space. This therefore shows the natural tendency of the planner to expand towards uncovered space (similarly to the RRT).

E. Updating the Planning Graph with New Information

We finally describe how to update the disk-graph roadmap to eventually reflect the latest state of the map while retaining good planning abilities. The first step is to recompute all the vertices free-space disks radii to check if they have been impacted by new obstacles added to the occupancy map; when modified, they are registered as modified vertices. Among them, we then check for vertices for which we now have $d_C(q_v) < R_0$, registering them as invalid. The second step is to delete invalid edges : we check those linked to modified vertices for new collisions with obstacles and the edge validity condition (Eq. 3). At this step, all invalid nodes and edges are deleted from the graph, potentially splitting the roadmap into multiple connected components; to solve this issue, we first discard all connected components with a size below a minimal number of vertices; second, we identify the main component as the one containing a reference configuration q_{ref}. We finally reconnect the other components, checking pair-wise with the relaxed edge condition (Eq. 4), discarding those for which no reconnection to the main component is possible.

This step is the most costly computationally, especially with our naive distance implementation; however, it only applies to the case of using the disk-graph in a context where the map is dynamically updated, and has therefore no effect when building a roadmap from a known environment.

IV. Experimental Results

In this section, we first present the numerical results obtained with our method for a single-query path-planning task, demonstrating on-par performance with state-of-the-art planners in terms of path cost while providing a much sparser roadmap, improving subsequent query times. We then consider a roadmap construction task, showing how our method can construct a sparser roadmap than the baseline methods without hindering path cost. We also describe the qualitative results obtained with our proposed frontiers-driven exploration controller.

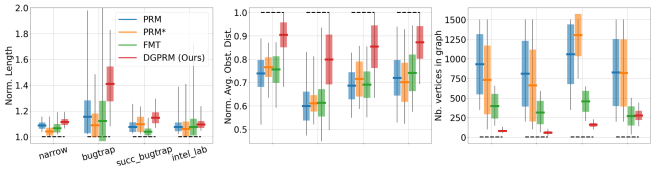


Fig. 6: Extended single-query path-planning results: *(a)*: Average path length normalized by minimal path length; *(b)*: Average path safety normalized by maximal path safety; *(c)*: Average final graph sizes. Transparent boxes show standard deviations.

A. Numerical Results in Static Environments

The following methods are considered for comparison: PRM, PRM*, FMT* (*single-query task only*) and topology-informed (TI-)GNG (*roadmap construction task only*). All the methods used here for comparison are adapted from open-source implementations, except for the topology-informed GNG for which we directly include the results provided by the authors in [20].

1) Single-Query Path Planning Task: We first compare the performance of our method on a single-query path planning task, showing how biased sampling can lead to improved performance for environments with features of varied sizes. Three simple handcrafted 2D occupancy maps, highlighting path-planning difficulties (narrow passages, dead-ends), and a more realistic occupancy map generated from real 2D measurements[2], are proposed for evaluation (see Fig. 4); a single start-goal pair is considered for each of them. As a main comparison metric, we use the success rate of the planners in relation to the number of *processed* samples: for each method, the planning success (*i.e.*, discovery of a valid path by the planner) is averaged over 50 single-query plans, each limited to n_{max} environment samples. To provide a more in-depth comparison, we also compare the average path length and path safety achieved by each method on successful planning attempts: the path safety cost is defined as the average distance between the path and the obstacles, and computed using a fine subdivision of the final path. Similarly, we compare the average size of the final graphs for successful planning attempts.

