GNGraph: Self-Organizing Maps for Autonomous Aerial Vehicle Planning

Edwin Paúl Herrera-Alarcón, Massimo Satler, Marco Vannucci, and Carlo Alberto Avizzano

***Abstract*—The present letter tackles the problem of planning a collision-free path in a known environment from a general point of view. We address the problem by using an unsupervised learning algorithm to generate a sparse graph representing the topological structure of the environment and use it for planning paths in 3D spaces. We propose GNGraph, an integrated solution combining the Growing Neural Gas algorithm to generate the sparse graph, a stop criterion to guarantee the graph's connectivity and a collision check to assess the edges and nodes validity. The proposed solution has been tested on simulated and real environment maps, and compared against a state-of-the-art graph planning algorithm among other global planning methods.**

***Index Terms*—Aerial systems: perception and autonomy, aerial systems: applications, AI-based methods, autonomous vehicle navigation, integrated planning and learning.**

I. Introduction

MICRO Aerial Vehicle (MAV) exploration in cluttered obstacle environments is a challenging problem that demands obstacle detection and planning feasible trajectories in real-time. On the other hand, map-based navigation is a problem that relies on map learning, localization, and path planning. Environment exploration and map-based navigation are two fundamental problems in robotics, which are profoundly related and consequential. While the exploration in cluttered environments is interested in enhancing the volumetric representation of the surroundings, map-based navigation is keen to use the previously obtained information for fast and secure planning among the mapped space. From the solution to both problems, several civil applications have been satisfied, like industrial inspection [1]–[3], surveillance and object-oriented exploration [4], [5], 3D surface building reconstruction [6], precision agriculture [7], to mention a few.

In this letter, we depict a scenario where the MAV has an exploration mission to build a map of the space where it will operate, e.g., infrastructure inspection [2] or building reconstruction [8] applications, and then reuse this map for fast and efficient planning. Considering the movement versatility of MAVs, they have the potential to reproduce a detailed volumetric map of a scene in a relatively short time. However, their limited flying autonomy demands necessary pit stops during a real-life mission. Therefore, it is mandatory to return to the home station and then return to continue its mission from the *last checkpoint*.

Werner et al. [9] presented how it is possible to model in a simple formalism the essential notions of navigational knowledge using route graphs. While, Collins et al. [10] evidenced the benefits, both computationally and memory-efficient, of describing the global map through a sparse topological graph.

Self-organizing Map (SOM) algorithms adapt their structure to fit high-dimensional input data into a lower-dimensional space while preserving the data's topology. Because of these essential features, SOM algorithms have been widely used for data clustering and learning data distributions.

The proposed work focuses on creating a compact, low-dimensional representation of 3D spaces through a sparse topological graph. The proposed solution exploits an unsupervised Hebbian learning algorithm and is devised for a fast run-time global planning query on low-computing power embedded systems typically used on MAVs.

The Growing Neural Gas (GNG) algorithm [11] uses graphs as network structures, where each node has an associated weight vector corresponding to the node's position in the input space. GNG is an unsupervised clustering algorithm that maps arbitrary data used for training into a (often much) smaller set of clusters preserving their original distribution and topology. These features are fundamental in topology learning tasks since they favor the generation of a higher number of clusters in the regions of the domain where original data are denser (distribution preservation) and a comparable spatial localization (topology preservation). The main advantage of GNG over the SOM is that this latter approach requires the *a priori* setting of the number of clusters to be generated, which may affect the quality of the final clusterization. With GNG, the number of clusters is continuously updated and eventually grown until a user-defined upper limit. Our method exploits the relationships established by the network between nodes to generate the edges and the algorithm's growing characteristic to ensure a reduced dimensionality concerning the input data. The proposed algorithm exploits a Euclidean Signed Distance Field (ESDF) map of an environment to extract its 3D Generalized Voronoi Diagram (GVD). Subsequently, the diagram is pruned to generate a sparse graph that represents the environment's topology. The algorithm's efficiency is evaluated by analyzing the sparse graph and testing its performance for planning purposes.

The contributions of this work are the following:

1) a novel approach based on an unsupervised learning algorithm for the generation of a sparse graph representing the topological skeleton of 3D spaces;

Manuscript received 24 February 2022; accepted 9 July 2022. Date of publication 1 August 2022; date of current version 9 August 2022. This letter was recommended for publication by Associate Editor R. Liu and Editor A. Banerjee upon evaluation of the reviewers' comments. *(Corresponding author: Edwin Paúl Herrera-Alarcón.)*

Edwin Paúl Herrera-Alarcón, Massimo Satler, and Carlo Alberto Avizzano are with the Perceptual Robotics Laboratory at the IIM Institute, Department of Excellence in Robotics and A.I., Scuola Superiore Sant'Anna, 56100 Pisa, Italy (e-mail: e.herreraalarcon@santannapisa.it; m.satler@sssup.it; carlo@sssup.it).

Marco Vannucci is with the ICT-COISP at the TeCIP Institute, Department of Excellence in Robotics and A.I., Scuola Superiore Sant'Anna, 56100 Pisa, Italy (e-mail: m.vannucci@santannapisa.it).

Digital Object Identifier 10.1109/LRA.2022.3195192

2) the definition of a criterion to assess graph connectivity as well as to be used for effectively stopping the iteration of the learning algorithm;
3) a comparison against a state-of-the-art graph-based algorithm and other global planning algorithms in simulated and real environments.

The remaining of this work is organized as follows: Section II provides an overview of different solutions used to extract the topology of the scene for planning. In Section III, the problem is stated and the proposed approach is described, while experiments are depicted and analyzed in Section IV. Finally, Section V summarizes the results and future project developments.

