1-18-2024

Sphere-Graph: A Compact 3D Topological Map for Robotic Navigation and Segmentation of Complex Environments

Meryl Spencer
Michigan Technological University, maspence@mtu.edu

Reid W. Sawtell
Michigan Technological University, rwsawtel@mtu.edu

Sarah Kitchen
Michigan Technological University, snkitche@mtu.edu

Sphere-Graph: A Compact 3D Topological Map for Robotic Navigation and Segmentation of Complex Environments

Meryl Spencer, *Member, IEEE*, Reid Sawtell, and Sarah Kitchen, *Member, IEEE*

***Abstract*—Topological maps are a common framework for enabling autonomous robotic navigation. To be effective for robotic exploration the maps must be able to be generated quickly and compact enough to store on lightweight hardware. Here we propose a novel 3D topological map called Sphere-Graph which has adaptive edge lengths, can be quickly generated, and can be used to semantically identify hallways and rooms to produce a compact representation of complex environments. We give examples of the Sphere-Graph representation of large 3D urban and cave environments.**

***Index Terms*—Autonomous vehicle navigation, mapping, object detection, segmentation and categorization.**

I. Introduction

REDUCING a complex 3D environment to a more compact description is necessary for many tasks including robotic exploration and navigation. Graph structures are a popular method of reducing the complexity of the environment to a manageable degree, but the problem of translating a 3D volumetric space to a graph is non-trivial, particularly since there are an infinite variety of ways to encode such volumes, each with varying degrees of usefulness. The objective of the Sphere-Graph algorithm is to compute one such mapping with several desirable properties. In particular, we would like the graph to stay relatively sparse by maximizing the volume represented by each node in the graph, yet relatively complete in the sense that it extends to approximately all volume. Additionally, we would like nodes to be placed near the maximally free space in the 3D volume to assist with navigating along obstacle-free paths.

Our method is inspired by the medial axis [1], which in the case of a 2D world naturally forms a continuous graph-like skeleton of the interior of a closed bounded region. However, in three dimensions, this method produces a set of intersecting surfaces rather than a graph. Sphere-Graph instead computes a graph approximation of the medial axis (in any dimension), parameterized by a minimum sphere size. When the minimum sphere size is the robot clearance the graph forms an obstacle-free representation of the environment.

Manuscript received 7 August 2023; accepted 7 January 2024. Date of publication 18 January 2024; date of current version 2 February 2024. This letter was recommended for publication by Associate Editor Y. Jia and Editor A. Banerjee upon evaluation of the reviewers' comments. This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Grant HR00 1118CO124. *(Corresponding author: Meryl Spencer.)*

The authors are with the Michigan Tech Research Institute, Michigan Tech University, Ann Arbor, MI 48104 USA (e-mail: maspence@mtu.edu; rwsawtel@mtu.edu; snkitche@mtu.edu).

Digital Object Identifier 10.1109/LRA.2024.3355735

The adaptive scale and medial-axis approximating properties of the Sphere-Graph representation of a closed space allows one to extract the underlying topology of complex 3D environments. This information is useful in tasking teams of robots, as well as understanding important bottlenecks and alternate routes through the space. Additionally, semantic information such as hallways, doorways, and rooms can be extracted solely from the Sphere-Graph through straightforward graph operations. This information is entirely geometry based and does not require any training. Additionally, it works in both man-made and natural environments.

A. Related Work

Though topological methods for environment segmentation and route planning have been acknowledged for decades [2], they are not yet fully integrated into active SLAM approaches, in which occupancy-grids are still the dominant approach [3] to environment representation. Occupancy grids and octrees can be turned into topological maps by making each free voxel a node and placing edges between adjacent free voxels [4], [5], [6], [7]. However, this does not compact the environment any more than the occupancy map. There are several similar approaches to Sphere-Graph but they are all not medial axis approximating or not appropriate for online generation, as we will discuss.

Random graphs generated in the local free space of an environment have been used for creating local topological graphs of complex subterranean environments [8], [9]. In practice, such graphs provide a 3D interior mesh of the space at a high resolution, which while convenient for local navigation are not appropriate for large-scale topological semantic navigation. In contrast, Sphere-Graph intrinsically compresses open volume into the datum of a sphere center for a sphere appropriately scaled to that volume, and furthermore, features of the resulting global graph of the environment can be used to semantically separate e.g. hallways from rooms based purely on topology as described in Section V.

A method of producing a 3D Voronoi topological graph of an environment is provided in [10]. However, it is not guaranteed to be connected and does not incorporate non-zero robot size into collision-free paths. GNGraph [11] proposes solutions to these problems, but the method is stochastic and requires the full boundary of the environment to be known.