For the baseline planners, the hyperparameters were chosen empirically to maximize performance; they however share a maximal edge length constraint. This maximal length is also used as the d_{samp} scale factor for the biased sampling of the DGPRM ($d_{samp} = 50$ for environments *(a)* and *(b)*, 100 in *(c)* and *(d)*). The number of closest neighbors is set to 8 for the standard PRM and 5 for the DGPRM, which

[2]http://www.ipb.uni-bonn.de/datasets/

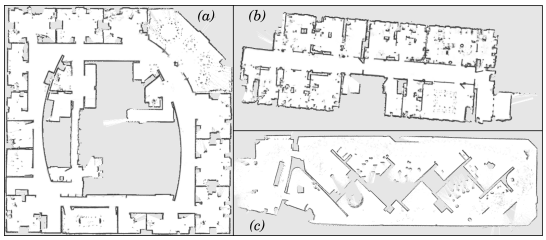


Fig. 7: The 2D environments used for the roadmap construction task: *(a)* Intel Research Lab, *(b)* Freiburg building, *(c)* FHW

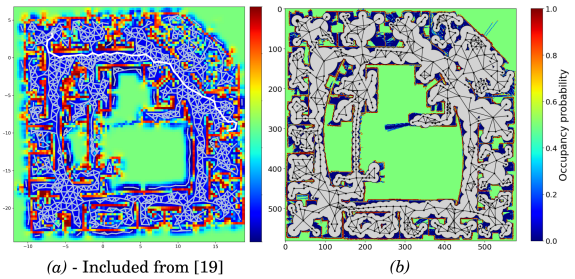


Fig. 8: A visual comparison between the final roadmap obtained with our method and the topology-informed GNG, in the Intel Research Lab environment: *(a)* Fig. 1 from [20], GNG roadmap trained on a Hilbert occupancy map, providing a 1000 vertices-roadmap (scale: meters), *(b)* Disk-graph roadmap expanded on an occupancy grid map, with 478 final vertices (scale: pixels)

provided the best balance between average success rate and graph sparsity in our test environments.

The average success rates for each environment are presented in Fig. 5: for the environments *(a)* and *(b)*, our method requires additional samples compared to the baseline FMT* and PRM*. Note how the performance of the standard PRM decreases in the bugtrap environment, due to the small reachable area around the goal, making the uniform sampling less efficient. In environments *(c)* and *(d)*, the PRM and PRM* performances strongly decrease due to narrow passages and small-sized features which require many more samples if sampling uniformly. The FMT* relies on uniform sampling as well but remains more efficient by expanding selectively towards cost-to-arrive space.

Fig. 6 presents the path costs and final graph sizes in each environment, for all successful planning attempts of every baseline planner. We see that the DGPRM provides longer paths on average in all environments, sometimes up to 50% longer than the shortest path found; however, we also note that this is balanced by the improved path safety, with the DGPRM always providing the safest path, although being queried here with a shortest length cost; this is the consequence of the expansion towards nodes of maximal radius. In terms of final graph size, the PRM and PRM* results are biased by their construction but provide a "worst-case" baseline where all the samples are added to the roadmap graph; however, the DGPRM achieves successful planning with sparser map coverage than even the FMT.

2) Roadmap Construction Task: We now consider a roadmap construction task: here, the objective is to measure how efficiently (*i.e.* sparsely) the disk-graph roadmap can cover the environment while maintaining connectivity,

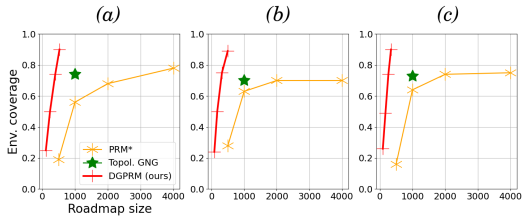


Fig. 9: Roadmap construction results - environment coverage vs final roadmap size

and how costly the obtained paths are. The environments proposed for evaluation are shown in Fig. 7. We use two evaluation metrics defined in [20] to allow for direct comparison:

- The **success rate** (SC) measures the connectivity of the roadmap; in [20], it is defined as the proportion of start-goal pairs connected in the roadmap. For our method, the disk-graph is always connected by construction, so we rather define the success rate on the number of connected start-goal pairs in the map free space, effectively measuring it as the relative surface of free space overlapped by the disk-graph roadmap.
- **Success-weighted path cost** (SPC) is given by

$$SPC(G) = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{c(\Gamma_i)}{c(\Gamma_i^*)} \tag{11}$$

where c is the path cost, here the Euclidean arc-length, i indexes the start-goal pairs in the roadmap, Γ_i^* is an optimal path for start-goal pair i, Γ_i the path provided by the roadmap, and S_i denotes boolean success.