II. Related Work

Map-based navigation involves three processes: map-learning, localization, and path planning. Filliat et al. [12] describe the map-learning and localization as interdependent tasks in which building a map requires the position to be estimated relative to the partial map learned until that moment, while the localization requires the use of a feature map that already exists. On the other side, path-planning is a relatively independent process that takes place once the map has been built and the system's position estimated.

Thrun [13] evidenced the complexity between producing highly-accurate maps and efficiently approaching the planning problem afterward, proposing a method that integrates grid-based mapping with topological mapping. His work applied topological grid-based planning in ground robots, where he built the topology of the map from the GVD, focusing on *critical points* (points on the GVD that minimize local clearance) to divide the map into separate regions.

Since then, several works have been improving the generation of GVDs for spatial representations and planning. Kalra et al. [14] proposed the Dynamic Brushfire algorithm, which uses path planning techniques to update the GVD when the underlying environment changes or when new information concerning the environment is received. Lau et al. [15] extended this approach creating a one-voxel-thin GVD that better represented the topology of the underlying space while using the Euclidean Distance Maps (DM) with low computational cost. Kim et al. [16] followed the method proposed by Lau et al. [15] for MAV planning to recreate a 2D GVD from ESDF to maintain efficient planning computation under the assumption that flight altitude would not change drastically. Also, Fang et al. [17] stuck to an approach using 2D GVD complemented with 3D sampling for MAV navigation.

More recently, Oleynikova et al. [18] [19] introduced an approach to obtain the 3D GVD from ESDF and using it for 3D aerial vehicle planning. Their approach is based on Foskey et al. [20], Simplified Medial Axis (θ-SMA), for the calculation of all points belonging to the medial axis (skeleton). This approach is used by Rosinol et al. [21] in the Kimera Project, a spatial perception engine for scene representation, to extract the topological graph of places. Each node samples the free space in this project while the edges denote traversability among nodes.

In addition, fascinating work has been done related to the generation of a sparse graph while exploring the environment. Collins et al. [10] presented a memory-efficient map built from the robot's spatial history to generate a global knowledge of the environment while exploring it. Even if this approach is highly efficient, it learns only the topology of the space, limiting an exploration mission to obtain only feasible paths rather than a volumetric representation of the environment. Similarly, Shah et al. [22] built a topological map using observations obtained from previously collected data for visual goal navigation.

Unsupervised learning methods have been used for different tasks related to autonomous agents. Arleo et al. [23] used unsupervised Hebbian learning to build a representation of the environment while exploring it for localization. Their representation is a continuous 2D manifold from a high-dimensional input space that combines idiothetic and allothetic information. Best et al. [24] and Ma et al. [25] used it for perception and task allocation in a multi-robot system to maximize data collection. Moreover, several works have focused on using GNG while exploring [26], [27]. However, just a few approaches used it to describe the traversable space of an environment for navigation [28], [29].

III. Proposed Approach

The proposed method can be divided into three main steps to generate a graph that can be used to plan at run-time on the system. The first step consists in the computation of the medial axis starting from the ESDF map of the environment. The medial axis represents the set of points with the characteristic of having more than one closest point to the collision boundary. The calculation has been done following the approach proposed by Oleynikova et al. [19].

Then, the self-adaptation phase is based on a competitive learning algorithm to generate a raw graph from the points previously evaluated. It is worth noting that, since the graph is built using medial axis candidates from the ESDF, it will encode the geometry of the scene. To stop the "self-organizing" learning algorithm, we choose to analyze the spectrum of the growing graph. Due to the characteristics of the Fiedler Value (λ_2), it is a plausible criterion which is able to identify minimal connectivity while assuring the generation of one component graph. Connectivity in graph-based planning is essential because no solution can be found if the starting and goal positions rely on separate subgraphs.

Finally, a finishing phase to adapt the raw sparse graph to a collision-free sparse graph taking into the account the drone size.

A. Problem Description

Given a graph $G = (V, E)$, where $V \in \mathbb{R}^n$ is a set of nodes or vertices, and E is a set $\{u, v\} \in V$, called edges. The reduced graph associated to G is the graph where no self-loops are allowed as edges, $E = \{u, v\} \in V : u \neq v$. In addition, for every pair of nodes, there is at most one edge between them. Graphs with these characteristics are often called *simple* graphs.

We consider two other characteristics for the graph: to be undirected and connected. *Undirected* graphs are a type of graph where no direction is associated with the edges, allowing to traverse them in both directions. On the other hand, a graph is *connected*, if for each pair of vertices $\{u, v\}$ there is at least one path that links them.

The problem addressed in this work refers to the generation of a simple and connected undirected graph $G = (V, E)$, which represents the 3D topological skeleton of a map for planning. In shape analysis, the topological skeleton is a thin version of a shape equidistant to its boundaries. The skeleton relies on the

collision-free space C_{free}, of the explored volume $V_{exp} \subset \mathbb{R}^3$. Therefore, this graph is a subset of the collision-free space of an explored volume, allowing fast and efficient planning.

B. Medial Axis Discretization

The medial axis of a shape is the set of all points having more than one closest point on the object's boundary. It is often called the topological skeleton because it is a minimum-unit wide skeleton of the shape, with the same topology as the original object.

As previously mentioned, we used the method presented in [19] to obtain a discretization of the free space. The map's discrete skeleton is built by iterating through the voxels that belong to the ESDF to find ridges of an angle function. A point belongs to the ridge if its 26-connectivity neighbours have parent vectors with a minimum angle separation.