In [12], the authors propose a 3D topological graph of the environment based on connecting stochastically placed free-space polyhedrons. This approach is designed for tight integration into a high-level exploration planner and the resulting topological graph is unstable as new nodes are merged with existing explored

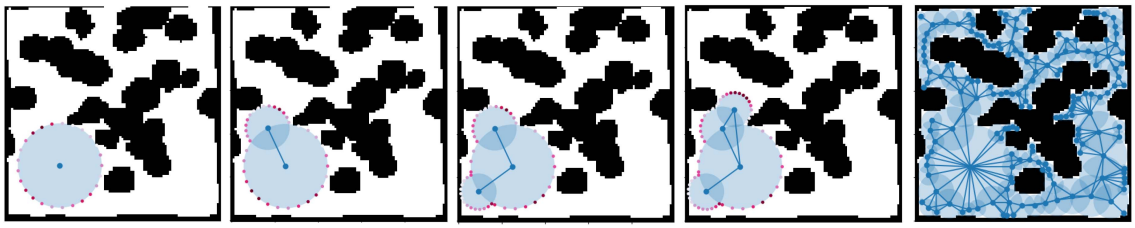

Fig. 1. Example of the Sphere-Graph algorithm in 2D. The sphere centers are blue and the points $n \in N$ are pink. The shade of the points in N corresponds to their distance to the nearest point on the boundary where darker shades indicate greater distance from the boundary. Nodes are added successively at the farthest point from the closed space in panels 1–4. The resulting graph is shown in panel 5.

spaces. It is unclear what properties the topological graph would have if run on a fully known environment for the purposes of extracting semantic properties.

SphereMap uses a similar approach to Sphere-Graph in which each node represents a sphere with a radius equal to the distance to the nearest obstacle [13]. New sphere positions in SphereMap are generated stochastically as opposed to deterministically (as in Sphere-Graph) so the method does not necessarily approximate the medial axis of the environment, and is not guaranteed to have the degree properties necessary for the type of semantic segmentation in Section V.

B. Contributions

The main contribution of this work is the algorithm for creating a topological graph of a complex three-dimensional environment with the following properties:

1) Adaptive edge lengths
2) Guaranteed minimum free space for robotic navigation in cluttered environments
3) Automatic semantic segmentation of "room-like" and "hallway-like" locations. "Hallway-like" locations only allow for forward or backward movement whereas "room-like" locations have multiple entrances and exits and require high-level choices to be made when path planning.
4) Simple extraction of underlying environment topology as described in Section V

These properties make the graph especially useful for multi-robot navigation by providing a high-level understanding of the intersection routes through the environment with a low-level obstacle-free path for local navigation. Additionally, we provide two explicit examples of the graph on large three-dimensional urban and natural environments.

II. Sphere-Graph Algorithm

A. Base Algorithm

The SphereGraph algorithm is built on a greedy space-filling algorithm and produces a graph by connecting the centers of the spheres used to fill the interior space of a closed bounded environment. The space filling algorithm begins at a seed location within the interior volume of closed space X defined by a boundary ∂X that may not be connected (e.g. interior walls may not connect directly to exterior walls in a building, but jointly are required to define the boundary of the unobstructed interior space). The algorithm assumes that the whole environment has been sensed. The extension to an unexplored boundary is straightforward, but out of scope for this paper. The choice of starting location makes little practice difference for the final graph as long as the starting point is in the open space and at least the minimum sphere size away from the boundary.

The radius of the sphere is set to the distance to the closest point on the surface of the surrounding environment. As a consequence, the resulting sphere is the largest that can be formed including only unobstructed space surrounding the seed location. At each consecutive step of the algorithm, a new center location is chosen among points on the boundary of the union of the previously generated spheres, and a new sphere of maximum unobstructed size is generated there. The new center location chosen is furthest from the boundary of the interior volume, thus we are greedily choosing the next point to maximize the volume of unobstructed space enclosed therein, while also not lying in the interior of previously generated spheres.

Another important consequence of selecting the new point on the surface of an existing sphere in each iteration is that we are guaranteed that the spheres overlap and that the straight line distance between the sphere centers is unobstructed. This process continues until there is no such point that can be chosen without the sphere radius being smaller than a pre-specified minimum size r_{min}. The resulting union of spheres represents a volumetric approximation of the unobstructed space of the environment, in which each sphere is guaranteed to significantly overlap with at least one other sphere.

The Sphere-Graph Algorithm (Algorithm 1) produces a graph where each node has an associated radius r within which there are guaranteed to be no obstacles. If after i iterations of the algorithm, we have a set

$$S^{(i)} = \{(\vec{v}_1, r_1), \ldots, (\vec{v}_i, r_i)\}$$

of pairs consisting of sphere centers $\vec{v}_j$ and radii r_j, denote the boundary of the union of the associated spheres by $\partial S^{(i)}$. Formally, let $B_{r_j}(\vec{v}_j)$ denote the closed ball of radius r_j centered at $\vec{v}_j$, then $\partial S^{(i)} := \partial \bigcup_{j=1}^{i} B_{r_j}(\vec{v}_j)$. The next sphere center $\vec{v}_{i+1}$ selected in the algorithm is the point on $\partial S^{(i)}$ furthest from any point in ∂X, and $r_{i+1} = \min_{\vec{p} \in \partial X} d(\vec{v}_{i+1}, \vec{p})$, where d is a distance metric. That is, we select the point on the boundary furthest from all obstructions, and the radius to be the minimum distance from that point to any obstruction. Then

$$S^{(i+1)} := S^{(i)} \cup \{(\vec{v}_{i+1}, r_{i+1})\},$$

and any points in the intersection of $\partial S^{(i)}$ and the sphere defined by $(\vec{v}_{i+1}, r_{i+1})$ are marked invalid (i.e. removed from the list of candidate spheres for the next iteration of the algorithm). The algorithm terminates when no new spheres can be generated with radius $> r_{\text{min}}$. In this way, we guarantee all spheres generated are contained in the interior of the volume. The two helper functions

Algorithm 1: Function for generating a Sphere-Graph.

