IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

HCOA*: Hierarchical Class-Ordered A* for Navigation in Semantic Environments

Evangelos Psomiadis, *Graduate Student Member, IEEE*, and Panagiotis Tsiotras, *Fellow, IEEE*

***Abstract*—This letter addresses the problem of robot navigation in mixed geometric/semantic 3D environments. Given a hierarchical representation of the environment, the objective is to navigate from a start position to a goal, while satisfying task-specific safety constraints and minimizing computational cost. We introduce Hierarchical Class-ordered A* (HCOA*), an algorithm that leverages the environment's hierarchy for efficient and safe path-planning in mixed geometric/semantic graphs. We use a total order over the semantic classes and prove theoretical performance guarantees for the algorithm. We propose three approaches for higher-layer node classification based on the semantics of the lowest layer: a Graph Neural Network method, a k-Nearest Neighbors method, and a Majority-Class method. We evaluate HCOA* in simulations on two 3D Scene Graphs, comparing it to the state-of-the-art and assessing the performance of each classification approach. Results show that HCOA* reduces the computational time of navigation by up to 50%, while maintaining near-optimal performance across a wide range of scenarios.**

***Index Terms*—Autonomous vehicle navigation, motion and path planning, AI-enabled robotics.**

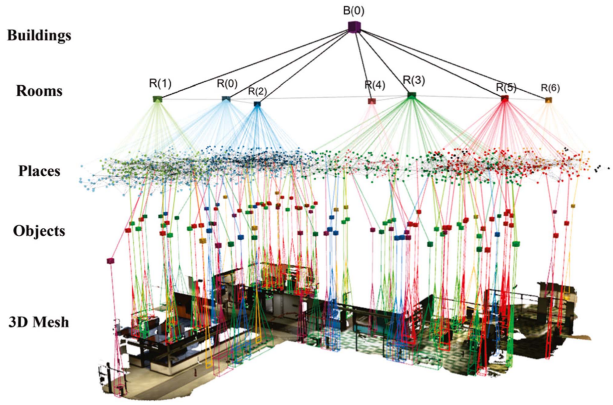


Fig. 1. 3D Scene Graph generated from the uHumans2 office scene dataset [14] using Hydra [2]. The graph comprises five layers, as shown in the figure. Room nodes are denoted as R(·), while building nodes are represented as B(·).

I. Introduction

As ROBOTIC sensing technologies advance, enabling robots to perceive vast and diverse information, two fundamental questions arise: *What information from this extensive data stream is most important for a given task?* and, second, *how can the robot effectively utilize this information for decision-making?* Hierarchical semantic environment representations, such as 3D Scene Graphs (3DSGs) [1], [2], [3], provide rich and structured abstractions that mirror human-like reasoning, thus facilitating the selection and organization of information, An example of a 3DSG is shown in Fig. 1.

Previous research in hierarchical path-planning has primarily addressed the first question [4], [5], [6], [7]. In [4] the authors introduce Hierarchical Path-Finding A*, a hierarchical A* [8] variant for grid-based maps. Their approach partitions the map into clusters with designated entrance points, which are used for high-level path-planning. The authors in [5] propose a hierarchical graph search algorithm for graphs with edge weights represented as intervals. In [7], the authors introduce S-Nav, a hierarchical path-planning algorithm for navigation in Situational Graphs, a multi-layer graph representation of indoor environments built on top of 3DSGs.

Semantic path-planning has focused on the second question above by incorporating semantics into the decision-making process. In [9], the authors propose SMaNa, a semantic-aware mapping and navigation framework that employs A* with edge weights computed using node distances and semantic labels. To avoid reliance on manually crafted, user-defined cost functions, several works adopted the concept of total order over semantic classes. In [10], a weighted function combining edge cost and semantic class is introduced to determine optimal paths while respecting a total semantic order. However, this approach requires global graph properties, making it computationally expensive. To address this limitation, the authors in [11] propose Class-ordered A* (COA*), an extension of traditional A* that incorporates semantic information, based on a total order. The algorithm is shown to be both complete and optimal under the defined total order and is validated in various planning scenarios. This work was later extended to dynamic environments in [12], where LPA* [13] was adapted for online replanning in dynamic maps with semantic information.

Building on the ideas introduced in [5] and [11], this letter proposes a hierarchical class-ordered path-planning algorithm that uses a total order on semantic classes. The notion of total order is applicable to a wide range of scenarios, including navigation in partially known environments [11] and safety-critical applications. In this work, we focus on encoding a set of safety constraints as relationships between semantic categories, which are implicitly enforced during path-planning.

Received 31 March 2025; accepted 19 July 2025. Date of publication 11 August 2025; date of current version 27 August 2025. This article was recommended for publication by Associate Editor X. Dong and Editor A. Bera upon evaluation of the reviewers' comments. This work was supported in part by ARL award DCIST CRA under Grant W911NF-17-2-0181 and in part by ONR award under Grant N00014-23-1-2304. *(Corresponding author: Evangelos Psomiadis.)*

The authors are with the D. Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0150 USA (e-mail: epsomiadis3@gatech.edu; tsiotras@gatech.edu).