The SPC is actually computed for 500 random start-goal pairs in the environment, using a dense (4000 vertices) PRM* to compute the reference path cost. The success rates of the PRM*, TI-GNN and of our method for the proposed environments are shown in Fig. 9, demonstrating the improved sparsity in the roadmaps generated by our method; this sparsity reduces the cost of all subsequent planning calls, particularly since the roadmap we construct is always connected. We note however that it comes at the expense of a higher sampling and processing cost, many samples being rejected during the expansion phase even before being considered as candidates. Table I provides the SPC values obtained for the IT-GNG and our method, showing that the sparsity of our roadmap does not impact the path cost in a severe way: our method is able to provide paths with a similar quality as the TI-GNG while containing around half the vertices in the environments tested here (Fig. 8).

B. Qualitative Exploration Results

We finally present the results obtained with the exploration controller described in Section III-D. The frontiers characterization is extended with an open-source SLAM algorithm (namely Hector-SLAM [12]) and a standard non-linear path following controller [1] to form a complete frontiers-driven exploration strategy. Additionally, replanning

TABLE I: Detailed roadmap evaluation on success rate (SR), i.e. environment coverage, and success-weighted path cost (SPC) as proposed in [20].

Env.	Method	Roadmap size	SR	SPC
Intel Lab	DGPRM (Ours)	616	0.95	0.91
	TI-GNG	1000	≈0.75	≈0.85
Freiburg bldg.	DGPRM	540	0.9	0.89
	TI-GNG	1000	≈0.7	≈0.9
FHW	DGPRM	411	0.94	0.87
	TI-GNG	1000	≈0.75	≈0.9

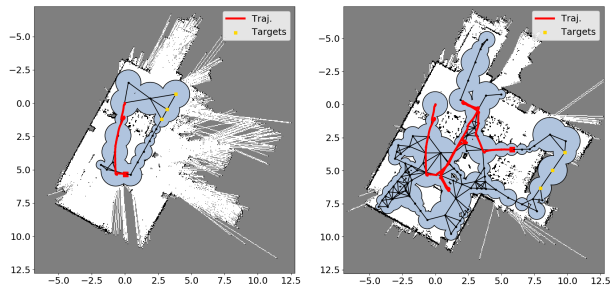


Fig. 10: Qualitative results of an exploration conducted with the Pioneer in a real environment. Current frontier vertices are highlighted in yellow.

is triggered if the current path becomes invalid or if the current target frontier no longer belongs to the frontiers set. This proposed controller is implemented using the ROS middleware to facilitate the usage of external open-source libraries, and deployed on the Pioneer mobile platform. The robot is set up with no initial knowledge in an indoors kitchen environment and left to explore. In the experiments, we used a minimal radius slightly higher than the robot radius for added security; we managed an update frequency of the disk-graph (recomputation of nodes radii and reconnections) of approximately 1Hz for graphs up to 100 to 200 nodes, noting the implementation optimization is still at an early stage. On Fig. 10, the occupancy map is shown at two different stages during exploration, along with the robot trajectory and the state of the disk-graph roadmap.

V. Discussion and Future Work

In this paper, we showed how biased sampling around free-space disks can be applied to the construction of a sparse but efficient roadmap in a 2D occupancy map. We also demonstrate the inherent exploratory behavior of our method. However, as noted previously, the implementation can still be improved to achieve better performance; we also envision the following topics for future work: computing the distance directly from a geometric characterization description of the obstacles would greatly decrease computational complexity; we also want to test the integration of the disk-graph roadmap with a more advanced exploration strategy. Finally, we expect the method to still be valid for higher-dimensional spaces, especially for 3D exploration.