procedure Sphere-Graph$\vec{v}_0, r_{\min}, \partial X$
  $\mathbf{S} = []$ ▷ Initialise an empty list of sphere nodes
  $\mathbf{E} = []$ ▷ Initialise an empty list of edges
  $\vec{p} = \min(|\vec{v}_0 - \vec{p'}|)$ **for** $\vec{p'} \in \partial X$ ▷$\vec{p}$ is the point in the boundary ∂X closest to v_0
  $d = |\vec{v}_0 - \vec{p}|$ ▷d is the distance from v_0 to the closest point on the boundary ∂X
  $\vec{v} = \vec{v}_0$
  while $d > r_{\min}$ **do** ▷ While the new sphere radius is greater than the minimum radius
  ▷ Make a sphere centered at $\vec{v}$ with radius d. Calculate the distance to the boundary for a set of discretized points on the surface of the sphere N_s. See Algorithm 2
    $\mathbf{s} = \{\vec{v_s}, r_s, d_s, \vec{n}_s, N_s\} = \texttt{CONSTRUCTSPHERE}(\vec{v}, d, \partial X)$
    for $\mathbf{s}' \in \mathbf{S}$ **do** ▷ For every existing sphere check for overlap with $\mathbf{s}$
      $o_1, d_{s'}, \vec{n}_{s'} = \texttt{MARKINVALIDPTS}(\mathbf{s}', \vec{v}, d)$ ▷ Mark the points on $\mathbf{s}'$ that intersect $\mathbf{s}$
      $o_2, d_s, \vec{n}_s = \texttt{MARKINVALIDPTS}(\mathbf{s}, \vec{v}_{s'}, r_{s'})$ ▷ Mark the points on $\mathbf{s}$ that intersect $\mathbf{s}'$
      if o_1**or**o_2 **then** ▷ If the spheres intersect
        $\{\mathbf{s}, \mathbf{s}'\} \in \mathbf{E}$ ▷ Add the pair of spheres to the edge list
      end if
      if $d_{s'} > d$ **then** ▷ Update the maximum distance to the closest point on the boundary
        $d = d_{s'}$, $\vec{v} = \vec{n}_{s'}$
      end if
    end for
    $\mathbf{s} \in \mathbf{S}$ ▷ Add the new sphere to the list of nodes
    if $\mathbf{s}_d > d$ **then** ▷ Update the maximum distance to the closest point on the boundary
      $d = \mathbf{s}_d$, $\vec{v} = \mathbf{s}_{\vec{n}}$
    end if
  end while
  return $\mathbf{S}$, $\mathbf{E}$
end procedure

in the algorithm are presented as Algorithm 2 and Algorithm 4 (see Appendix).

Fig. 1 gives an example of the first four iterations of the algorithm in a 2D environment. A starting point $\vec{v_0}$ is chosen in the lower left of the space and a circle centered at that point is generated that just touches the boundary of the closed space. At a set of discretized points on the boundary of the circle, the distance to the closest point in the closed space is determined. These points constitute N_s in Algorithm 1. The fundamental algorithm does not require discretization of the boundary. However, in practice, it is necessary for computation in spaces where the boundary can not be expressed in closed form. In Fig. 1 the relative distance to ∂X is denoted by the darkness of the pink point on the boundary of the circle. The point in this set with the farthest distance to the closed space is in the upper left of the sphere. It is used as the center of the next sphere as shown in the second panel. Once again, the distance to the boundary of the closed space is determined for the marked points on the surface of the sphere. All points in the intersection of the two spheres are marked as invalid and removed from the set of candidates for the next sphere center. The remaining valid points are shown in pink with darker colors indicating farther distances to the closed space boundary. In this union of points, the one with the farthest distance to the boundary is located on the lower right of the initial sphere. This point is the center of the third. The algorithm continues until the space is filled as shown in the last panel.

The result of the algorithm is a graph consisting of a set of nodes $\mathbf{S}$ and edges $\mathbf{E}$. Each node in the graph has a set of associated information derived from the sphere that generated the node. This datum is formally defined as $\mathbf{s} = \{\vec{v_s}, r_s, N_s\}$, where $(\vec{v}_s, r_s)$ are the sphere center and radius, as described above, and N_s is the set of points on the boundary of the sphere and their distances to the closest point in the closed space ∂X. Depending on the desired use of the graph, some or all of the data may be dropped. For most cases keeping only $(\vec{v}_s, r_s)$ is sufficient. The graph is deterministic up to choosing the starting location v_0, which must be in the free space of the environment and be more than $r_{\min}$ from the nearest obstacle. Since the graph grows rapidly to the midline of the environment the choice of v_0 has only minor effects on the placement of nodes that does not meaningfully change the reduced graph or the topological reduction in Section V.