Digital Object Identifier 10.1109/LRA.2025.3597868

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Navigation within hierarchical semantic environments has also been explored in prior work. In [15], the authors present IntelliMove, a hierarchical semantic map framework along with a semantic path-planner. The edge weights in the graph encode spatial relationships, such as distances between rooms or objects, and Dijkstra's algorithm is used to compute optimal paths. Furthermore, [16] addresses Task and Motion Planning in 3DSGs using a three-level hierarchical planner consisting of a discrete task planner, a navigation planner for path-planning, and a low-level planner. Their approach focuses on optimizing task planning to reduce computational cost, while path-planning operates on the unpruned 3DSG.

Learning-based methods have also been explored to facilitate task execution in a 3DSG. In [17], the authors propose HMS, a neural network-based approach for object search in a 3DSG, leveraging the 3DSG's hierarchy. In [18] and [3], the authors propose the Neural Tree, a Graph Neural Network (GNN) architecture designed for node classification. Although effective in classifying higher-layer nodes in 3DSGs, this approach requires a tree decomposition of the input graph, which increases computational time.

Main Contributions: Our approach integrates task semantics directly into the path-planning process within a unified algorithm. By leveraging the environment's hierarchy, we reduce computational resource demands while ensuring efficient and safe navigation. Our key contributions are as follows:

1) We introduce Hierarchical Class-ordered A*, a novel hierarchical semantic path-planning algorithm for navigation in hierarchical semantic environments.
2) We propose three methods for node classification on the higher layers of the environment's hierarchy: a Majority-Class (MC) method, a k-Nearest Neighbors (kNN) method, and a GNN method.
3) We validate our approach on two publicly available 3DSG datasets, demonstrating its effectiveness in real-world scenarios.[1]

Algorithm 1: Hierarchical Class-ordered A* (HCOA*).

Input: G, v_s, v_g, $h(\cdot)$
Output: π^0
for all $\ell = \ell_{n-1}, \dots, \ell_0$ **do**
  $v_s^\ell \leftarrow p(v_s, \ell)$; $v_g^\ell \leftarrow p(v_g, \ell)$
  $g(v_s^\ell) \leftarrow 0$; $\theta(v_s^\ell) \leftarrow 0 \cdot \mathbf{1}_K$; $f(v_s^\ell) \leftarrow h(v_s^\ell)$
  $g(v^\ell) \leftarrow \infty$; $\theta(v^\ell) \leftarrow \infty \cdot \mathbf{1}_K$, $\forall v^\ell \in G^{\ell'} \setminus \{v_s^\ell\}$
  PredecessorMap $\leftarrow \varnothing$ ▷Track path reconstruction
  $Q \leftarrow \{v_s^\ell\}$ ▷Initialize priority queue
  while Q is not empty **do**
    $v \leftarrow$ PopNode(Q); $Q \leftarrow Q \setminus \{v\}$
    if $v = v_g^\ell$ **then**
      $\pi^\ell \leftarrow$ Path(PredecessorMap, v_g)
      Break
    end if
    for all $u \in$ neighbors$(v, G^{\ell'})$ **do**
    $\theta(v,u) \leftarrow$ Semantics$(\phi_V^\ell(v), G^{0'}(u))$
    if $(\theta(v) + \theta(v,u) \prec \theta(u))$ **or**
    $(\theta(v) + \theta(v,u) = \theta(u)$ **and**
    $g(v) + w^\ell(v,u) < g(u))$ **then**
      $Q \leftarrow Q \cup \{u\}$
      PredecessorMap$[u] \leftarrow v$
      $\theta(u) \leftarrow \theta(v) + \theta(v,u)$
      $g(u) \leftarrow g(v) + w^\ell(v,u)$
      $f(u) \leftarrow g(u) + h(u)$
    end if
  end for
  end while
  for all $\hat{\ell} < \ell$ **do**
  $G^{\hat{\ell}} \leftarrow \{v \in G^{\hat{\ell}} : p(v, \ell) \in \pi^\ell\}$
  end for
  end for
  return π^0

II. Preliminaries

We model the environment as a hierarchical semantic graph (HSG). Specifically, let $G = (V, E, \mathcal{K})$ be a graph with n layers, where V is the set of nodes, $E \subseteq V \times V$ is the set of edges, and $\mathcal{K} = 1, \dots K$ represents a set of semantic classes ordered in decreasing priority. This priority is characterized by task-specific requirements (e.g., avoid certain areas or objects, or prefer known over unknown regions). We denote the set of layers as L, where $\ell = 0$ is the lowest abstraction (e.g., sensor) layer and $\ell = n$ is the highest abstraction layer (root). Each layer ℓ forms a connected weighted subgraph $G^\ell = (V^\ell, E^\ell, \mathcal{K}) \subset G$, with an associated weight function $w^\ell : E^\ell \to \mathbb{R}^+$, which corresponds to Euclidean distance in this work. We assume that each node in layer $\ell - 1$, for $\ell = 1, \dots n$, is connected to a single node in layer ℓ, which we refer to as its parent node. More generally, we define an ancestor as a node's parent or any higher-layer predecessor in the hierarchy. We define the projection function $p : V \times L \to V$ that maps each node to its corresponding ancestor in layer ℓ.