C. Delaunay Triangulation

Given a finite set of points in $S \in \mathbb{R}^d$ called sites, the Voronoi diagram is the partition of S into regions, where a region is defined by all points closer to a specific site $s \in S$ than to any other site. The dual of the Voronoi diagram, called Delaunay triangulation, is a unique triangulation so that the circumsphere of every triangle contains no sites in its interior. The Delaunay triangulation is a technique for creating a mesh of continuous and non-overlapping triangles from a discrete input set of points.

D. Competitive Hebbian Learning

Martinetz et al. [30] defined the induced Delaunay triangulation as a Delaunay triangulation with data distribution P. Two points are only connected if the common border of their Voronoi polygons lies, at least partially, in a region where $P(\upsilon) > 0$, where υ denotes the input data whose pattern is aimed to be learned.

Martinetz [31] introduced an approach, based on a Hebbian Learning rule, to build a graph. Hebb's rule [32] specifies how much the weight of the connection between two units should be increased or decreased in proportion to the product of their activation due to a specific input. Martinetz [31] incorporated this concept to generate lateral connections among units that are initially unconnected. In this approach, competition is brought in for the generation of lateral connections instead of among units, such that only strong, repetitive connections remain. Furthermore, it is demonstrated that CHL creates a connectivity structure among units that corresponds to the Induced Delaunay Triangulation if the distribution of the input set of points is dense. Moreover, if the aforementioned input is dense on a given manifold, the induced Delaunay triangulation preserves the topology of the given manifold.

E. Growing Neural Gas Algorithm

Fritzke [11] presented the GNG algorithm as a method to build Induced Delaunay Triangulation structures using a "Neural Gas"-alike approach for vector quantization. However, opposite to the Neural Gas algorithm [33] the method is incremental and features constant parameters.

GNG uses a growing mechanism for gradual adaptation and self-adjustment of its size, as it can be seen in Algorithm 1. The growing neural network starts from a minimal number of neurons state, which will increasingly adapt to the input data. The adaptation process will continue until a stopping criterion is reached.

Algorithm 1: Growing Neural Gas.

Initialize G by two nodes with random weight vectors
$c \leftarrow \emptyset$, $s \leftarrow \emptyset$
$\xi \leftarrow$ Shuffled Medial Axis Candidates
$s \leftarrow s + 1$
$v, \mu \leftarrow TWO\_NEAREST\_NODES(\vec{\xi})$
foreach *n* ***in*** N_v **do**
  $A_{n,v} \leftarrow A_{n,v} + 1$
end
$INC\_ERROR(c, s, v, \|\vec{w_v} - \vec{\xi}\|^2)$
$\vec{w_v} \leftarrow \vec{w_v} + \epsilon_b(\vec{\xi} - \vec{w_v})$
$\vec{w_n} \leftarrow \vec{w_n} + \epsilon_n(\vec{\xi} - \vec{w_n}), \forall n \in N_v$
Create an edge between v and μ
$A_{v,\mu} \leftarrow 0$
foreach *a,b* ***in*** *all edges in map* **do**
  if $A_{a,b} > A_{MAX}$ **then**
    Delete edge connecting a and b, also all nodes w/o edges.
  end
end
if $s = \gamma$ **then**
  GNG_NEW_NODE
  $c \leftarrow c + 1$
  $s \leftarrow 0$
end
$DEC\_ALL\_ERROR(\beta)$
if *Stopping Criterion Met* **then**
  break
else
  Go to step 3
end

Considering the configuration space $\mathcal{C}$ of the drone within the map is divided into two disjoint sets: an open set $\mathcal{C}_{free}$ containing the robot's poses that do not intersect objects and a closed set $\{\mathcal{C}_{obs} = \mathcal{C} \backslash \mathcal{C}_{free}\}$. The medial axis discretization is a set of points distributed densely on $\mathcal{C}_{free}$. Hence, by applying GNG to the medial axis, a resultant graph G is obtained, which corresponds to a particular induced Delaunay triangulation, where each edge lies inside the collision-free space. This graph can be considered as a discrete, path preserving, representation of $\mathcal{C}_{free}$ where each node is associated with a weight, which represents its position in the 3D space. Nevertheless, it does not guarantee to be the exact 3D topological skeleton of the map.

F. Spectral Graph Theory

Considering that GNG is an algorithm intended for clustering, we define the stopping criteria to ensure the characteristics we are looking for in the graph. Spectral graph theory studies the properties of graphs through the eigenvalues and eigenvectors of their associated matrices.

Definition 1: The Laplacian (L) of an undirected graph, with n vertices, is a symmetric and positive semi-definite matrix.

Therefore, its eigenvalues are real and non-negative.

$$\lambda_1(G) \leq \lambda_2(G) \leq \ldots \leq \lambda_n(G)$$

Every row and column of the Laplacian matrix adds up to zero ($L\vec{1} = \vec{0}$). In consequence, the first and smallest eigenvalue of L is zero ($\lambda_1 = 0$).

Theorem 1: An undirected graph with n vertices has eigenvalues in the range $0 \leqslant \lambda \leqslant n$. The multiplicity of the zero eigenvalue equals the number of subgraphs in the graph, and the multiplicity of the eigenvalue n ($\lambda_n = n$) is equal to one less than the number of subgraphs in the complementary graph.

Corollary 1.1: An undirected graph is connected if the null space is 1-dimensional and spanned by the vector $\vec{1}$. Therefore, $\lambda_2 > 0$ if the graph is connected.

This eigenvalue is known as **Fiedler Value**, and it gives a certain level of the graph's connectivity [34], [35]. The dimension of the null space of the Laplacian, the algebraic multiplicity of the eigenvalue zero, is the number of connected subgraphs of the graph. Therefore, Fiedler's value is a suitable stop criterion for the Algorithm 1, which not only limits the time needed to generate the graph but also guarantees the connectivity of a unique connected graph.