B. Edge Reduction

In the base Sphere-Graph Algorithm, an edge is added between any two spheres whose volume intersects. Adding edges in this way is computationally efficient because it can be done during the check for newly invalidated candidate sphere centers, which requires first checking for sphere overlap. However, this produces a very large number of edges as well as producing edges that pass very near nodes as shown in Fig. 2. While this is fine in theory, in practice for robot navigation, it manifests in routing challenges, as a robot will unexpectedly "arrive" at a node it is proximal to while traversing an edge, which tends to produce errors in higher-level path planning algorithms that can cause robots to get stuck in loops between nodes. Our solution is to remove all such edges for which the second shortest path between the node endpoints of an edge e is less than $\tau|e|$, where $|e|$ is the length of the edge. The parameter τ is a scalar greater

Algorithm 2: Given a sphere center $\vec{v_s}$ and radius r_s this algorithm constructs a sphere $\mathbf{s} = \{\vec{v_s}, r_s, d_s, \vec{n}_s, N_s\}$. The point on the sphere closest to the boundary is $\vec{n}_s$, with distance to the boundary d_s. N_s is a set of tuples of all discretized points on the sphere and their distance to the closest point on the boundary.

procedure construct sphere$\vec{v_s}, r_s, \partial X$ ▷N' a list of discretized points on the surface of the sphere
$N' = \text{discrete}(\vec{v_s}, r_s)$
$N_s = []$ ▷ Initialize an empty list N_s
$d_s = 0$ ▷ Initialize the maximum distance to ∂X ▷ For each discretized point on the sphere
for $\vec{n} \in N'$ **do** ▷ Get the closest point on ∂X and its distance
$\vec{p} = \min(|\vec{n} - \vec{p'}|)$ **for** $\vec{p'} \in \partial X$
$d_n = |\vec{n} - \vec{p}|$
$f = 1$ ▷f is a flag for use in Algorithm 4 (see Appendix)
$\{\vec{n}, d_n, f\} \in N_s$
▷ Append the tuple to N_s▷ Update the closest point to the boundary
if $d_n > d_s$ **then**
$d_s = d_n, \vec{n}_s = \vec{n}$
end if
end for
return $\mathbf{s} = \{\vec{v_s}, r_s, d_s, \vec{n}_s, N_s\}$
end procedure

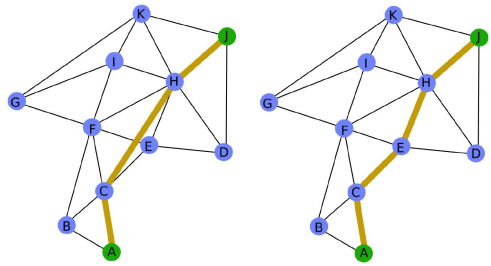

Fig. 2. Example of edge reduction. The edge C-H nearly passes through node E, which may cause path planning issues when traversing from A to J if the robot localization has noise. The robot is likely to become confused as if it is on edge C-H or at node E, which may cause it to become stuck in a higher-level reasoning loop. Removal of the edge clears any confusion. This problem is exacerbated in 3D where the error in the pose estimation in the z-axis can differ greatly from the x-y axis.

than one, where larger values lead to more edge pruning. In practice, we find $1.25 < \tau < 2.5$ works well.

C. Node Reduction

Although the base Sphere-Graph Algorithm can be used for navigation and mapping, there are several graph reduction techniques that make navigating and sharing the graph more robust to errors in pose estimation and limited communications bandwidth.

A positive feature of the Sphere-Graph Algorithm is that it is multi-scale and not tied to a grid. This makes it robust when used to generate navigation graphs for exploration in cave-like environments where very small winding passages coexist with large open spaces. In order to guarantee an obstacle-free navigational graph the minimum sphere size r_{min} should be set to the minimum area required for safe movement of the robot. It is therefore the property of the robot's size and maneuverability and not the environment.

In practice when the robotic platform is small and highly maneuverable (such as a small drone) the small value of r_{min} leads to many small spheres being generated in clusters filling corners and nooks in otherwise large open spaces. These wall-hugging spheres are often uninteresting from the perspective of mapping and exploration and greatly increase the size of the graph. In practice, it is generally good to set a larger cutoff r_{med} which is the minimum "interesting" volume for exploration. This cutoff is application and environment-dependent. For example in a large office building, $r_{\text{med}} = 0.5$m would capture the openings of doors and the space inside cubicles, but not the open area inside a trashcan. In a cave environment, r_{med} should be the approximate radius of the smallest passageway of interest.

After the base graph is created we remove any spheres that are smaller than r_{med} whose removal does not result in a disconnected graph. This greatly reduces the number of nodes in the graph while preserving the connectedness and navigability of tight passageways. It also makes the graph robust to the existence of passageways only slightly smaller than $r$med that connect two larger spaces since nodes are NOT removed if doing so would cause the graph to be disconnected.

For the rest of the paper we refer to a Sphere-Graph that has had edge and node reduction applied as a reduced graph, which is parameterized by the edge threshold τ and the medium radius cutoff r_{med}. We will show the data compression from using the reduced graph on two example environments in the next section. Additionally selecting an r_{med} approximately the size of hallways in urban environments produces a graph topology that is more conducive to semantic labeling as discussed in Section V.

TABLE I
COMPARISON OF SIZE OF DIFFERENT TOPOLOGICAL GRAPHS

	Urban Nodes	Urban Edges	Cave Nodes	Cave Edges
Point Cloud	9,462k	NA	2,948k	NA
1m Lattice	138k	375k	304k	828k
2m Lattice	16k	39k	35k	86k
Sphere-Graph	7k	27k	6k	18k
Reduced Graph	2k	6k	3k	8k

III. EXAMPLES: URBAN AND CAVE ENVIRONMENTS

In this section, we give two examples of the reduced Sphere-Graph of two large three-dimensional enclosed environments. The first environment is a large urban building with dozens of rooms spanning a 200-by-300 m area. The second environment is a cave system with twisting branching tunnels that cover about 0.03 km^3. These environments are based on the gazebo models created for the Subterranean Challenge [14], [15]. The leftmost panels of Fig. 3 show the extracted point clouds of the two environments. A Sphere-Graph representation of each environment was made using a minimum sphere radius of 1 m. The graph was reduced according to the parameters $\tau = 1.25$ and $r_{\text{med}} = 2$m. Table I gives the reduction in the size of the representation of the two spaces from the initial point cloud to the Sphere-Graph and then the reduced graph. For the urban