Each node in $\ell = 0$ is assigned a semantic class based on the given perception data and the specific task using the function $\phi_V^0 : V^0 \to \mathcal{K}$. We extend the labeling function ϕ_V^0 to layers $\ell \neq 0$. Details on the computation of ϕ_V^ℓ for $\ell \neq 0$ are given in Section V. Additionally, we define the edge classification function $\phi_E^\ell : E^\ell \to \mathcal{K}$ by $\phi_E^\ell(e) = \max(\phi_V^\ell(v), \phi_V^\ell(u))$ for $e = (v, u)$, thereby overestimating the edge class.

The environment model reflects the structure of 3DSGs, as shown in Fig. 1. For an overview of the advantages of this perceptual model and its construction, we refer the interested reader to [2]. In our framework, in addition to the spatial layers, we also utilize the object layer of the 3DSG to encode safety constraints and to infer the nodes' semantic classes, enabling more context-aware and task-relevant navigation.

Let $\Pi(v_s^\ell, v_g^\ell)$ denote the set of all acyclic paths in G^ℓ from $v_s^\ell \in V^\ell$ to $v_g^\ell \in V^\ell$, and let Π_k be the subset of all paths in Π for which the least favorable (i.e., highest) edge class is exactly k. Formally,

$$\Pi_k = \{\pi \in \Pi : N(\pi, k) > 0;\ N(\pi, k') = 0, \forall k' > k\}, \quad (1)$$

where $N(\pi, k) = |\{e \in \pi : \phi_E^\ell(e) = k\}|$ is the number of edges of class k in path π. We further define $\Pi_k^i \subseteq \Pi_k$ consisting of all paths in Π_k that contain exactly i edges of class k, that is $\Pi_k^i = \{\pi \in \Pi_k : N(\pi, k) = i\}$. To enable a consistent comparison of paths across different classes, we impose a total order such that $k < \ell \Rightarrow \Pi_k^i \prec \Pi_\ell^j$ for all i, j and $i < j \Rightarrow \Pi_k^i \prec \Pi_k^j$. This order ensures that any two paths with the same start and goal nodes can be compared.

[1]The code is available at https://github.com/epsomiadis3/HCOA-star.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

III. Problem Formulation

Consider a mobile robot tasked with navigating a complex 3D environment where certain regions have lower traversal priority. The robot is provided with a HSG of the environment (e.g., 3DSG), as outlined in Section II, where the semantic classes of the nodes in layer $\ell = 0$ are assigned based on perception, and tailored to the specific task (e.g., safety constraints). Let $v_s \in V^0$ and $v_g \in V^0$ denote the robot's starting and goal nodes, respectively. The objective is to determine the shortest path in $\ell = 0$ while minimizing traversal through the least favorable edges. Formally,

$$\pi^*(v_s, v_g) = \underset{\pi \in \Pi^*(v_s, v_g)}{\arg\min} \sum_{e \in \pi} w^0(e), \tag{2a}$$

$$\Pi^* = \min_{i \in \mathbb{N}} \min_{k \in \mathcal{K}} \Pi_k^i, \tag{2b}$$

where Π^* is the set of paths obtained by minimizing over all possible classes k and number of least favorable class i.

However, due to computational constraints, the robot seeks to avoid a full graph search over the entire G^0, as its structure is both large and semantically diverse, potentially rendering the search intractable.

A. Problem Statement

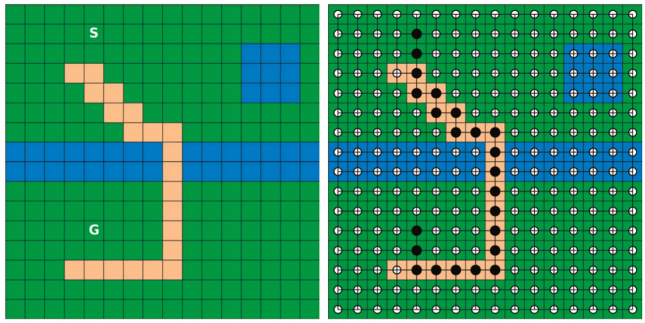
We propose a hierarchical semantic path-planning algorithm for efficient robot navigation in large-scale environments. The algorithm operates top-down across the layers of the HSG, iteratively computing the optimal semantic path at each layer while pruning nodes that are not included in the path. Additionally, we introduce three methods for node classification to predict the semantic classes of higher-layer nodes: a Majority-Class, a kNN, and a GNN method.

IV. Path-Planning

A. Hierarchical Class-Ordered A*

We introduce Hierarchical Class-ordered A* (HCOA*), a hierarchical semantic path-planning algorithm for HSGs. The algorithm initiates planning at layer $\ell = n - 1$, and recursively proceeds to lower layers. At each layer, it prunes the HSG based on the computed path from the previous layer and applies the same planning algorithm to the pruned HSG. Within each layer, the algorithm utilizes Class-ordered A* (COA*) [11], which finds the shortest path while minimizing the number of least favorable edges through lexicographic comparison. By leveraging the hierarchy, the algorithm performs multiple searches on smaller subgraphs rather than a potentially computationally expensive search over G^0.

