Skeleton Disk-Graph Roadmap: a Sparse Deterministic Roadmap for Safe 2D Navigation and Exploration

Thibault Noël[1,2], Antoine Lehuger[2], Eric Marchand[3], François Chaumette[1]

***Abstract*— In this paper, we describe a novel roadmap construction method in unknown environments, which relies on the extraction of the Hamilton-Jacobi skeleton of the free space. This skeleton is used to construct a graph of free-space bubbles, effectively compressing the skeleton information in a sparse data structure but retaining its topology. The bubbles also enforce safety directly in the roadmap structure. We first demonstrate the relevance of this approach for standard path-planning tasks. We also propose a frontiers-based exploration strategy able to autonomously and safely build a complete 2D map of the environment.**

I. Introduction

Exploration of unknown environments in mobile robotics has applications in search and rescue, industrial inspection and prospection, active SLAM, etc. However, it remains a complex task for a robot to execute autonomously. The main challenges are to tackle the high uncertainties associated to the environment model and to derive an efficient planning strategy that can accommodate the growing size and complexity of the environment, while providing high-quality navigation paths. The crucial components needed for a complete exploration strategy are thus:

- the mapping module, generally handled by a SLAM algorithm, which provides an accurate representation of the environment and manages localization.
- the planning module, which uses the map information to compute the trajectory of the robot; planning can often be divided into a global planning method providing feasible collision-free paths followed by a local post-processing method which optimizes the path for desirable properties (length, clearance).
- the navigation module executing the planned paths

In this paper, we focus on the planning aspect and propose a deterministic roadmap planner that achieves high coverage of the environment while remaining sparse and providing high-quality paths, illustrated on Fig. 1. We also demonstrate how this planner can be directly extended to a frontiers-driven exploration strategy.

A. Related work

Recent exploration works rely on two main types of strategies to maximize the discovery of unknown space: **information-driven** exploration relies on a continuous representation of the environment, such as continuous occupancy maps [5], [7] or signed distance fields (SDF) [1], [25], to estimate the mutual information (MI) between the current map and future measurements; the optimal exploration path can then directly be obtained as the result of a gradient descent in the robot's trajectory space. However, the MI estimation often remains too computationally costly for real-time usage in large environments. On the other hand, **frontiers-driven** strategies rely on the assumption that information is maximized at the boundaries between free and unknown space. Frontiers-driven methods can thus generally be applied directly to binary occupancy data. The frontiers can either be extracted geometrically [6], [29] or obtained locally by searching the map representation, for example with sampling-based methods [30]. The problem can then be regarded as making the optimal choice in a discrete frontiers set. However, the optimality criteria here only applies to the target frontier and not the full path, requiring an additional planning step to provide the path towards the frontier.

Sampling-based planners are the most common choice found in the literature to achieve this task, as they are well-suited to uncertain environments and combine well with frontiers-driven exploration, providing a reduced search space and a natural way to plan towards multiple frontiers. Tree-based approaches, mostly derived from the RRT [13], are fast to compute but do not exploit the information of previous planning calls. Roadmap approaches, inspired by the PRM [10], rather construct a graph representation of free space, which can then be queried using standard graph-search algorithms. They are thus more versatile as paths can be queried independently of the robot location, and a roadmap can be reused for other navigation tasks. Specifically for frontiers-driven exploration, the most critical features of the roadmap are **sparsity**, to avoid computational explosion as the environment (and thus the roadmap graph) grows in size and complexity, and **high coverage** to ensure that any frontier can be navigated to; in particular, finding and exploring

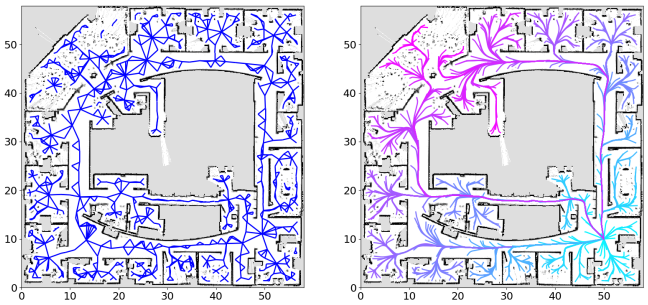

Fig. 1: An example of roadmap obtained using our method. The left figure shows the final roadmap, and the right figure shows the paths planned to each vertex from an arbitrary starting configuration, colored by path length.

[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.

narrow passages is a challenging task for pure sampling-based planners. This can impose a trade-off between query efficiency and path quality, often requiring to post-process the paths planned by the roadmap. It is also necessary for the roadmap to be constructed or updated in real-time as the robot updates the map.

In [4], a generic-purpose sparse roadmap is constructed by allowing degradation of the path length. Sparsity is thus improved in a controlled way without compromising coverage, but induces a direct path-quality trade-off. Other methods rather try to extract higher-level features from the environment to obtain a sparser roadmap: in [28], the obstacles are used to define a tensor field with an underlying graph structure, which is then used directly for guidance towards unexplored areas. Other approaches rather extract the Voronoi diagram associated to the obstacles or similar triangulations [3], [8], [22], [24], [27] to obtain a sparse representation of the map which also encapsulates distance information. However, those methods either operate on simplified map representations (typically on meshes or primitive sets such as segments, arcs) or are difficult to scale to large unstructured environments in terms of computational cost. Another representation with strong links with the Voronoi diagram is the Hamilton-Jacobi skeleton, used as a sparse environment representation in [16] and partially extended to frontiers-driven exploration in [23]. The skeleton allows to encapsulate maximal safety information but still needs to be postprocessed to extract the roadmap graph. Finally, we can also cite approaches relying on bubbles of free-space, where the environment gets covered with a collection of convex, collision-free graph vertices. In [17], the authors plan safety-aware trajectories for UAVs using 3D spheres, sampled around the robot location. We proposed a similar idea for 2D navigation in [18], using a biased sampling method to extend the current bubbles graph. Free-space bubbles have the advantage of providing a direct method to postprocess the paths [20], and can be extended to *burs* of free-space as shown in [12]; however, those methods remain sampling-based, making them more susceptible to failure in large environments with features of varied spatial scale.