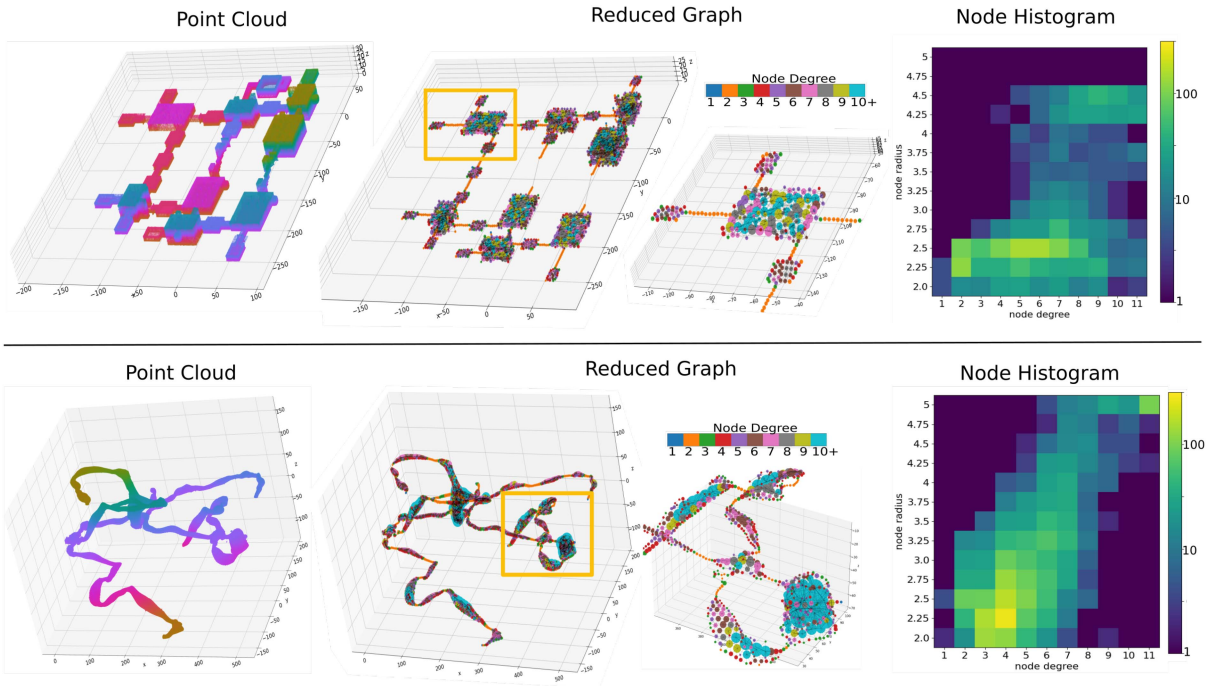

Fig. 3. Example of the reduced sphere-graph for two environments. The top panels show an urban environment and the bottom panels show a natural cave environment. From left to right are the point cloud of the environment, the reduced Sphere-Graph, and the 2D histogram of the radius and degree of the nodes in the reduced Sphere-Graph. The point clouds were based on the gazebo worlds [14], [15].

environment, the reduced graph is about 75 percent smaller than the original Sphere-Graph. In the cave environment, the reduced graph is about 50 percent smaller. Storing either graph is an order of magnitude better than storing the original point cloud. The middle two panels in Fig. 3 show the reduced graph for each environment and a zoomed-in view of the highlighted section of the graph. The size of the nodes is proportional to the sphere radius and the color of the nodes denotes the degree. The right two panels show a 2D histogram of the distribution of radius and degree for all nodes in the reduced graph on a log scale. In general, both graphs are dominated by nodes with smaller radii. The structure of the histograms reflects the underlying geometry of the two environments. The histogram for the urban environment has two peaks at 2.5 m and 4.5 m node radii. This reflects the 5 m height of the hallways and the 9 m height of the larger rooms. In contrast, the naturalistic environment shows an exponential decay in the size of the nodes which reflects the varying size of the tunnels. Because the Sphere-Graph is constructed to approximate the midline of the space the structure of the graph can be used to infer general characteristics of the environment without having to visualize the graph. Grid-based methods of building the graph will not have the same properties as each node is the same size and the vast majority of nodes will have the same degree. We will discuss this more in Section IV

IV. Comparison to Other 3D Topological Graphs

There are a number of different ways to create a topological graph from the boundary of an environment. In this section, we will compare Sphere-Graph to two of the most commonly used representations for robotic exploration and path planning: grid-based and medial axis based.

A. *Lattice-Based Topological Graphs*

We define a lattice-based topological graph as one in which a regular lattice (generally squares or triangles) is embedded in the entire plane of a 2D environment or the volume of a 3D environment. All nodes (and associated edges) in the closed space of the environment are removed from the graph. A positive of these graphs is that they are very easy to produce for both fully known environment boundaries and boundaries updated through robotic exploration. Additionally, if the lattice spacing is chosen to be greater than the operating size of the robot the graph can be used directly for navigation.