Theorem 2: Given $A \in \mathbb{C}^{nxn}$ the eigenvalues are in the union of the Gershgorin circles with center $a_{i,i}$ and radius $r_i = \sum_{j=1, j\neq i}^{n} |a_{ij}|$.

Considering a graph G is connected if its complementary graph $\overline{G}$ is disconnected. By applying Gershgorin circles to the Laplacian matrix, the real eigenvalues are inside the union of the circles of radius r and positioned in center r, where r are the values in the diagonal of the Laplacian $d(G)$. $d_{max}(G)$ represents the greatest value in the diagonal.

$$\lambda_1(G) \leq \lambda_2(G) \leq \ldots \leq \lambda_n(G) \leq 2d_{max}(G)$$

Such inequality provides another superior bound for the largest eigenvalue. Therefore, to avoid the continuous calculation of the Laplacian's eigenvalues, while the number of vertices is less than twice the maximum value in the Laplacian's diagonal $n < 2d_{max}(G)$ it is possible for $\lambda_n = n$. In consequence, $\overline{G}$ will have at least 2 subgraphs. Once the condition $n > 2d_{max}(G)$ is reached, it is not possible to guarantee anymore that the complementary graph is disconnected. Hence, eigenvalues must be calculated.

G. Graph Fitting

GNG does not consider any vehicle size constraints to create the graph. Therefore, some of their vertices and edges may not hold the specifics needed for collision-free planning. In collision-free planning, a path is free if every robot's configuration within the path is in $\mathcal{C}_{free}$. For this reason, a fitting phase is needed before planning.

The fitting phase considers two checks: vertex check and edge check, described within Algorithms 2 and 3 respectively. Considering the ESDF of the map, we test whether the node is at a distance greater than a suitably sized threshold that considers the dimension of the drone. The drone is modeled as a sphere of radius equal to threshold R_{th} to improve the computation for collision checks. If the position of the node is beyond the radius, the following node is tested. If it is not, its position is changed using the gradient of the ESDF and a magnitude relative to the distance missing to satisfy the constraint. This process is repeated until reaching the threshold constraint or exceeding a maximum number of trials.

Algorithm 2: Fix Vertices GNG Graph.

```
foreach u in V do
    if getDIST(getWeight(u)) ≥ R_th then
        continue
    else
        δ ← R_th − getDIST(getWeight(u))
        counter ← ∅
        while getDIST(getWeight(u)) < R_th &
         counter ≤ MAX_ITERATIONS do
            m ← getGRAD(getWeight(u)) + m
            new_pos ← newPOSITION(u, m, δ)
            if getDIST(new_pos) ≥ R_th then
                updateWEIGHTS(u, new_pos)
            else
                counter ← counter + 1
            end
        end
    end
end
```

The edge analysis is accomplished by down-sampling the edge connecting both vertices and testing whether the samples respect the distance constraint to the closest obstacle. The fixing process is analogous to vertex check if a collision is found, but in addition, the starting edge is deleted. Then, the down-sampled vertices are fixed, and new connections are generated. In Fig. 1(a), we depict a collision-free path within an edge and, in Fig. 1(b), the proposed solution in case of finding a collision.

H. Planning Algorithm

Our method uses a two-step search. First, we use a sample-based method as Rapidly-Exploring Random Tree (RRT) to find a path connecting the closest vertices of the graph to both the start and the goal position. The RRT algorithm is run in a reduced space containing the previously stated points, such that the time spent to find a possible solution is as short as possible.

The second step is on the graph path computation from the starting vertex to the goal vertex, whose load is minimal because graph vertices and edges are built in a k-D tree structure. This two-step search is motivated by the fact that there is only one graph, so there is going to be a solution to the global planning problem once it is reached.

IV. Experiments

We evaluated the functionality and the effectiveness of the proposed approach in simulated environments[1] of a collapsed building and collapsed station, and further on the datasets[2] provided by Oleynikova et al. [19] collected in real environments. For the simulated environment, the ESDF map are computed using the framework described in our previous work [5]. On the other hand, in the datasets the environment map was already provided and in particular, we focused on the

[1]Both environments are part of the Gazebo model database.

[2][online]: https://www.github.com/ethz-asl/mav_voxblox_planning

Algorithm 3: Fix Edges GNG Graph.

```
foreach u,v in E do
    if getVERTEX_DIST(u, v) ≤ R_th then
        continue
    else
        foreach n in edgeDOWNSAMPLE(u, v) do
            if getDIST(n) ≥ R_th then
                continue
            else
                addVERTEX(vertex_to_fix, n)
            end
        end
        if vertex_to_fix is not empty then
            removeEDGE(u, v)
            fixVERTICES(vertex_to_fix)
            addEDGES(u, v, vertex_to_fix, E)
            vertex_to_fix ← ∅
        end
    end
end
```

maps collected using a Realsense RGB-D camera in this work. All the environments are typical indoor/outdoor Search and Rescue and Industrial scenarios.

Table I summarizes the value of the parameters used within the experiments for both graph generation and path planning.

Several techniques have been proposed over the years to speed up the GNG. Mainly, we applied a parameter that defines a threshold for the minimum distance between the data sample and the closest neuron for which the weights of a neuron will not be updated. Moreover, we computed the eigenvalues of the Laplacian every time a new node is generated. The rest of the parameters follow the original GNG algorithm structure.

A. Comparison Criteria

The experiments are going to be analyzed in two dimensions. First, analyzing the features of the graph and then, comparing its performance in a planning problem.