B. Proposed approach

To address the challenges posed by autonomous exploration, we propose a novel roadmap planner operating directly on the 2D occupancy grid map (OGM) produced by a standard SLAM algorithm. We first describe how to extract the Hamilton-Jacobi skeleton of the free space; we then use it to sample free-space bubbles forming the roadmap vertices and propose a simple edges construction process, before describing how the roadmap graph can be partially updated as the environment changes. We also detail the associated path post-processing method. We finally propose a frontiers-driven exploration strategy directly derived from the proposed roadmap. An evaluation of the planner is conducted against two state-of-the-art methods and the exploration strategy is demonstrated in various environments. The main contributions proposed in this work can be summarized as:

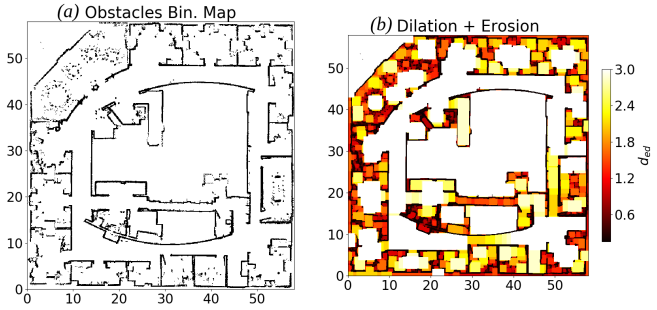

Fig. 2: Occupancy preprocessing - left: binary obstacles map, right: filtered obstacles obtained for various values of the d_{ed} parameter (obstacles dilation and erosion distance). ***Note**: All units on the figures representing 2D maps are in meters.*

- a deterministic roadmap construction method, relying on the map skeleton and on free-space bubbles to obtain a high-coverage, sparse roadmap that naturally enforces safety constraints;
- a statistical evaluation of the proposed roadmap quality against state-of-the-art approaches, in terms of coverage, sparsity, computational performance and path quality;
- a frontiers-based exploration strategy derived from the proposed roadmap, demonstrated in simulated and real environments.

This paper is complemented by a companion video and an open-source implementation of the proposed method[1].

II. Hamilton-Jacobi Skeleton Extraction

In this section, we show how to obtain a high-level representation of the environment, namely its Hamilton-Jacobi skeleton [26], from the occupancy map. We first detail the occupancy preprocessing step used to smooth the map noise, then derive the Euclidean signed distance field (ESDF) associated to the environment, before finally using its gradient to extract the skeleton.

A. Occupancy Map Preprocessing

We rely on a standard OGM as the main input, as it is the most common representation for the environment in many SLAM systems (in the case of a 3D environment, often rather represented as an occupancy octree, an OGM can be obtained simply by projecting the octree on the ground plane). In an OGM, each grid cell represents the occupancy probability p_{occ} that the cell is occupied by an obstacle.

In a real scenario, the OGM data is affected by noise; this can induce undesirable variability in the signed distance field and spurious branches in the final skeleton. To reduce the impact of this noise, we first binarize the occupancy map using a threshold p_{obst}. We then apply successively a dilation and an erosion operations, using the same kernel size for both. This kernel size is determined using a distance parameter d_{ed}, converted to grid-cell size. We thus obtain a binary map with smoother obstacle edges and devoid of small-scale (below d_{ed}) spatial features. Fig. 2 illustrates the effect of the dilation-erosion process for various values of d_{ed}.

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

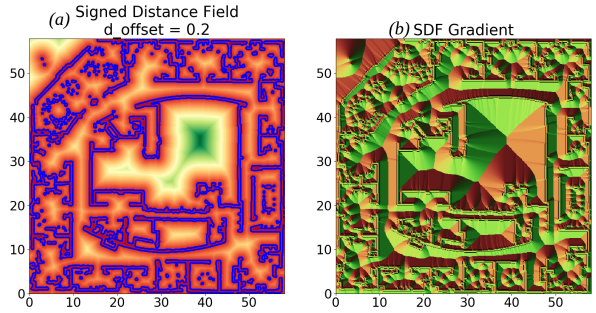

Fig. 3: Signed distance field - left: signed distance field (zero level set in blue), right: SDF gradient (red and green colors indicate the values of the x and y components)

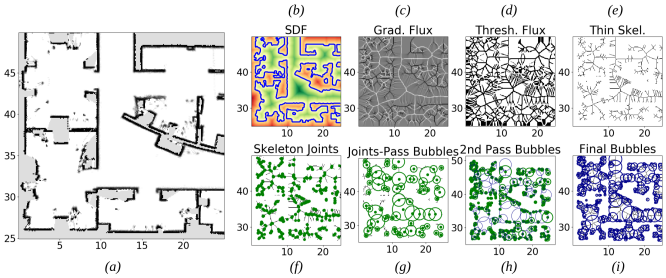

Fig. 4: Summary of the bubbles extraction from the SDF, demonstrated in a sub-part of the Intel research lab environment: from the SDF and its gradient, we compute the gradient flux and threshold it to obtain the Hamilton-Jacobi skeleton; the skeleton is thinned and its joints are extracted; we then extract the free-space bubbles in 2 passes and in radius-descending order.