This representation is generally more useful in two-dimensional spaces where tight spaces and open areas are of similar scale. To ensure that the full enclosed space is included in the topological graph the spacing between the nodes must be smaller than the smallest expected hallway/tunnel width. If not one risks that the discretization effects of the graph will miss small passages and truly connected areas will become disconnected. In spaces with large-scale differences (such as caves or warehouses), this leads to very large graph sizes as the large open spaces are filled with tightly packed nodes from the fixed lattice size. This problem is exacerbated in three-dimensional spaces.

It is difficult to extract general properties of the environment and geometric semantic information from these topological graphs because the vast majority of nodes have the same degree. Only nodes directly adjacent to the boundary will have less than the maximum lattice degree. Additionally, since all nodes are equally spaced, they contain no information about the relative size of the space they are in unless that information is separately determined and recorded. It is not, in general, possible to do the type of semantic geometric segmentation in Section V using these graphs.

B. Relationship to the Medial Axis Topological Graph

Sphere-Graph is a medial axis approximating algorithm. Formally the medial axis of a closed surface is the collection of all points on the interior for which the set of closest points on the boundary contains more than one point. In two dimensions the medial axis of an enclosed space is a set of lines intersecting at junctions that are at the local maxima of the distance function. The medial axis can be easily converted to a graph by placing nodes at all of the intersections of the medial axis lines.

Finding the analytic medial axis of a continuous 2D area is generally infeasible. However, there are several good approximation algorithms for discretized images. These approximate "skeletonizations" are more commonly used for robotic navigation. In three dimensions the medial axis of a closed area is generally a complex surface of intersecting planes, for which there is no obvious conversion into a graph structure [16].

Medial-axis topological graphs have advantages over lattice topological graphs in that they are generally much smaller. Additionally, they match the intuitive topology of the space. Nodes with degree three are generally places where three branching tunnels/hallways meet. The major downside of these methods is that they do not provide an obstacle-free path for navigation, since the conversion to a graph does not place nodes along curving passageways, and they are too computationally expensive to perform in real-time on large spaces. Additionally, they are not well defined in three dimensions.

C. Scaling and Computational Complexities

The computational complexity of generating the Sphere-Graph is dominated by the repeated calls to the subroutines of Algorithms 2 and 4. The dominating factor in Algorithm 2 is the evaluation of the distance between points on the discretized sphere and the closest point on the boundary. The exact scaling is dependent on the computational representation of the boundary. If the boundary is defined by a point cloud stored as a KD-tree then the algorithm scales like the $\mathcal{O}(Nlog(p))$ where p is the number of points in the point cloud, and N is the number of points on the discretized sphere. Algorithm 4 scales like $\mathcal{O}(SN)$ where S is the number of spheres and N is again the number of discretized points on the sphere assuming a mean degree of 6. The entire algorithm scales like $\mathcal{O}(SNlog(p))$ Significant computational gains can be made by reducing the number of points evaluated on each sphere.

The Sphere-Graph representation of a 3D space is much more memory efficient than the full point cloud or a similar scale lattice graph representation. Table I gives the size of the cubic lattice graphs for the two demonstration environments at both a one and two-meter resolution. The one-meter resolution lattice graphs are more than two orders of magnitude larger than the reduced graphs. The two-meter resolution lattice graphs are ten times larger than the reduced graphs and are missing some of the tighter connecting hallways and tunnels.

V. Geometry Based Semantic and Topological Mapping

One goal of the Sphere-Graph representation of a space was to combine the free space navigability and fast generation of a lattice-based topological graph with the topological information contained in a medial axis graph. The following section describes how the unique properties of the Sphere-Graph can be used to create a high-level semantic understanding of the environment based only on the geometry. This is an application of Sphere-Graph separate from direct obstacle-free navigation.

Algorithm 3: Given a reduced Sphere-Graph $G = \{\mathbf{E}, \mathbf{S}\}$ each node $s \in \mathbf{S}$ is assigned a clique. See [17] for description of k-clique-communities algorithm.

procedure cluster nodes$G = \{\mathbf{E}, \mathbf{S}\}$ ▷ Successively contract all nodes of degree 2 from G
  while degree$(s) = 2$ for any $s \in \mathbf{S}$ **do**
    for $s \in \mathbf{S}$ **do**
      if degree$(s) = 2$ **then**
        $u, v =$ neighbors(s)
        $(u, v) \in \mathbf{E}$ ▷ Add edge between u and v
        remove(s)
      end if
    end for
  end while ▷ Identify all cliques via k-clique-communities [17]
  cids = k-clique-communities(G, 3)
  for $s \in \mathbf{S}$ **do**
    s.clique = cids(s)
  end for
end procedure

Because the Sphere-Graph tends towards placing large spheres along the medial axis of a space the reduced graph creates chains of degree two nodes down hallways and tunnels, subject to an appropriately chosen r_{med}. Fig. 3 shows two examples of the reduced graph with degree two nodes in orange. Higher degree nodes are in room or large cave-like locations. We can use this distinction to automatically detect 'room-like' and 'hallway-like' areas just from the topological graph without any camera imagery or neural network training. Unlike image-based semantic segmentation, we are using the inherent qualities of the geometry to identify rooms and hallways that can be applied to any space including natural environments.