1) Graph's Connectivity: the algebraic connectivity gives an overall picture of the connectivity of a graph and if it is connected.

2) Graph's Density: we define the density of a graph from the relationship between its actual number of edges and the maximum number of edges allowed. The graph's density is a metric value that can be used to define how sparse the graph is. The graph's density ranges from $\emptyset$ to 1 where being closer to zero means sparsity.

$$Max_{Edges}(V) = \binom{|V|}{2} = \frac{|V||V-1|}{2}$$

$$Density(V,E) = \frac{E}{Max_{Edges}(V)} = \frac{2E}{|V||V-1|}$$

3) Graph's Size: we define the graph's size by analyzing the diameter of the graph and the amount of memory required to store the information in a file. The diameter of the graph is the maximum distance between a pair of vertices.

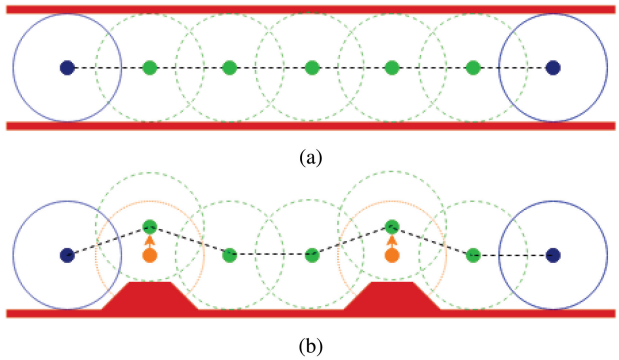


Fig. 1. Collision-free between two nodes connected by an edge. (a) Edge down-sampling for sphere test. Green nodes satisfy the radius constraint. (b) Collision detected, orange nodes do not respect the radius constraint. Point displacement using environment gradient.

TABLE I
PARAMETERS USED FOR THE SPARSE GRAPH GENERATION AND PATH PLANNING THROUGH ALL THE EXPERIMENTS

Parameter	Notation	Value
Drone's Sphere Radius	R	0.5 m
Radius Threshold	R_{th}	0.55 m
Down-sampling Distance		0.55 m
Starting Number of Nodes		2
Max. Age Between Nodes	A_{MAX}	50
Iter. Before Add Neuron	γ	50
L.R. Winner Neuron	ϵ_b	0.2
L.R. Neurons Con. Winner	ϵ_n	0.05
Stop Criteria - Fiedler's Value	λ_2	0.025
Min. Distance for Update		0.5 m

4) Graph Generation: time needed to generate the graph.

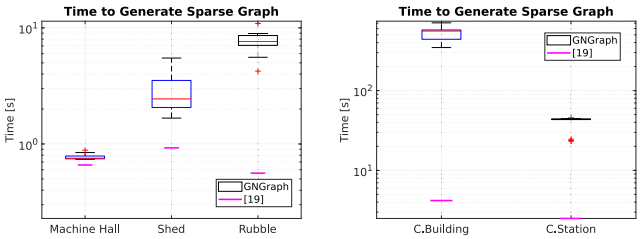
5) Planning: for planning, four key points are considered: *(i)* the time spent finding a solution; *(ii)* the success rate under a series of experiments; *(iii)* the minimal distance between the center of the sphere (drone) and the obstacle in case of collision; *(iv)* the path length of the proposed solution.

B. Graph Generation Comparison

Starting from the map of the previously introduced environments, we computed the graph with the proposed solution and with the method proposed by Oleynikova et al. [19] (skeleton). Since the GNGraph method is based on the GNG algorithm being an iterative non-deterministic algorithm, we generated different graphs. In more detail, we created fifteen randomly generated graphs for each environment, whose results are reported using their mean and standard deviation in Tables II and III.

Fig. 2(a) and (b) show the amount of time required to compute the sparse graph for each dataset and the simulated environment, respectively. Considering the Skeleton [19] method is deterministic, it is shown as a horizontal line.

The pipeline used by Oleynikova et al. [19] to generate the graph is different from the one in this work. Their algorithm first constructs a graph, which may contain several subgraphs, and then attempts to reconnect disconnected subgraphs. Their reconnection approach uses a two-step search: first, along with the skeleton diagram, and later, in the traversable space of the ESDF that attempts to hook subgraphs using close to straight-line representations. Therefore, while their procedure is



(a) Datasets (b) Simulated Environments

Fig. 2. Time Comparison between sparse graph generation.

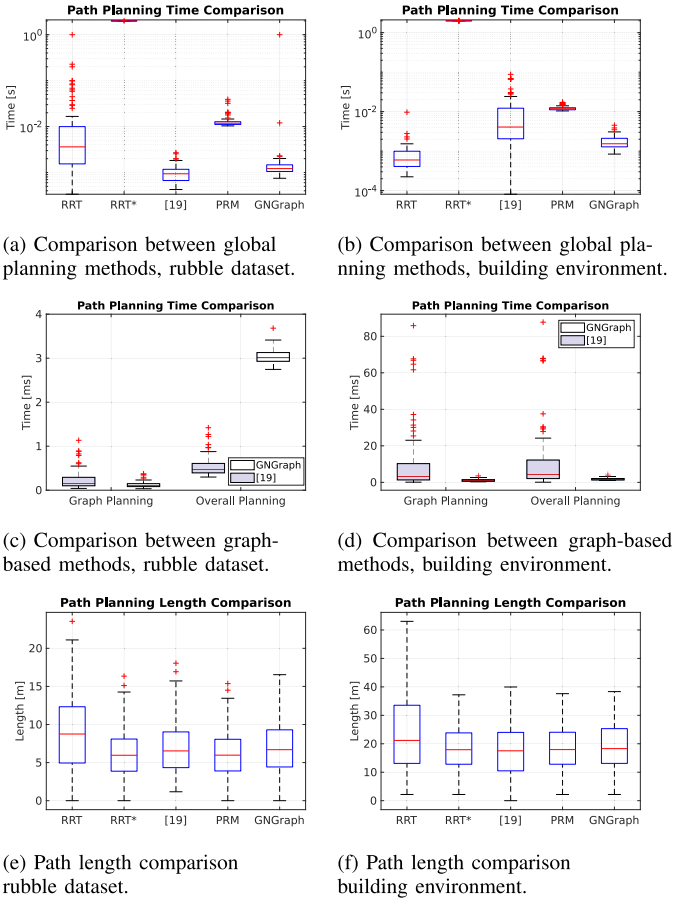
much faster than the proposed algorithm, it does not guarantee the graph's connectivity.