HCOA* is presented in Algorithm 1. We use the function $h : V \to \mathbb{R}$ to denote an admissible heuristic function, similar to standard A*. Lines 2–6 initialize the variables, where the function $p(v, \ell)$ returns the ancestor of node v in layer ℓ. Lines 7–24 execute COA*, which will be detailed in Section IV-B. Notably, Line 14 computes the semantic class of the edge (v, u) based on the semantic classes of nodes v and u. If $\ell = 0$, the nodes' semantics are determined from perception data and the task, whereas for $\ell \neq 0$, they are predicted using the methods described in Section V. Additionally, $G^{0'}(u)$ is the induced subgraph of node $u \in V^\ell$ on layer 0 given by $G^{0'}(u) = \{u' \in V^0 : p(u', \ell) = u, \text{ where } u \in V^\ell\}$, while $G^{\ell'}$ is the pruned graph at layer ℓ (Line 27). Lines 25–27 refine the HSG by pruning nodes that do not share an ancestor with the paths in the higher layers.



(a) (b)

Fig. 2. (a) Grid world with three semantic classes: road, grass, river. **S** is the starting cell while **G** is the target; (b) Graph of the environment and path produced by COA*.

B. Class-Ordered A*

In this section, we present an overview of the simplified COA* implementation used in HCOA* to help the reader understand the simplifications made to the baseline algorithm. Further details, along with proofs of the COA* optimality and completeness, can be found in [11].

Consider a single-layer, weighted, semantic graph G^ℓ. We characterize each node v by the triple (q, g, θ_V), where q is the predecessor node (saved in the PredecessorMap in Algorithm 1), $g : V^\ell \to \mathbb{R}^+$ is the cost-to-come function, and $\theta_V : V^\ell \to \mathbb{N}^K$ specifies the number of edges of each semantic class along the path up to node v. We extend θ_V to single edges by defining $\theta_E : E^\ell \to \mathbb{N}^K$ as an one-hot vector with all elements zero except at ϕ_E. The subscript is omitted in Algorithm 1. The function PopNode selects the next node to expand, and Path constructs the optimal path by backtracking through the predecessor nodes from the goal.

Fig. 2 presents an example of COA* in a grid world, where the robot must navigate from **S** to **G** while satisfying safety constraints prioritized in descending order: (i) avoid water, and (ii) avoid grass. To enforce these constraints, we assign semantic labels to nodes via ϕ_V^ℓ.

C. Performance Guarantees

The following propositions establish the algorithm's completeness along with sufficient conditions for optimality. For both propositions, the assumptions for completeness and optimality of COA* hold [11]. We conclude the section with a discussion on the algorithm's computational complexity.

Proposition 1: Let G be an HSG and let nodes $v_s, v_g \in V^0$. HCOA* is *complete*, meaning that it is guaranteed to find a path $\pi^0 = \pi(v_s, v_g)$, whenever one exists.

Proof: Let π^0 exist. The structure of G ensures that for each layer $\ell \in L$, there exists a corresponding path $\pi^\ell = \pi(v_s^\ell, v_g^\ell)$, where $v_s^\ell = p(v_s, \ell)$ and $v_g^\ell = p(v_g, \ell)$. At each subgraph G^ℓ, COA* is guaranteed to find the optimal path $\pi^{\ell *}$, if one exists [11]. Suppose, ad absurdum, that the graph pruning performed in Lines 25–27 results in $\Pi(v_s^{\ell-1}, v_g^{\ell-1}) = \varnothing$, preventing

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

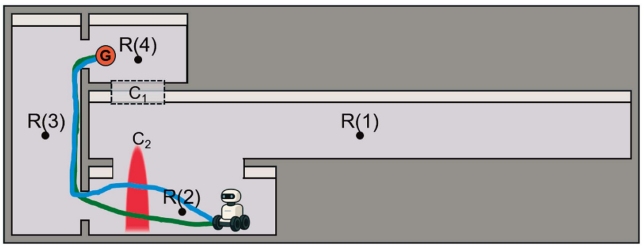


Fig. 3. Example demonstrating Proposition 2. $\mathbf{G}$ denotes the goal location, $R(\cdot)$ indicates rooms, and C_1, C_2 represent specific conditions: C_1: open door, and C_2: dangerous area.

COA* from finding a solution in layer $\ell - 1$. This implies that $\pi^{\ell *}$ is disconnected, contradicting COA*'s completeness. ■

Proposition 2: Let G be an HSG with nodes $v_s, v_g \in V^0$. Suppose that in layer $\ell = 1$, the acyclic path $\pi^1(v_s^1, v_g^1)$ is unique, and that for all $u_s, u_g \in V^0$ sharing the same parent, $P = p(u_s, 1) = p(u_g, 1)$, every node u along the optimal path $\pi^*(u_s, u_g)$ satisfies $p(u, 1) = P$. Then, HCOA* is guaranteed to find the optimal path $\pi^*(v_s, v_g)$.