Individual rooms can be identified by contracting all hallway nodes in the graph and clustering the remaining nodes based on their triangle connectivity following Algorithm 3. The left panels of Fig. 4 shows the result of this clustering applied to the two reduced graphs from Section III. Each room in the urban environment is correctly identified and more open 'room-like' portions of the cave system are also identified. This highlights the benefit of using a geometry-based segmentation of rooms as open spaces where hallways connect. The algorithm is highly extensible and does not require any foreknowledge of what a 'room' in the environment looks like, nor its specific size or shape, or level of clutter.

Once rooms are identified we can create a compact representation of the space by merging together all nodes with the same room ID. The result of this is shown in the center two panels of 4. This representation is especially human-interpretable since directions can be given in the form of "go down the hallway to the large room then take the right hallway and pass through two small rooms to arrive at your destination". This allows for human-machine teams to coordinate and move through large environments using the same underlying map. Additionally, it is an enabler for high-level path planning and coordination since

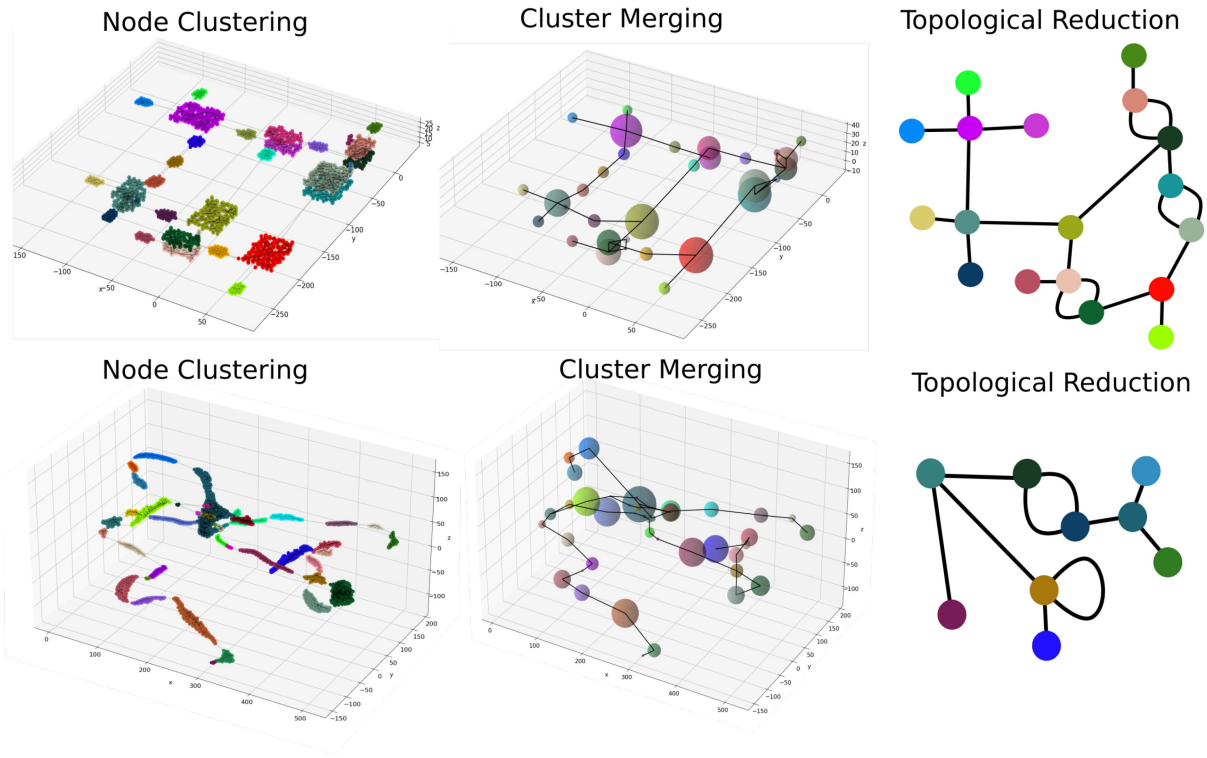

Fig. 4. Semantic and topological labeling for the reduced graphs from Fig. 3. The top panels are the urban environment and the bottom panels are the natural cave environment. Nodes in the left panel are colored according to their cluster label from Algorithm 3. The middle panel shows the result of merging all nodes from the same cluster together. The right panels show the topological reduction of the graphs. The node colors are consistent throughout.

rooms represent decision points where high-level plans about what hallway to traverse next need to be made.

We can further compact the graph by contracting room nodes that lie along a single hallway. These nodes are common in cave environments where a tunnel widens into a larger interior space and then narrows back into a tunnel. These spaces may have complex vertical and horizontal areas to explore, but they do not require large-scale choices other than to continue or turn back, unlike rooms that are at the center of two or more junctions. The right two panels show the result of contracting all of the degree 2 rooms from the merged clusters in the middle panel. This topological reduction shows only the terminating rooms and junctions of the space. Multiple edges between the same two nodes represent multiple exits that lead to the same destination. This topological reduction graph shows the fundamental branching structure of the environment. This representation can be important for identifying nodes and edges that are critical to maintaining the connectivity of the space. These locations are especially important in military and disaster scenarios where controlling or maintaining movement through the space is critical to larger mission planning.