In Table II, we summarized the features of the resultant graphs. GNG can obtain a sparse graph with a reduced number of components (vertices and edges), preserving the topology of the environment. As shown in the table, the number of components, the graph diameter, and its file size are smaller in the GNGraph compared with the Oleynikova et al. [19] graph. Moreover, the stopping criterion proposed in this work, on the one hand, assures the graph's connectivity. On the other hand, it significantly compromises the time needed to generate the sparse graph, as shown in Fig. 2, which increases with the number of vertices and edges.

C. Global Planning Comparison

To evaluate the effectiveness of GNGraph for path computing in a given environment, we compared its performance with respect to Oleynikova et al. [19] and other planning methods, i.e., RRT, RRT*, and Probabilistic Roadmap (PRM). We randomly defined 100 planning tasks selecting the starting and the goal position in $\mathbb{R}^3$. We computed the path planning solution for all the methods and compared the planning time, the path length, the success rate, and the collisions among the path. Given the similarity in the obtained results, we show the outcome of a representative environment for both the dataset and the simulation environments.

Fig. 3 shows the computing time required for planning a path and the path length given by the global planners in the Rubble and the Collapsed Building environment, respectively. It can be seen in Fig. 3(a) and (b) that the planning time of Oleynikova et al. [19] and GNGraph algorithms share the same order of magnitude that is smaller than the others. More in detail, Fig. 3(c) and (d), highlight the computing time required by Oleynikova et al. [19] and GNGraph to plan the path on the graph and the overall computational time which incorporates the required time to find a path connecting the starting and goal points to the graph. As expected GNGraph requires less time to compute a graph planning because it has less number of components in the graph. However, during the experimental tests, we identified opposite behaviors regarding the computing time of these two planning methods. In the simulated environments, GNGraph resulted faster than Skeleton, whereas the contrary happened in the dataset environments. The results are consequence of the environment structure as well as of the chosen strategy to connect starting and goal points to the sparse graph (A* search for Skeleton and RRT for GNGraph). Dataset environments present an enclosed structure where the free space has limits on both sides. In this case, the A* algorithm manages to reach



(a) Comparison between global planning methods, rubble dataset.

(b) Comparison between global planning methods, building environment.

(c) Comparison between graph-based methods, rubble dataset.

(d) Comparison between graph-based methods, building environment.

(e) Path length comparison rubble dataset.

(f) Path length comparison building environment.

Fig. 3. Global Planning Benchmark in the Rubble and Collapsed Building environments. In these figures, a single graph of the proposed method is randomly chosen to represent the behavior.

for a vertex faster than RRT. Hence, the overall planning time of Skeleton is faster. On the contrary, the simulated environments combine large open space volumes with enclosed zones, where RRT finds a solution faster, and accordingly, GNGraph is faster.

By looking at the whole picture of the planning experiments, it is possible to notice that RRT as a global planner has a planning time that is nearly comparable to the medial-axis graph-planning methods. RRT overcomes their timing results in the simulated environments due to the combination of open and closed space zones embedded in the environment's geometry. Nevertheless, RRT does not guarantee the path length to be minimal, as shown in Fig. 3(e) and (f). On the other hand, RRT* manages near-optimal path length solutions compromising the time, while PRM obtains a good trade-off between planning time and path length without overcoming the results from Skeleton and GNGraph.

D. Results Discussion

Firstly, we analyze the resultant graphs generated by both medial-axis graph-based methods. Noticing the density of the graphs generated by both methods, they can be defined as sparse. The time required by Oleynikova et al. [19] approach to generate the resultant graph is less than the proposed method, but it does not guarantee connectivity. Therefore, there are going to be areas

TABLE II
SPARSE GRAPH COMPARISON ON EACH ENVIRONMENT BETWEEN OLEYNIKOVA ET AL. [19] AND GNGRAPH

Environment	Algorithm	Subgraphs	Vertices	Edges	Density	Diameter	File Size
Machine Hall	[19]	1	550	1096	$7.26e^{-3}$	43	111.7 kB
	GNGraph	1	68.67 ± 9.66	82.67 ± 10.67	$(3.63 \pm 0.54)e^{-2}$	30.2 ± 3.10	$8.55\pm$ 1.13 kB
Rubble	[19]	1	443	801	$8.18e^{-3}$	30	82.8 kB
	GNGraph	1	242.8 ± 17.32	415.47 ± 30.34	$(1.49 \pm 0.34)e^{-2}$	27.5 ± 3.31	$40.89\pm$ 2.98 kB
Shed	[19]	3	821	1544	$4.59e^{-3}$	–	111.7 kB
	GNGraph	1	255.6 ± 30.74	345.13 ± 51.33	$(1.07 \pm 0.14)e^{-2}$	46.47 ± 6.99	$35.81\pm$ 5.07 kB
Collapsed Building	[19]	4	5332	9959	$0.70e^{-3}$	–	1.0 MB
	GNGraph	1	1036.0 ± 57.04	2611.53 ± 133.31	$(4.94 \pm 0.26)e^{-3}$	26.07 ± 1.1	$243.45\pm$ 15.22 kB
Collapsed Station	[19]	2	4022	7523	$9.3e^{-4}$	–	761.87 kB
	GNGraph	1	430.47 ± 43.58	1140.4 ± 330.42	$(1.16 \pm 0.14)e^{-2}$	23.2 ± 0.86	$99.5\pm$ 10.25 kB