Proof: Define $G^{0'} \subseteq G^0$ as the pruned graph after n iterations of HCOA*. Suppose, ad absurdum, that a segment of the optimal path $\pi^*(v_s, v_g)$ has been removed, i.e., there exists a sub-path $\widehat{\pi} \subseteq \pi^*(v_s, v_g)$ such that $\widehat{\pi} \not\subset G^{0'}$. Two cases arise: (i) The parents of $\widehat{\pi}$ in layer $\ell = 1$ form a cycle that starts and ends at a parent of a node in $\pi^*(v_s, v_g) \cap G^{0'}$. This contradicts the second assumption, which states that the nodes of the optimal path at $\ell = 0$, where the starting and goal node share the same parent, must all have the same parent. (ii) The parents of $\widehat{\pi}$ in layer $\ell = 1$ are part of a sub-path that starts at the parent of one node in $\pi^*(v_s, v_g) \cap G^{0'}$ and ends at a different parent of another node in $\pi^*(v_s, v_g) \cap G^{0'}$. This contradicts the first assumption, as it implies the existence of two distinct paths in layer $\ell = 1$, violating uniqueness.

The two assumptions of Proposition 2 may seem restrictive at first, but they often hold in indoor environments. Fig. 3 shows an example, where $\mathbf{G}$ denotes the goal, $R(\cdot)$ denotes a room, and C_1, C_2 represent specific conditions, namely C_1: open door and C_2: dangerous area. When both C_1 and C_2 are false, Proposition 2 holds, and HCOA* successfully finds the optimal path (shown in green). However, when C_1 is true, the path produced by HCOA* (still shown in green) becomes suboptimal. This occurs because the room is represented, in this example, by its center node. For large rooms such as corridors (e.g., $R(1)$), this representation becomes misleading. Additionally, if C_2 is true, the optimal path requires moving from $R(2)$ to $R(1)$ and then back to $R(2)$. This cycle cannot be captured by COA* operating at the room layer, as it computes acyclic paths. Hence, HCOA* produces a suboptimal path in the places layer in this case (shown in blue).

In general, the spatial representation of nodes in higher layers, as well as their computed semantic classes, can significantly affect HCOA*'s performance, potentially leading to suboptimal solutions. Nevertheless, as demonstrated in Section VI, HCOA* manages to find the optimal path in most tested scenarios. One potential improvement is to refine the representation of rooms by associating multiple nodes with each room, for example, using nodes near doors rather than using a single node at the geometric center of the room.

Regarding the computational complexity of the algorithm, note that, in the worst case, COA* with an admissible heuristic has the same complexity as Dijkstra's algorithm, which is $\mathcal{O}(|E^0| + |V^0| \log |V^0|)$ when implemented with a Fibonacci heap. Hence, the worst-case complexity of HCOA* is given by $\mathcal{O}(\sum_{\ell=0}^{L}(|E^{\ell'}| + |V^{\ell'}| \log |V^{\ell'}|))$, where $|V^{\ell\prime}|$ and $|E^{\ell\prime}|$ denote the number of vertices and edges, respectively, of the pruned graph at each layer ℓ.

V. Semantic Class Prediction

Let $\phi_V^\ell : V^\ell \to \mathcal{K}$ be the function that assigns semantic classes to nodes in layers $\ell = 1, \ldots, n$. Ideally, given an optimal path $\pi^*(v_s, v_g)$ computed in G^0, the semantic class of a node $P \in V^\ell$ ($\ell = 1, \ldots, n$) is determined conservatively by selecting the highest semantic class present in the nodes in $\ell = 0$ that both belong to the optimal path and are part of the subgraph $G^{0'}(P) = \{u \in V^0 : p(u, \ell) = P, \text{ where } P \in V^\ell\}$ induced by P on layer 0. Formally,

$$\phi_V^\ell(P) = \max_{u \in \pi'(P)} \phi_V^0(u), \tag{3a}$$

$$\pi'(P) = \pi^*(v_s, v_g) \cap G^{0'}(P). \tag{3b}$$

This process follows a bottom-up approach, necessitating the computation of the optimal path in layer $\ell = 0$. Given that HCOA* works top-down, we seek alternative methods to find the semantic classes of the higher layers. We introduce three methods for predicting the semantic class of higher-layer nodes: a Majority-Class (MC), a k-Nearest Neighbors (kNN), and a Graph Neural Network (GNN) approach. The latter two methods require a supervised training phase, which necessitates the construction of a dataset.

To this end, we construct a dataset $\mathcal{D} = \{G^{0'}(P), \phi_V^\ell(P)\}$ by selecting nodes $P \in V^\ell$ for $\ell \neq 0$ and extracting their induced subgraphs $G^{0'}(P)$. We then assign semantic classes to the nodes in $G^{0'}(P)$ based on the given task. For navigation tasks involving object avoidance, we determine region priorities by placing disks of a specified radius and semantic class around certain objects. Next, we run COA* on $G^{0'}(P)$ and compute $\phi_V^\ell(P)$, where the start is a border node.

A border node $v^* \in V^0$ is a node that is connected by an edge to another (border) node $u^* \in V^0$ whose ancestor in layer ℓ is different from that of v^*. Formally, there exists $e = (v^*, u^*)$ where $v^*, u^* \in V^0$ and $p(v^*, \ell) \neq p(u^*, \ell)$. Border nodes play a crucial role in the classification of $P \in V^\ell$. If the start and goal nodes do not share the same ancestor, the optimal path must traverse a border node in layer ℓ. Conversely, if both nodes have P as their ancestor, the classification of P becomes irrelevant, as HCOA* will not need to compute a path in layer ℓ.