Remark: We chose the occupancy threshold p_{obst} so that unknown space is considered *a priori* free; this is necessary for the frontiers-based strategy detailed in Sec. IV.

B. Signed Distance Fields

A SDF is defined over the workspace $\mathcal{W}$ as the minimal distance between a given configuration $q \in \mathcal{W}$ and an obstacles subspace, often characterized by its oriented surface or as a set of configurations. When using the standard Euclidean distance $d(u,v) = ||u-v||_2$, the corresponding SDF is often referred to as the exact Euclidean SDF (ESDF). Methods exist in the literature to approximate the ESDF directly from sensor measurements, such as the truncated SDF (TSDF) [19]; however, those methods are often not integrated in standard SLAM systems, on which we rely for the occupancy mapping. Thus, we rather compute the ESDF from an obstacles set $\mathcal{S} = \{q \in \mathcal{W} \mid p_{occ}(q) > p_{obst}\}$. Its formal definition over $\mathcal{W}$ is the following:

$$d_{\mathcal{S}}(q) = \begin{cases} \min_{s\in\mathcal{S}} ||q-s||_2 & \text{if } q \notin \mathcal{S} \\ -\min_{q_0 \in \mathcal{W}\setminus\mathcal{S}} ||q-q_0||_2 & \text{if } q \in \mathcal{S} \end{cases} \quad (1)$$

Computing the full ESDF directly from occupancy is more computationnally expensive than TSDF estimation but can be achieved in linear time with respect to the map size [15]. We additionally define the **witness configuration** $w_{\mathcal{S}}(q)$ associated to configuration q as the closest configuration on the obstacles surface $d_{\mathcal{S}} = 0$ so that $||q - w_{\mathcal{S}}(q)|| = |d_{\mathcal{S}}(q)|$.

C. ESDF Gradient

Coming back to the definition of the ESDF, its gradient can be expressed using the witness configuration as:

$$J_{d_{\mathcal{S}}}(q) = \frac{\partial d_{\mathcal{S}}}{\partial q} = \frac{2(q - w_{\mathcal{S}}(q))}{2\sqrt{(q-w_{\mathcal{S}}(q))^T(q-w_{\mathcal{S}}(q))}} = \frac{q - w_{\mathcal{S}}(q)}{d_{\mathcal{S}}(q)} \quad (2)$$

The ESDF gradient thus corresponds, in free space (resp. occupied space), to the unit direction from (resp. towards) the corresponding witness configuration on the obstacles surface. Since computing the witness configuration is necessary to obtain the complete ESDF, this expression also shows that the gradient is obtained "for free" when computing the ESDF. We can also remark that shifting the obstacles surface by a constant offset d_0 only moves the witness configuration along the normal to the obstacles surface, leaving the gradient unchanged. The ESDF can thus be parameterized with such an offset to enforce safety constraints (typically, d_0 is chosen as the characteristic radius of the robot model, which ensures that all regions with $d_{\mathcal{S}} - d_0 > 0$ are safe for navigation). An example of ESDF and its gradient are shown on Fig. 3.

D. Hamilton-Jacobi Skeleton

We finally extract the Hamilton-Jacobi skeleton which will be sampled to construct the final roadmap. The Hamilton-Jacobi skeleton is strongly linked to the notion of medial-axis transform (MAT): for a closed set $A \subset \mathbb{R}^2$, it is formally defined as the set of centers of maximal open disks contained in the set complement. In our case, it can equivalently be seen as the set of free-space configurations that are equidistant to strictly more than one obstacle configurations. This definition allows to derive a computation method from the ESDF: the skeleton corresponds to the ridges of the ESDF gradient flux. Thus, to extract the skeleton, we compute an estimation of the gradient flux using a local 8-neighborhood; this estimation is thresholded to obtain the initial skeleton map, on which we apply a final thinning step. This process is illustrated on Fig. 4-(b,c,d,e). However, this approach suffers from drawbacks which make it unsuited to direct use in a planning method: in particular, the thresholding step is tuning-sensitive due to the approximation made when computing the ESDF gradient flux. Choosing the threshold too high will lead to a skeleton containing spurious edges, while choosing it too low increases the risk of obtaining a disconnected skeleton. Moreover, the skeleton is still represented as a 2D map, which does not reflect its inherent graph structure and is impractical to query. In the next section, we show how using free-space bubbles allows to overcome tuning sensitivity and obtain a graph representation of the skeleton.

III. Skeleton Disk-Graph Roadmap

In this section, we describe how the Hamilton-Jacobi skeleton map obtained in the previous section is exploited to construct a sparse roadmap, which we refer to as the **skeleton disk-graph roadmap (SDGRM)**. Relying on similar definitions as in [18], we use the skeleton to sample the centers of free space bubbles, ensuring maximal safety distance with respect to the obstacles. The idea behind using

bubbles is to automatically adapt the nodes density of the roadmap to the local SDF. We also detail the post-processing steps applied to the paths planned with our roadmap, which produce smoother paths while retaining safety.

A. Disk-Graph Roadmap Construction

Let us first recall the definition of *free-space bubbles*: a bubble is defined for a configuration $q_0 \in \mathcal{W}$ as a convex subset B in which all configurations are closer to q_0 than the closest obstacle. Formally, this reads as:

$$B(q_0) = \{q \in \mathcal{W} \mid d_{\mathcal{S}}(q_0) > ||q_0 - q||_2\} \tag{3}$$

In the case of a 2D workspace, a bubble is a disk centered on q_0; we also denote the radius of the bubble by $\rho_B(q_0)$. The roadmap we construct uses such bubbles as vertices, leading to a disk-graph structure.