TABLE III
GLOBAL PLANNING COMPARISON ON EACH ENVIRONMENT

Environment	Algorithm	Success Rate	Min. Obs. Distance
Machine Hall	RRT	1.00	-
	RRT*	1.00	-
	[19]	0.89	0.476 ± 0.020
	PRM	1.00	-
	GNGraph	0.87 ± 0.057	0.477 ± 0.010
Rubble	RRT	0.99	-
	RRT*	0.99	-
	[19]	0.86	0.477 ± 0.011
	PRM	0.99	-
	GNGraph	0.87 ± 0.02	0.421 ± 0.011
Shed	RRT	0.99	-
	RRT*	0.96	-
	[19]	0.66	0.461 ± 0.022
	PRM	0.99	-
	GNGraph	0.69 ± 0.077	0.457 ± 0.083
Collapsed Building	RRT	1.00	-
	RRT*	1.00	-
	[19]	0.79	0.323 ± 0.124
	PRM	1.00	-
	GNGraph	0.93 ± 0.018	0.447 ± 0.069
Collapsed Station	RRT	0.98	-
	RRT*	0.97	-
	[19]	0.89	0.467 ± 0.018
	PRM	0.98	-
	GNGraph	0.78 ± 0.047	0.399 ± 0.195

that are not reachable. On the other hand, once the condition $n > 2d_{max}(G)$ is reached in our algorithm, using the Fiedler's value as stopping criterion is a computationally expensive process as shown in Fig. 2. Moreover, even if the time needed to compute the graph by the proposed solution is higher, it is not a problem for the MAV planning capability since it is a one-time offline process. For the MAV companion computer it is more suitable to embed an optimized environment representation in terms of memory allocation and computing process at run-time. Table II shows the amount of memory required to store the resultant graphs which are smaller for the proposed method.

Secondly, we analyze the success rate in planning a path in the given environment. Table III shows the results to the path planning problem for the global planning methods. GNGraph maintains a high success rate which is comparable to the global planner proposed by Oleynikova et al. [19] while considerably reducing the size of the graph. In general, graph-based methods can fail in finding a solution for two reasons: either the starting or goal positions are not feasible with the graph (i.e., they cannot be connected to the graph) or the resulting path is not collision-free. In the latter case, the minimum obstacle distance column in Table III points out the distance from the centre of the drone to the colliding obstacle.

V. CONCLUSION

In this work, we presented GNGraph: a map-based general method to build a low-dimensional topological graph of 3D spaces, specifically designed for global path planning for MAVs. The proposed solution is based on the GNG algorithm for edge generation and vertex pruning. Furthermore, exploiting spectral graph theory we proposed a criterion used to ensure the graph's connectivity as well as to stop the learning phase of the unsupervised learning algorithm.

By applying the learning process to the medial axis candidates, the resulting graph inherits the benefits from the medial axis discretization as the robustness to point-cloud noise and resolution, while significantly reducing the dimension of the topological graph. Furthermore, we demonstrated that planning efficiency is comparable to state-of-the-art global planning algorithms through experiments on simulated and real-world environments. Results showed that the RRT algorithm is faster than graph-based methods in open space-like environments due to the capability to quickly sample in a void environment. On the other hand, graph-based solutions overcome RRT in crowded spaces due to the time spent by RRT in sampling collision-free nodes, slowing down the entire solution.

Future work would include optimizing the computation of the stopping criterion (Fiedler's value) to decrease the time required to generate the sparse graph in order to execute all the process at run-time and hence extending our approach to dynamic environments.

REFERENCES

[1] VH. R. Nguyen Jenssen and D. Roverso, "Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning," *Int. J. Elect. Power Energy Syst.*, vol. 99, pp. 107–120, 2018.

[2] M. Satler, M. Unetti, N. Giordani, C. Avizzano, and P. Tripicchio, "Towards an autonomous flying robot for inspections in open and constrained spaces," in *Proc. IEEE 11th Int. Multi-Conf. Syst. Signals Devices*, 2014, pp. 1–6.

[3] P. Tripicchio, M. Satler, M. Unetti, and C. Avizzano, "Confined spaces industrial inspection with micro aerial vehicles and laser range finder localization," *Int. J. Micro Air Veh.*, vol. 10, no. 2, pp. 207–224, 2018.

[4] J. Gu, T. Su, Q. Wang, X. Du, and M. Guizani, "Multiple moving targets surveillance based on a cooperative network for multi-UAV," *IEEE Commun. Mag.*, vol. 56, no. 4, pp. 82–89, Apr. 2018.

[5] E. P. Herrera-Alarcón, D. Bagheri-Ghavifekr, G. Baris, M. Mugnai, M. Satler, and C. Avizzano, "An efficient object-oriented exploration algorithm for unmanned aerial vehicles," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2021, pp. 330–337.