A. Majority-Class Method

The Majority-Class (MC) method is a simple and fast approach for predicting the semantic class of a node $P \in V^\ell$ for $\ell \neq 0$. The predicted class is determined by counting the occurrences of each semantic class in the induced subgraph $G^{0'}(P)$ and selecting the most frequent one. That is,

$$\widehat{\phi}_V^\ell(P) = \arg\max_{k \in \mathcal{K}} N_V(G^{0'}(P), k), \tag{4}$$

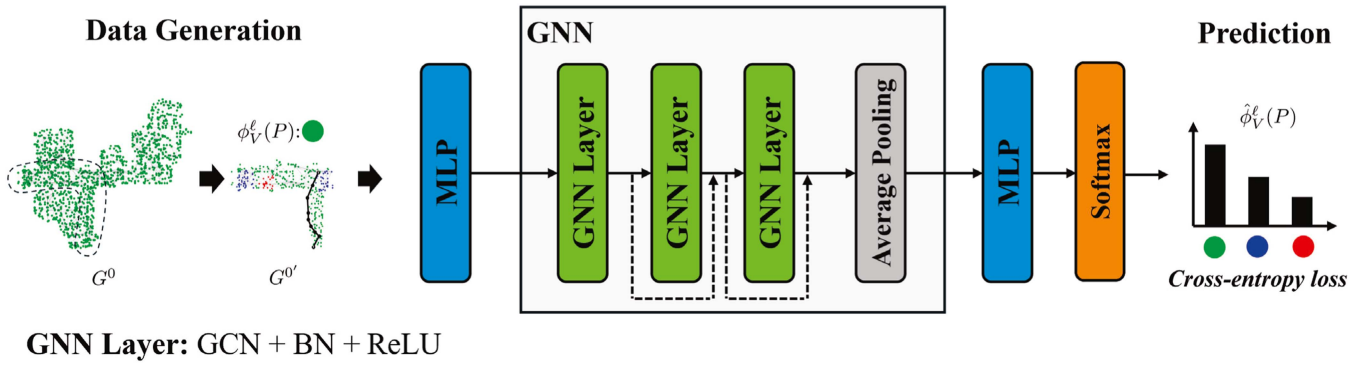


Fig. 4. Proposed GNN architecture for node $P \in V^\ell$ ($\ell \neq 0$) classification, utilizing the semantic classes (green, blue, red) of nodes in $\ell = 0$. The dataset $\mathcal{D} = \{G^{0'}(P), \phi_V^\ell(P)\}$ is generated by running COA* on induced subgraphs of nodes. The network consists of two MLPs for pre-processing and post-processing, three GNN layers with skip connections, and an average pooling operator. Training is conducted using cross-entropy loss over the semantic classes.

where $N_V(G, k) = |\{v \in G : \phi_V^0(v) = k\}|$ is the number of nodes of class k in G.

B. K-Nearest Neighbors Method

The kNN algorithm [19] is a non-parametric, instance-based learning method that classifies data points based on the majority vote of their k nearest neighbors in the feature space. Given the induced subgraph $G^{0'}(P)$ with node set $V^{0'}$, we extract a graph-level feature vector f consisting of: (i) the proportion of each semantic class in the graph

$$\frac{N_V(G^{0'}(P), k)}{|V^{0'}|}, \quad k \in \mathcal{K}, \tag{5}$$

and (ii) the majority class among the border nodes $\arg\max_{k\in\mathcal{K}} N_{V^*}(G^{0'}(P), k)$, where V^* denotes the set of border nodes and k the index of a semantic class (not to be confused with k used in kNN). We intentionally limit the number of features, as the inference time of kNN increases with the dimensionality of the feature space.

To ensure uniform scaling, we standardize all feature vectors f. Graph similarity is measured via Euclidean distance, and classification is performed by assigning the majority class among the k nearest neighbors from the training set:

$$\widehat{\phi}_V^\ell(P) = \arg\max_{k\in\mathcal{K}} \sum_{i=1}^{\mathsf{k}} \mathbb{1}[\phi_V^\ell(P_i) = k], \tag{6}$$

where $P_1, P_2, \ldots, P_{\mathsf{k}}$ are the k nearest neighbors of P in the training set, and $\mathbb{1}[\cdot]$ is the indicator function that equals 1 if the condition is true and 0 otherwise.

C. Graph Neural Network Method

The GNN method is a supervised learning approach that leverages the graph structure and features through message passing. The proposed model is shown in Fig. 4. We use border nodes as input features, which are concatenated with the semantic classes of the nodes in $G^{0'}$ represented as one-hot encodings. A 2-layer MLP is used to preprocess the input.