The disk-graph construction consists in two main steps: we first extract the bubbles of free-space associated to the skeleton, ensuring the skeleton is entirely covered by bubbles. We then use an *edge validity criterion* inspired from the ones used in [18] to construct the edges by checking all pairs of bubbles. Let us now detail each of those steps.

Extraction of free-space bubbles. To compute the bubbles constituting the graph nodes, we treat the skeleton map as a pool of samples and pick bubbles in descending radius order to be added to the graph nodes. Each time a new bubble is added, the samples are filtered to eliminate those which are inside the new bubble. This process continues until no more skeleton samples can be chosen so that $\rho_B(q) > \rho_{min}$, where ρ_{min} is a parameter that sets the minimal radius of a free-space bubble. Moreover, this step is actually handled in two separate passes; using the thinness of the skeleton, we can extract the *skeleton joints*, i.e., the pixels in the skeleton map that have strictly more than two neighbors in the sense of 8-neighborhood. The first pass only considers those skeleton joints, as they represent branching configurations and should be prioritized to be part of the final roadmap. The second pass handles the remaining skeleton configurations. The bubbles extraction process is illustrated on Fig. 4-(f, g, h, i).

Edges construction. Having obtained the graph nodes as free-space bubbles, we can now compute the graph edges to obtain our roadmap. Using bubbles provides a convenient way to check for edges, simply by checking overlaps between pairs of bubbles; however, it could occur that two bubbles overlap but the obstacles distance along the segment formed by their two centers passes below the minimal safety distance ρ_{min}. To avoid this case, we include an additional step and compute the geometrical barycenter of the overlap area, which is given by:

$$q_c = q_0 + \frac{1 + (r_0 - r_1)(r_0 + r_1)}{2} \cdot \frac{q_1 - q_0}{||q_1 - q_0||^2} \tag{4}$$

where q_0, q_1 (resp. r_0, r_1) are the centers (resp. radii) of the bubbles considered. The distance map is queried at q_c to verify the safety condition. An edge is added only if $\rho_B(q_c) > \rho_{min}$.

Overall, this graph construction process greatly reduces the sensitivity of the method to the threshold chosen at the previous skeleton extraction step; if the skeleton has spurious edges, they will be eliminated as candidates when contained inside a larger bubble, and if the skeleton misses valid connections, bubbles can overlap over the gaps, allowing for edges in the final graph when relevant.

Planner update. When using the roadmap for exploration, the environment is inherently dynamic, but we can actually avoid recomputing the whole graph for each new SDF; by caching the previous SDF and using a brute force search over existing nodes, we can detect the bubbles whose radius must be modified and they get discarded from the current roadmap. We then extract the skeleton as usual, and mask it with the unchanged bubbles before applying the graph construction process; new bubbles are only sampled in the modified zones and the graph is kept in its current state where applicable.

B. Planning Using the Disk-Graph Roadmap

Once the disk-graph is constructed, standard graph-search approaches allow to plan between any two graph nodes (assuming connectivity). In our case, we use the Dijkstra algorithm. To extend the planning capabilities of the graph to any reachable configurations in the environment, we propose an idea of the proof of completeness in the following.

Planner completeness. Here, we show how for all pairs of configurations for which a feasible path exists in the distance map (under safety constraints), a path can be computed using the proposed disk-graph. Let us first assume connectivity of the graph: disconnected components correspond, by construction, to areas which are isolated in the distance map. We also consider the problem from the point of view of only one of the path ends (start or goal configuration) because it is symmetric, and assume they both satisfy the safety distance constraints. We have two possible cases :

- if the goal configuration lies inside one of the graph bubbles, we can plan directly to the corresponding center configuration; we can then add a final path segment to the goal, which is guaranteed to be free.
- if the goal configuration satisfies the safety constraints but lies outside of the graph bubbles, we can not directly plan towards one of the graph vertices. However, starting from the goal configuration, we can iteratively follow the distance gradient direction towards the skeleton. Since by construction, the skeleton is entirely covered by free-space bubbles, we are guaranteed to enter a free-space bubble at some point along the gradient walk. We can then apply the same argument as before and thus guarantee that a feasible path exists.

In practice, the second case rarely happens, because the robot navigation is guided by paths remaining inside the bubbles.

C. Path Post-Processing

The path obtained is a list of waypoints corresponding to the positions of the roadmap nodes, and is thus a succession of segments linked by zero-radius turns. However, since the nodes are overlapping bubbles, we can use the path-smoothing technique proposed for the Elastic-Bands local

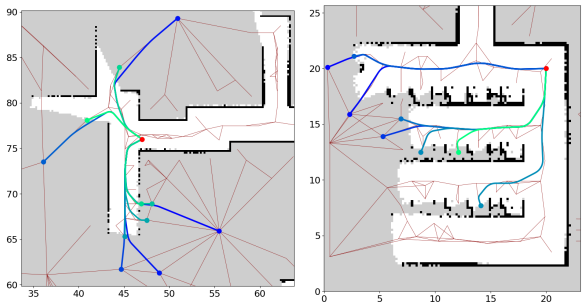

Fig. 5: Frontiers extraction at 2 different planning steps during the exploration of the CMU indoors environment. The roadmap is in dark red, the current agent position in brighter red, and each frontier and the path towards it are colored by path length. The maximum path length d_{search} is set to $20m$.

planner [20], [21]. Doing so, the path is approximated using a B-spline, for which the control points are directly computed from the bubbles positions and radii. This spline is guaranteed to remain inside the bubbles, maintaining safety. We note that for robots with non-zero turning radius, there is *a priori* no guarantee on the feasibility of the post-processed path. However, the B-spline expression allows to compute the curvature in closed form at any point on the path, already giving a verification method for a given path. Modifying the control points associated to each bubble or the B-spline expression could help constraining the minimal curvature of the path, but we did not investigate this aspect further here.