[6] G. Hardouin, J. Moras, F. Morbidi, J. Marzat, and E. Mouaddib, "Next-best-view planning for surface reconstruction of large-scale 3D environments with multiple UAVs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 1567–1574.

[7] P. Tripicchio, M. Satler, G. Dabisias, E. Ruffaldi, and C. Avizzano, "Towards smart farming and sustainable agriculture with drones," in *Proc. Int. Conf. Intell. Environ.*, 2015, pp. 140–143.

[8] A. Murtiyoso and P. Grussenmeyer, "Documentation of heritage buildings using close-range uav images: Dense matching issues, comparison and case studies," *Photogrammetric Rec.*, vol. 32, no. 159, pp. 206–229, 2017.

[9] S. Werner, B. Krieg-Brückner, and T. Herrmann, "Modelling Navigational Knowledge by Route Graphs," in *Spatial Cognition II*. Berlin, Germany: Springer, 2000, pp. 295–316.

[10] M. Collins and M. Nathan, "Efficient planning for high-speed MAV flight in unknown environments using online sparse topological graphs," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 11450–11456.

[11] B. Fritzke, "A growing neural gas network learns topologies," in *Proc. 7th Int. Conf. Neural Inf. Process. Syst.*, Cambridge, MA, USA: MIT Press, 1994, pp. 625–632.

[12] D. Filliat and J-A. Meyer, "Map-based navigation in mobile robots:: I a review of localization strategies," *Cogn. Syst. Res.*, vol. 4, no. 4, pp. 243–282, 2003.

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

[14] N. Kalra, D. Ferguson, and A. Stentz, "Incremental reconstruction of generalized voronoi diagrams on grids," *Robot. Auton. Syst.*, vol. 57, no. 2, pp. 123–128, 2009.

[15] B. Lau, C. Sprunk, and W. Burgard, "Efficient grid-based spatial representations for robot navigation in dynamic environments," *Robot. Auton. Syst.*, vol. 61, no. 10, pp. 1116–1130, 2013.

[16] D. Kim, G. C. Kim, Y. Jang, and H. Jin Kim, "Topology-guided path planning for reliable visual navigation of MAVs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 3117–3124.

[17] Z. Fang, C. Luan, and Z. Sun, "A 2D voronoi-based random tree for path planning in complicated 3D environments," in *Proc. Int. Conf. Intell. Autonom. Syst.*, 2016, pp. 433–445.

[18] H. Oleynikova, Z. Taylor, R. Siegwart, and J. Nieto, "Sparse 3D topological graphs for micro-aerial vehicle planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–9.

[19] H. Oleynikova et al., "An open-source system for vision-based micro-aerial vehicle mapping, planning, and flight in cluttered environments," *J. Field Robot.*, vol. 37, no. 4, pp. 642–666, 2020.

[20] M. Foskey, M. C. Lin, and D. Manocha, "Efficient computation of a simplified medial axis," *J. Comput. Inf. Sci. Eng.*, vol. 3, pp. 274–284, 2003.

[21] A. Rosinol et al., "Kimera: From slam to spatial perception with 3D dynamic scene graphs," *Int. J. Robot. Res.*, 40, no. 12–14, pp. 1510–1546, 2021.

[22] D. Shah, B. Eysenbach, G. Kahn, N. Rhinehart, and S. Levine, "Ving: Learning open-world navigation with visual goals," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 13215–13222.

[23] A. Arleo, F. Smeraldi, and W. Gerstner, "Cognitive navigation based on nonuniform gabor space sampling, unsupervised growing networks, and reinforcement learning," *IEEE Trans. Neural Netw.*, vol. 15, no. 3, pp. 639–652, May 2004.

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

[25] S. Ma, W. Guo, R. Song, and Y. Liu, "Unsupervised learning based coordinated multi-task allocation for unmanned surface vehicles," *Neurocomputing*, vol. 420, pp. 227–245, 2021.

[26] D. Calisi, A. Farinelli, L. Iocchi, and D. Nardi, "Autonomous exploration for search and rescue robots," *WIT Trans. Built Environ.*, vol. 94, pp. 305–314, 2007.

[27] C. Hahn, S. Feld, M. Zierl, and C. Linnhoff-Popien, "Dynamic path planning with stable growing neural gas," in *Proc. Int. Conf. Agents Artif. Intell.*, 2019, pp. 138–145.

[28] B. Dellinger, R. Jenkins, and J. Walton, "Automated waypoint generation with the growing neural gas algorithm," in *Proc. 13th Int. FLAIRS Conf.*, 2017, pp. 404–407.

[29] F. Tencé, L. Gaubert, J. Soler, P. De Loor, and C. Buche, "Stable growing neural gas: A topology learning algorithm based on player tracking in video games," *Appl. Soft Comput.*, vol. 13, , no. 10, pp. 4174–4184, 2013.

[30] T. Martinetz and K. Schulten, "Topology representing networks," *Neural Netw.*, vol. 7, no. 3, pp. 507–522, 1994.

[31] T. Martinetz, "Competitive hebbian learning rule forms perfectly topology preserving maps," in *Proc. Int. Conf. Artif. Neural Netw.*, 1993, pp. 427–434.

[32] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. New York, NY, USA: Wiley, Jun. 1949.

[33] T. Martinetz and K. Schulten, "A "neural-gas" network learns topologies," *Artif. Neural Netw.*, vol. 1, pp. 397–402, 1991.

[34] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Math. J.*, vol. 23, no. 2, pp. 298–305, 1973.

[35] W. N. Anderson Jr. and T. D. Morley, "Eigenvalues of the Laplacian of a graph," *Linear Multilinear Algebra*, vol. 18, no. 2, pp. 141–145, 1985.