The GNN consists of three layers, each incorporating a Graph Convolutional Network (GCN) operator [20] for message passing, followed by batch normalization and a ReLU activation. We adopt a 3-layer architecture, as deeper GNNs tend to suffer from oversmoothing and slower inference, both of which are critical limitations in path-planning applications. To mitigate oversmoothing, we introduce skip connections between the GNN layers. Additionally, we apply average pooling to handle environments of varying sizes. The pooled representation is then processed through another 2-layer MLP, followed by a softmax function. Finally, we use cross-entropy as the loss function for node classification. Formally, given a dataset $\mathcal{D}$ and learnable parameters $\vartheta_{\text{GNN}}, \vartheta_{\text{MLP}}$, the predicted semantic class is obtained by

$$\min_{\vartheta_{\text{GNN}}, \vartheta_{\text{MLP}}} \sum_{(G^{0'}(P), \phi_V^\ell(P)) \in \mathcal{D}} \mathcal{L}_{\text{CE}} \left(\widehat{\phi}_V^\ell(P), \phi_V^\ell(P) \right), \tag{7}$$

where $\widehat{\phi}_V^\ell(P) = f(G^{0'}(P); \vartheta_{\text{GNN}}, \vartheta_{\text{MLP}})$ represents the model function parametrized by $\vartheta_{\text{GNN}}, \vartheta_{\text{MLP}}$.

VI. Simulations

We perform simulations on two publicly available datasets generated with Hydra [2]: the uHumans2 office 3DSG (Fig. 1) and the subway 3DSG (Fig. 6(a)). Fig. 1 also shows the layer names. The `place` layer ($\ell = 0$) is a graph with nodes as obstacle-free locations and edges indicating traversability. The `room` layer ($\ell = 1$) consists of nodes representing room centers and edges connecting neighboring rooms.

Our task involves navigation with object or room avoidance. The pipeline proceeds as follows: we first translate safety-related commands (e.g., "avoid computers") into semantic classes at the `place` layer, and impose a total order based on their priority. Next, depending on the room classification method employed, we train the corresponding models to predict semantic room classes. Finally, we select the best-performing model and execute HCOA* to compute the path.

In all simulations, we impose two safety constraints resulting in three semantic classes $\mathcal{K}$, ordered by decreasing priority. We leverage the `room` and `object` layers to identify relevant rooms and objects, and assign semantic labels to nearby `place` nodes accordingly. The resulting classes are visualized (in decreasing priority) as: 1 (Green), 2 (Blue), and 3 (Red).

The first section presents the comparison of the three methods for semantic class prediction. The second section showcases path-planning scenarios in the office 3DSG, comparing HCOA* to COA*. We use the same weight function across all graph

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

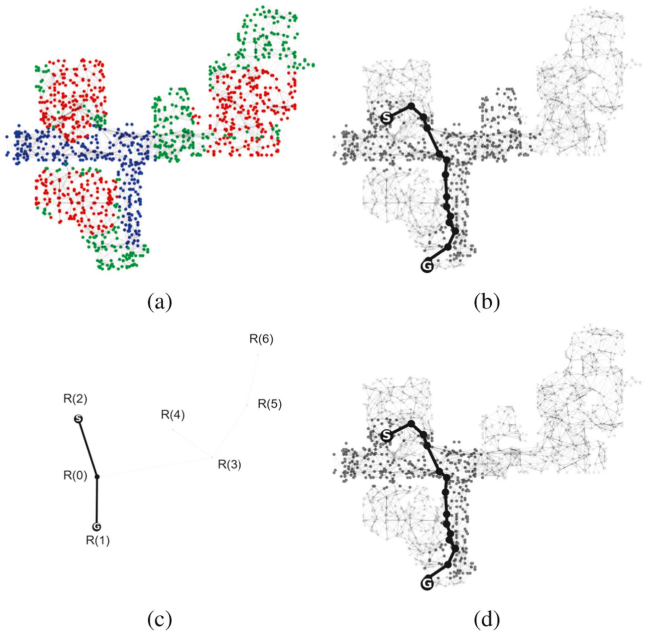


Fig. 5. (a) `Place` subgraph of the uHumans2 office (52m× 45 m) with its semantic classes (green, blue, red); (b-d) The starting node is denoted by **S**, and the goal node by **G**. The path of each algorithm is shown in black. Expanded nodes are depicted in dark gray, while unexpanded nodes in light gray: (b) Path of COA* on the `place` subgraph; (c) Path of HCOA* on the `room` subgraph; (d) Path of HCOA* on the `place` subgraph.

layers and denote it by w for simplicity, corresponding to the Euclidean distance between nodes. We also include results from a modified A* algorithm, denoted MA* (similar to [9]), to demonstrate the importance of imposing a total order over semantic classes. In this variant, the edge cost in A* is modified to $w' = w + \alpha^k$, where $k \in \mathcal{K}$ and α is a scaling factor. The third section presents analogous experiments in the subway 3DSG, also comparing HCOA* to COA* and MA*. In these experiments, HCOA*-GNN utilizes the best GNN model from the previous section for room inference.

All the simulations were performed using Python 3.8.10 and PyTorch 2.4.1 on a computer with 2.2 GHz, 12-Core, Intel Core i7-8750H CPU, 16 GB RAM and an Nvidia RTX 2060 GPU, 6 GB VRAM.