IV. Exploration Strategy

In this section, we adopt the standard frontiers-driven paradigm for exploration, showing how our roadmap can be used to extract and order environment frontiers for efficient exploration. We also detail the navigation aspect to provide a complete exploration pipeline.

A. Frontiers Extraction

Frontiers are classically defined in an occupancy map as the free cells neighboring unknown cells; however, extracting those frontiers can become costly when the OGM grows in size. However, we can rely directly on the roadmap nodes to define frontiers more efficiently. To do so, we define candidate frontier nodes as the bubbles for which the known surface they cover is below a threshold s_{thresh}. We then check the neighbor nodes of those candidates; if at least one is centered on a known free space cell, we know that the edge linking them crosses the environment frontiers, and thus append the candidate to the final frontiers set. This process is illustrated on Fig. 5. We can note that some of the identified frontiers are located far into unknown space; this can occur when a small free bubble and a large unknown one are linked by a frontier-crossing edge, since we actually use the center of the unknown bubble as the frontier candidate.
Additionally, the frontiers search is limited to a local subgraph: denoting $L(q_0, q_{goal})$ the length of a path between the current robot position q_0 and a given goal, this subgraph is defined as $G_{search} = \{v \in G \mid L(q_0, v) < d_{search}\}$, d_{search} being a parameter of the strategy. In case no frontier is found in G_{search}, we linearly increment d_{search} with the number of failed planning attempts (i.e., $d_{search}(n) = n \cdot d_{search}(0)$ where n is the index of the current attempt, assuming all previous ones failed). We thus obtain a list of k frontier configurations denoted by $f_1, \dots, f_k$.

B. Best Frontier Selection

To select the best frontier among those available, we start by using the roadmap to plan the corresponding paths towards each frontier f_i, which we denote by $\Gamma_1, \dots, \Gamma_k$. We then compare those paths on two simple criteria:

- **energy cost** c_E: frontiers that are costly to reach are penalized. In practice, for 2D navigation, we use the path length as a direct proxy for the energy cost.
- **expected reward** r: we also evaluate the expected reward along the path using the occupancy map. To do so, we use the bubbles that are part of the path to mask the occupancy map and compute the total unknown area inside this mask.

Both get normalized and we use a weighted linear combination $c_{tot}(\Gamma_i) = \alpha_E \cdot c_E(\Gamma_i) + \alpha_r \cdot r(\Gamma_i)$ to select the lowest-cost path as the new exploration path. In practice, we generally set $\alpha_E = 1$, $\alpha_r \in [0.1, 0.5]$ to always favor the path length criterion and avoid excessive backtracking.

C. Navigation

The navigation is handled by a standard non-linear path-following controller [2]; we enforce two interruption conditions, when the path no longer satisfies the distance safety threshold or when the bubble centered on the current path end q_{goal} becomes entirely known.
Additionally, since the robot we use is capable of zero-radius turns, a safety condition limits the turning radius: if the robot gets closer to the obstacles than a threshold d_{safe}, the linear velocity command is reduced to achieve a turning radius below $0.5 \cdot d_{\mathcal{S}}(q)$, based on the current angular velocity.

V. Experimental Results

In this section, we first evaluate our method on a standard roadmap construction task, in a fixed environment. We then consider a more realistic exploration task, where the environment is initially unknown, and apply the strategy presented in Section IV, both in simulated and real environments.

A. Roadmap Quality

Quality metrics. To evaluate the quality of the roadmap obtained with our method, we focus on two main aspects: *roadmap graph quality*, which reflects the desirable properties of the roadmap graph, and *path quality*, which reflects the quality of the paths obtained when planning using the roadmap. The graph metrics we consider are the following:

- **reachability**: we compute reachability as the proportion of valid paths planned using the roadmap graph for a uniformly sampled set of start-goal pairs in free-space.
- **sparsity**: as a proxy for the sparsity of the roadmap graph, we use the ratio of the number of edges and number of nodes.

Fig. 6: Roadmap quality is evaluated in environments of varying complexity: (a) successive bugtraps (800×800px), (b) Freiburg building (368×911px), (c) Intel research lab (581×579px), (d) CMU indoors (670×516px), (e) CMU tunnels (1611×1367px), (f) large-scale maze (1204×1204px)

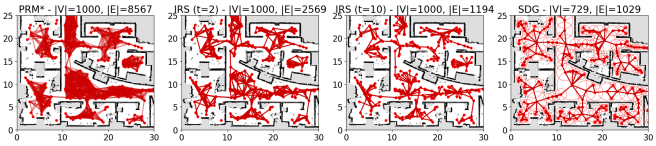

Fig. 7: Aspect comparison of the evaluated roadmaps - detail of the Intel research lab environment, with the total number of vertices equivalent between SDGRM and uniform sampling approaches. Note that the samples used are the same for PRM* and the two versions of IRS.

We also evaluate the construction time for each roadmap.