VI. Conclusion

How large and complex 3D environments are represented has a huge effect on how easy it is to implement high-level AI tasks such as real-time exploration or coordinated multi-agent coverage. Additionally, constraints on onboard memory and limited communication bandwidth are common obstacles to the coordinated exploration of large spaces. In this paper, we have presented a new method for a compact representation of 3D spaces which we call Sphere-Graph. The topological graph is a non-stochastic size adaptive representation of the free space on the interior of an 2 or 3-dimensional closed environment. Each node in the graph is a 2 or 3-dimensional sphere with the largest radius that contains only free space. Therefore the graph can be used for direct low-level path planning for UAVs in 3D or for UGVs in 2D.

The adaptive radius of the nodes allows the graph to fully represent spaces with both large open areas and tight constrained spaces with an order of magnitude fewer nodes than a traditional fixed lattice topological graph. This is especially important when the graph must be stored on resource-constrained robots or shared across low bandwidth channels. Additionally, because each node is constructed to maximally fill the local space the nodes in the graph are largely on the medial axis of the space. This is especially useful in 3D environments where the exact medial axis is computationally prohibitive to compute.

We can take advantage of the medial-axis approximating qualities to significantly reduce the graph while maintaining the most important elements for high-level planning and human interoperability, namely the locations of ‘room-like’ and ‘hallway-like’ places in the environment. We can also further compact the graph to the fundamental topological shape of the environment so that each significant branching point is maintained. This representation is important for understanding vital paths and alternate routes which are important for disaster relief and military strategies.

Appendix

This appendix contains a subroutine for Algorithm 1

Algorithm 4: Given a sphere $\mathbf{s} = \{\vec{v_s}, r_s, d_s, \vec{n}_s, N_s\}$ and the center $\vec{v}$ and radius r of another sphere, this algorithm marks the discretized points in N_s as invalid if they are in the geometric intersection of the two spheres. It then returns a flag denoting if the spheres intersect and the updated closest valid point to the boundary.

procedure mark invalid ptss, $\vec{v}, r$
  if $|\mathbf{s}_{\vec{v}} - \vec{v}| < \mathbf{s}_r + r$ **then**
  ▷ If the spheres overlap
    for $\{\vec{n'}, d', f'\} \in N_s$ **do**
  ▷ for each point in N_s ▷ If the point is in the intersection of the spheres
      if f'**and**$|\vec{n'} - \vec{v}| < r$ **then**
        $f' = 0$ ▷ Mark the point as invalid
      end if
    end for ▷ Update the closest point to the boundary
    $d_s, \vec{n_s} = \max(d')$ **for** $\{\vec{n'}, d', f'\} \in N_s$ **if** $f' = 1$
    return $1, d_s, \vec{n_s}$
  else
    return $0, d_s, \vec{n_s}$
  end if
end procedure

ACKNOWLEDGMENT

The views, opinions, and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

REFERENCES

[1] R. Fabbri, L. F. Estrozi, and L. D. F. Costa, "On Voronoi diagrams and medial axes," *J. Math. Imag. Vis.*, vol. 17, pp. 27–40, 2002.

[2] S. Thrun and A. Bücken, "Integrating grid-based and topological maps for mobile robot navigation," in *Proc. Nat. Conf. Artif. Intell.*, 1996, pp. 944–951.

[3] J. A. Placed et al., "A survey on active simultaneous localization and mapping: State of the art and new frontiers," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1686–1705, Jun. 2023.

[4] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[5] A. Bucksch and R. Lindenbergh, "CAMPINO–A skeletonization method for point cloud processing," *ISPRS J. Photogrammetry Remote Sens.*, vol. 63, no. 1, pp. 115–127, 2008.

[6] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3D," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot.*, 2011, pp. 351–356.

[7] K. Kazakov, V. Semenov, and V. Zolotov, "Topological mapping complex 3D environments using occupancy octrees," in *Proc. 21st Int. Conf. Comput. Graph. Vis.*, Moscow, Russia, 2011, pp. 111–114.

[8] T. Dang, F. Mascarich, S. Khattak, C. Papachristos, and K. Alexis, "Graph-based path planning for autonomous robotic exploration in subterranean environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 3105–3112.

[9] J. Williams et al., "Online 3D frontier based UGV and UAV exploration using direct point cloud visibility," in *Proc. IEEE Int. Conf. Multisensor Fusion Integration Intell. Syst.*, 2020, pp. 263–270.

[10] 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.

[11] E. P. Herrera-Alarcón, M. Satler, M. Vannucci, and C. A. Avizzano, "GNGraph: Self-organizing maps for autonomous aerial vehicle planning," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 10721–10728, Oct. 2022.

[12] F. Yang, D.-H. Lee, J. Keller, and S. Scherer, "Graph-based topological exploration planning in large-scale 3D environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 12730–12736.

[13] T. Musil, M. Petrlík, and M. Saska, "SphereMap: Dynamic multi-layer graph structure for rapid safety-aware UAV planning," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 11007–11014, Oct. 2022.

[14] OpenRobotics. "(December) finals prize round world 03. open robotics," 2021. [Online]. Available: https://fuel.gazebosim.org/1.0/OpenRobotics/worlds/FinalsPrizeRoundWorld03

[15] OpenRobotics, "(December) Urban circuit practice 03. open robotics," 2021. [Online]. Available: https://fuel.gazebosim.org/1.0/OpenRobotics/worlds/UrbanCircuitPractice03

[16] D. Attali, J.-D. Boissonnat, and H. Edelsbrunner, "Stability and computation of medial axes: A state-of-the-art report," in *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, T. Moller, B. Hamann, R. D. Russell, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg 2009, pp. 109–125.

[17] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, pp. 814–818, Aug. 2005.