References

[1] C. Canudas de Wit, H. Khennouf, C. Samson, and O. J. Sørdalen. *Nonlinear control design for mobile robots*, pages 121–156. Recent Trends in Mobile Robots. 1994.

[2] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[3] Andrew Dobson and Kostas Bekris. Sparse roadmap spanners for asymptotically near-optimal motion planning. *International Journal of Robotics Research*, 33:18–47, 01 2014.

[4] G. Francis, L. Ott, and F. Ramos. Functional path optimisation for exploration in continuous occupancy maps. In *Int. Symp. on Robotics Research*, 2017.

[5] J. Gammell, S.. Srinivasa, and T. Barfoot. Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. *IEEE Int. Conf. on Robotics and Automation*, May 2015.

[6] J.D. Gammell, S. Srinivasa, and T. Barfoot. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Sep 2014.

[7] T. Henderson, V. Sze, and S. Karaman. An efficient and continuous approach to information-theoretic exploration. In *IEEE Int. Conf. on Robotics and Automation*, pages 8566–8572, May 2020.

[8] D. Hsu, L. Kavraki, J.C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *Workshop on the algorithmic foundations of robotics, WAFR '98*, page 141–153, August 1998.

[9] L. Janson, E. Schmerling, A. Clark, and M. Pavone. Fast marching tree: a fast marching sampling-based method for optimal motion planning in many dimensions. *Int. J. of Robotics Research*, 34(7):8883–921, July 2015.

[10] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *Int. J. of Robotics Research*, 30(7):846–894, July 2011.

[11] L. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. on Robotics and Automation*, 12:566–580, September 1996.

[12] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *IEEE Int. Symp. on Safety, Security and Rescue Robotics*, November 2011.

[13] J.J. Kuffner and S.M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 995–1001, 2000.

[14] S. LaValle. Rapidly-exploring random trees: a new tool for path planning. 1998.

[15] S. O'Callaghan and F. Ramos. Gaussian process occupancy maps. *Int. J. of Robotics Research*, 31(1):42–62, January 2012.

[16] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart. Signed distance fields: A natural representation for both mapping and planning. In *Workshop on Geometry and Beyond, RSS'2016*, 2016.

[17] S. Quinlan. *Real-time modification of collision-free paths*. PhD thesis, Stanford University, 1994.

[18] S. Quinlan and O. Khatib. Elastic bands: connecting path planning and control. In *IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 802–807, 1993.

[19] F. Ramos and Lionel Ott. Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. *The International Journal of Robotics Research*, 35:1717 – 1730, 2016.

[20] M. Saroya, G. Best, and G. Hollinger. Roadmap learning for probabilistic occupancy maps with topology-informed growing neural gas. *IEEE Robotics and Automation Letters*, 6(3):4805–4812, July 2021.

[21] K. Saulnier, N. Atanasov, G. J. Pappas, and V. Kumar. Information theoretic active exploration in signed distance fields. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4080–4085, 2020.

[22] R. Senanayake and F. Ramos. Bayesian hilbert maps for dynamic continuous occupancy mapping. In *1st Annual Conference on Robot Learning*, pages 458–471, Nov 2017.

[23] Doron Shaharabani, Oren Salzman, Pankaj Agarwal, and Dan Halperin. Sparsification of motion-planning roadmaps by edge contraction. *The International Journal of Robotics Research*, 33, 09 2012.

[24] Thierry Siméon, Jean-Paul Laumond, and Carole Nissoux. Visibility-based probabilistic roadmaps for motion planning. journal of advanced robotics, 14(6), 477-494. *Advanced Robotics*, 14:477–493, 01 2000.

[25] C. Stachniss. *Robotic Mapping and Exploration*, volume 55. Springer Tracts in Advanced Robotics, 2009.

[26] B. Yamauchi. A frontier-based approach for autonomous exploration. In *IEEE Int. Symp. on Computational Intelligence in Robotics and Automation CIRA'97.*, pages 146–151, 1997.