For the path quality evaluation, we standardly consider the length of the obtained path as the main performance indicator; moreover, using the distance skeleton that is extracted to construct the roadmap, we introduce a safety metric computed as the integrated distance to the skeleton along the path. Formally, for a path $\Gamma : [0,1] \rightarrow \mathcal{W}$ and defining $w_{sk}(q)$ the witness configuration on the skeleton associated to $q \in \mathcal{W}$, the safety metric reads as:

$$S(\Gamma) = -\int_0^1 ||w_{sk}(\Gamma(t)) - \Gamma(t)||\, dt \tag{5}$$

In practice, we compute it by discretizing the path with a fixed step size.

Evaluation baselines. To show how our method improves the sparsity and safety of the final roadmap while retaining very good coverage and close to optimal path lengths, we compare it against two standard roadmap approaches: the PRM* [9], an asymptotically-optimal version of the PRM, which provides close-to-optimal paths but a very dense final roadmap; and the Incremental Roadmap Spanner (IRS) [14], a sparse roadmap approach which aims at reducing the density of the PRM* roadmap while controlling the path optimality compromise with a stretch factor guiding the sparsification. In our evaluation experiments, we used two IRS roadmaps with stretch factors of 2 and 10, to better illustrate the trade-off between sparsity and path quality. A visual comparison of those baselines with our method is displayed in Fig. 7.

Experimental setup. The test environments used for the roadmap quality evaluation are displayed in Fig. 6. The PRM* and IRS roadmaps were constructed 10 times for each maximal number of vertices; for the SDGRM, only one construction is evaluated since the result is deterministic. Reachability is evaluated for each roadmap using 1000 start-goal pairs selected randomly in free-space. The results are presented in Fig. 8-a : for all six environments, our method achieves close to optimal reachability; moreover, for equivalent performance, it contains far less vertices than any of the proposed baselines. The performance gain of SDGRM is particularly clear in the two largest environment (*CMU Tunnels*, *Large Maze*): the reason for it seems to be the narrowness of the passages in those, making PRM* and IRS more susceptible to disconnections due to the uniform sampling approach they build on.

The sparsity results are presented in Fig. 8-b, aggregated in all environments as a function of the reachability. We observe that the proposed baselines are mostly consistent over all environments; PRM* always presents the highest relative number of edges to achieve high reachability results, while IRS-2 and IRS-10 achieve much better sparsity scores; in particular, for IRS-10, the sparsity of the roadamp seems to be mostly constant (slightly above 1), independently of the reachability performance. For SDGRM, the sparsity score outperforms PRM* and IRS-2 while also being consistently below 2, independently of the environment.

Fig. 9 presents the construction time evaluation in the same conditions as the reachability and sparsity evaluation; we first observe that in all environments, our method achieves faster construction times than the proposed baselines. PRM* seems to have linear time complexity with respect to the number of vertices, which makes sense as this simply scales the number of pairwise checks to add the edges; on the other hand, the performance of IRS is much more degraded as the roadmap grows in size. This is due to the fact that IRS achieves sparsity by querying the partial roadmap during construction to check if new vertices should be added; in large environnments such as the maze, this leads to computational explosion and makes IRS practically unusable. Fig. 9 also shows in more details the breakdown of the computational cost for our method, showing that the radius-descending sampling of the skeleton and the pairwise overlaps checks to add the edges have a similar time cost. Overall, the results presented in Fig. 8 and 9 show that our method surpasses the proposed baselines in terms of reachability and sparsity, also achieving these better properties within a shorter construction time. Taking into account that higher sparsity also makes subsequent queries of the roadmap faster, our method presents a real interest in terms of execution time.

For the path quality evaluation, the baselines are evaluated as before on 1000 random start-goal pairs, taking into account only the pairs for which all planners return a valid path.

For each valid task, the results are normalized by the best score achieved for this task. For SDGRM, we distinguish the results between the raw paths returned by the roadmap

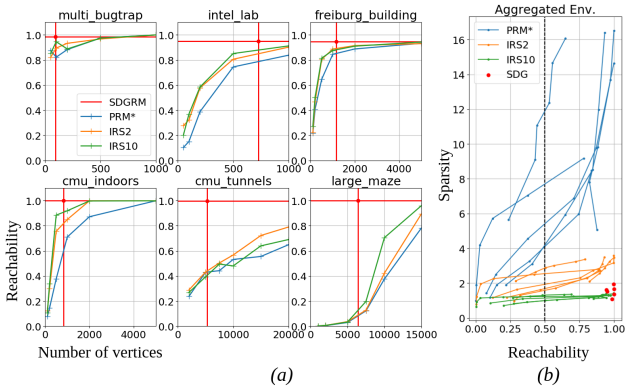

Fig. 8: Roadmap graph quality evaluation: *(a)*: reachability as a function of number of vertices in each of the test environments, *(b)*: graph sparsity results, as a function of reachability, aggregated over all environments.

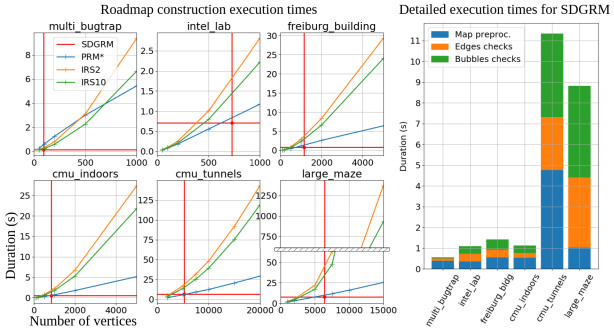

Fig. 9: Roadmap construction time: *(Left)* Duration of the roadmap construction as a function of number of vertices in the same conditions as for the reachability evaluation, *(Right)* Detailed breakdown of the execution time for SDGRM in each environment

and the paths after smoothing. Results are summarized in Table I: as expected, PRM* consistently plans the paths with optimal length; however, the results for SDGRM and SDGRM-S remain close to optimal, especially when considering the smooth paths. In terms of safety on the other hand, SDGRM clearly outperforms the proposed baselines: using the skeleton to construct the roadmap reflects its optimal safety properties in the paths it produces. Overall, the results of Fig. 8 and Table I show that SDGRM covers most of the workspace, producing paths with greatly improved safety while remaining close to optimal in terms of length.

B. Exploration Results

Finally, we present the results obtained with the exploration strategy proposed in Section IV. The full system is ROS-based and consists in a Pioneer mobile robot equipped with a 2D LiDAR and an RGB-D camera. To construct the environment map, we use a standard 3D SLAM system (namely, RTAB-Map SLAM [11]). In this work, we only use the 2D floor projection of the 3D octomap as an input to the roadmap; however, using the RGB-D data in combination with the 2D laser data helps obtaining an overall better estimation of the map. In addition to the real robot, we use a simulator developed in Unity that provides realistic 3D environments and sensor data. Five experimental setups are demonstrated: two large-scale environments in simulation and three more "corridor-like" real environments. The

TABLE I: Normalized path length and safety on 1000 start-goal pairs: for each start-goal task, both metrics are computed and normalized by the best result obtained across all 5 roadmaps. The table presents the averages and standard deviations of those normalized metrics for each planner. The SDGRM-S entries in the table correspond to the same roadmap as for SDGRM, but including the post-processing step on the obtained paths.

Env.	Roadmap	Norm. Length	Norm. Safety
Multi Bugtrap	PRM*	$\mathbf{1.000 \pm 0.003}$	4.81 ± 2.12
	IRS-2	1.11 ± 0.03	5.10 ± 2.20
	IRS-10	1.42 ± 0.11	7.35 ± 2.93
	SDGRM	1.24 ± 0.12	$\mathbf{1.01 \pm 0.09}$
	SDGRM-S	1.10 ± 0.06	1.47 ± 0.46
Intel Lab	PRM*	$\mathbf{1.0 \pm 0}$	2.43 ± 0.61
	IRS-2	1.04 ± 0.05	2.45 ± 0.50
	IRS-10	1.20 ± 0.05	2.97 ± 0.61
	SDGRM	1.10 ± 0.04	$\mathbf{1.03 \pm 0.06}$
	SDGRM-S	1.06 ± 0.03	1.04 ± 0.05
CMU Indoors	PRM*	$\mathbf{1.0 \pm 0}$	4.58 ± 0.66
	IRS-2	1.04 ± 0.01	4.85 ± 0.75
	IRS-10	1.16 ± 0.03	5.20 ± 0.79
	SDGRM	1.09 ± 0.02	$\mathbf{1.00 \pm 0.01}$
	SDGRM-S	1.06 ± 0.01	1.32 ± 0.15

exploration trajectories are presented in Fig. 10. They are complemented by additional metrics provided in Table II, consisting in total travel distance L, total explored surface $\mathcal{A}$, average efficiency ε, i.e., the ratio of explored surface and travel distance, and number of planning calls n_P.

We first observe a desirable tendency to produce non-overlapping trajectories, except when reaching dead ends and backtracking in known space. A side effect of this property though, is to produce few opportunities for loop closures for the SLAM algorithm. This in turn reduces the overall map quality when sensor noise is present. An option to mitigate this might be to encourage loop closures with an additional path selection criteria.

Overall, the exploration trajectories produced by our method are mostly smooth thanks to the path-smoothing step applied after planning, and always remain close to the skeleton. Moreover, the proposed strategy tends to favor the current direction of progress, in the sense that the robot will significantly alter its current goal only when reaching dead ends or intersections with multiple unexplored frontiers.

VI. Discussion and Future Work

In this paper, we presented a novel method to construct a sparse, high-coverage roadmap, capable of providing high-quality navigation paths. We also showed how it can be directly exploited for frontiers-driven exploration in various scenarios. Currently, our approach remains limited to 2D navigation. Even though generic motion-planning for high-dimensional robots is out of scope of this work, we believe an extension to 3D navigation of our method would be feasible,

TABLE II: Exploration metrics

Env.	L (m)	$\mathcal{A}$ (m^2)	ε (m^2/m)	n_P
CMU Indoors	1121	3066	2.8	173
CMU Tunnels	5808	15144	2.6	898
Lab. Corridor	132	244	1.83	-
Parking Lot	101	1107	11.6	18
Lab. Patio	157	230	1.5	99

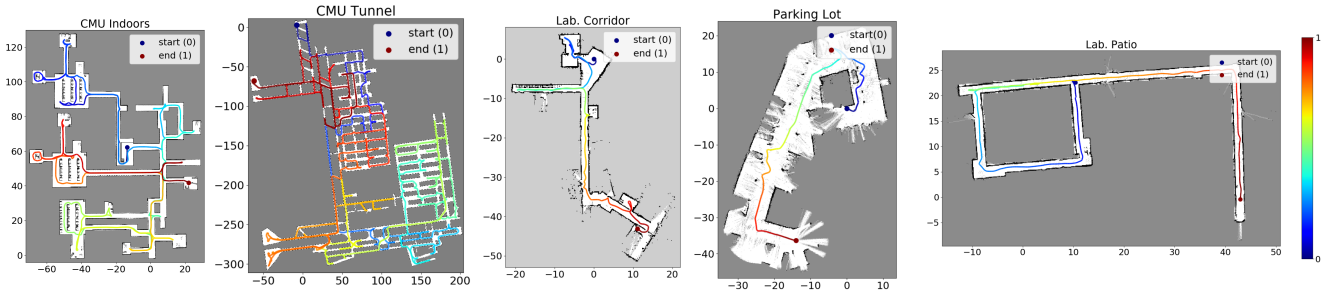

Fig. 10: Exploration trajectories obtained with our method in two simulation (CMU Indoors, CMU Tunnels) and three real (Lab. Corridor, Parking Lot, Lab. Patio) environments. Each is displayed in the final best estimate of the map.

especially in regard of the recent works done on SDF by the computer graphics community. The main difficulty is to adapt the skeleton extraction process in 3D, but the rest of the method remains the same. Another limitation is the usability of the 3D data obtained with our exploration method; currently, our approach produces a complete occupancy map, but this does not translate to the pointcloud which is only partial at the end of exploration. In future works, we plan to focus on 3D-informed camera control to also maximize 3D data acquisition, opening possibilities for more complex tasks such as autonomous 3D reconstruction.

REFERENCES

[1] A. Asgharivaskasi, S. Koga, and N. Atanasov. Active mapping via gradient ascent optimization of shannon mutual information over continuous se (3) trajectories. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 12994–13001, 2022.

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

[3] H. Choset, S. Walker, K. Eiamsa-Ard, and J. Burdick. Sensor-Based Exploration: Incremental Construction of the Hierarchical Generalized Voronoi Graph. *Int. J. of Robotics Research*, 19(2):126–148, Feb 2000.

[4] A. Dobson and K. Bekris. Sparse roadmap spanners for asymptotically near-optimal motion planning. *Int. J. of Rob. Research*, 33:18–47, 2014.

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

[6] W. Gao, M. Booker, A. Adiwahono, M. Yuan, J. Wang, and Y. W. Yun. An improved frontier-based approach for autonomous exploration. In *Int. Conf. on Control, Automation, Robotics and Vision*, pages 292–297, 2018.

[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] M. Kallmann. Dynamic and robust local clearance triangulations. *ACM T. on Graphics*, 33(5):1–17, 2014.

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

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

[11] M. Labbé and F. Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[12] B. Lacevic, D. Osmankovic, and A. Ademovic. Burs of free c-space: A novel structure for path planning. In *IEEE Int. Conf. on Robotics and Automation*, pages 70–76, 2016.

[13] S. LaValle. Rapidly-exploring random trees: a new tool for path planning. *Technical Report - Computer Science Department, Iowa State University (TR 98–11)*, 1998.

[14] J. Marble and K. Bekris. Asymptotically near-optimal planning with probabilistic roadmap spanners. *IEEE T. on Robotics*, 29(2):432–444, 2013.

[15] A. Meijster, J. Roerdink, and W. Hesselink. A General Algorithm for Computing Distance Transforms in Linear Time. In *Mathematical Morphology and its Applications to Image and Signal Processing*, volume 18, pages 331–340. Kluwer Academic, 2002.

[16] J. Modayil, P. Beeson, and B. Kuipers. Using the topological skeleton for scalable global metrical map-building. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, volume 2, pages 1530–1536, 2004.

[17] T. Musil, M. Petrlík, and M. Saska. Spheremap: Dynamic multi-layer graph structure for rapid safety-aware uav planning. *IEEE Robotics and Automation Letters*, 7(4):11007–11014, 2022.

[18] T. Noël, S. Kabbour, A. Lehuger, E. Marchand, and F. Chaumette. Disk-Graph Probabilistic Roadmap: Biased Distance Sampling for Path Planning in a Partially Unknown Environment. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2022.

[19] 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.

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

[21] 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.

[22] R. Ramachandran, S. Wilson, and Sp. Berman. A probabilistic approach to automated construction of topological maps using a stochastic robotic swarm. *IEEE Robotics and Automation Letters*, 2(2):616–623, 2017.

[23] M. Rezanejad, B. Samari, I. Rekleitis, K. Siddiqi, and G. Dudek. Robust environment mapping using flux skeletons. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems)*, pages 5700–5705, 2015.

[24] G. Roland and M. Overmars. Enhancing corridor maps for real-time path planning in virtual environments. computer animation and social agents. *Computer Animation and Social Agents*, pages 64–71, 2008.

[25] K. Saulnier, N. Atanasov, G. J. Pappas, and V. Kumar. Information theoretic active exploration in signed distance fields. In *IEEE Int. Conf. on Robotics and Automation*, pages 4080–4085, 2020.

[26] K. Siddiqi, S. Bouix, A. Tannenbaum, and S. Zucker. Hamilton-Jacobi Skeletons. *Int. Journal of Computer Vision*, 48:215–231, 2002.

[27] E. G. Tsardoulias, A. T. Serafi, M. N. Panourgia, A. Papazoglou, and L. Petrou. Construction of Minimized Topological Graphs on Occupancy Grid Maps Based on GVD and Sensor Coverage Information. *Journal of Intelligent & Robotic Systems*, 75(3):457–474, 2014.

[28] K. Xu, L. Zheng, Z. Yan, G. Yan, E. Zhang, M. Niessner, O. Deussen, D. Cohen-Or, and H. Huang. Autonomous reconstruction of unknown indoor scenes guided by time-varying tensor fields. *ACM Trans. on Graphics*, 36(6):1–15, 2017.

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

[30] H. Zhu, C. Cao, Y. Xia, S. Scherer, J. Zhang, and W. Wang. Dsvp: Dual-stage viewpoint planner for rapid exploration by dynamic expansion. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 7623–7630, 2021.