A. Performance Metrics

To evaluate the performance of the prediction of a room's semantic class, we compute the accuracy, which measures the proportion of correctly classified rooms in each dataset:

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} \mathbb{1}[\widehat{\phi}_V^{\ell}(P_i) = \phi_V^{\ell}(P_i)]}{n}, \tag{8}$$

where n is the number of samples. For the planning scenarios, we compute the algorithms' computational time along with the number of expanded nodes (Line 8 in Algorithm 1).

B. Semantic Class Prediction

In this section, we present the results from predicting the semantic class of `room` nodes. To generate the dataset, we constructed 14,000 graphs by extracting the induced subgraphs from all rooms in the Office 3DSG and randomly assigning semantic classes to nodes within a disk of randomly chosen center locations, repeating this process 2,000 times per room. Then, we ran COA* to determine the semantic class of the room, starting from a randomly selected border node. The dataset was split into 80% training, 10% validation, and 10% testing. The GNN model was trained using the Adam optimizer [21] with a learning rate of 10^{-2}, over 1,600 epochs, a dropout rate of 0.2 and a batch size of 64. The GNN and MLP layers contain 32 neurons. For training the kNN model, we set $\mathsf{k} = 5$, and used half of the training dataset.

Table I summarizes the results of `room` node classification. For the test set, we analyzed the performance separately for

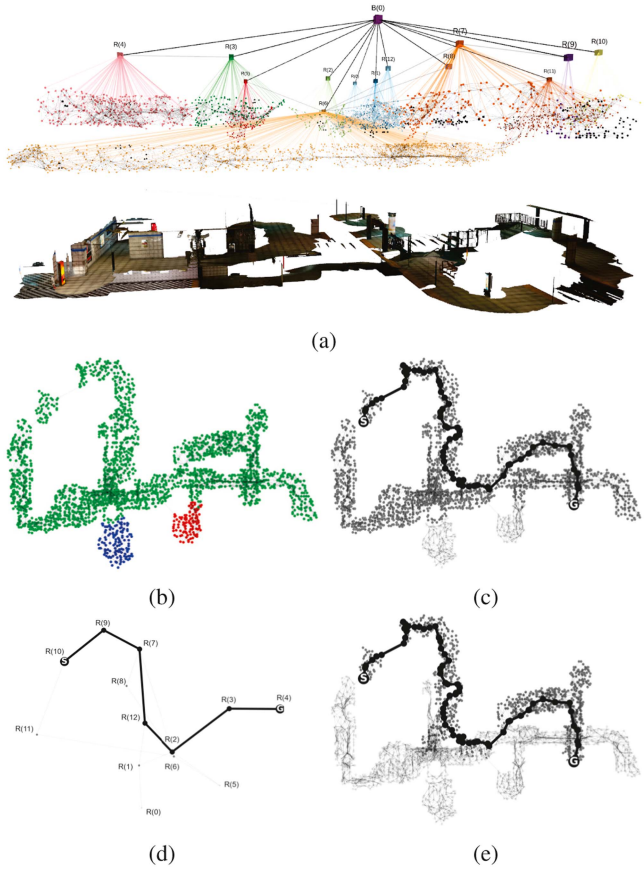


Fig. 6. (a) uHumans2 subway scene; (b) `Place` subgraph of the uHumans2 subway (88m× 60 m) with its semantic classes (green, blue, red); (b)–(d) The starting node is denoted by **S**, and the goal node by **G**. The path of each algorithm is shown in black. Expanded nodes are depicted in dark gray, while unexpanded nodes in light gray: (b) Path of COA* on the `place` subgraph; (c) Path of HCOA* on the `room` subgraph; (d) Path of HCOA* on the `place` subgraph.

TABLE I
SEMANTIC CLASS PREDICTION

Metrics	MC	kNN	GNN
Training Time	-	10 s	97 min
Validation Acc. (%)	42.79	52.43	**63.43**
Test Acc. (1-20 bn Rooms) (%)	57.00	70.50	**74.00**
Test Acc. (21-30 bn Rooms) (%)	33.75	48.50	**58.25**
Test Acc. (31-40 bn Rooms) (%)	37.00	44.25	**54.75**
Test Acc. (41-50 bn Rooms) (%)	41.50	43.00	**47.50**

TABLE II
PATH-PLANNING ON UHUMANS2 OFFICE SCENE

Algorithm	Exp. Nodes	Time (10^{-3}s)	Opt. Path
HCOA*-MC	**412**	**4.2 ± 2.5**	✓
HCOA*-kNN	**412**	5.6 ± 0.2	✓
HCOA*-GNN	**412**	11.5 ± 2.5	✓
COA*	549	4.9 ± 0.4	✓

TABLE III
PATH-PLANNING ON UHUMANS2 SUBWAY SCENE

Algorithm	Exp. Nodes	Time (10^{-3}s)	Opt. Path
HCOA*-MC	**1029**	**9.3 ± 4.5**	✓
HCOA*-kNN	**1029**	15.6 ± 5.1	✓
HCOA*-GNN	**1029**	36.6 ± 4.4	✓
COA*	2480	18.3 ± 4.5	✓

rooms with different numbers of border nodes (bn). The results indicate that the GNN approach outperforms the other two methods. This result is expected, as the GNN learns complex spatial, structural, and semantic features, unlike the MC baseline and the kNN method, which relies on hand-crafted features. Furthermore, we observe that all methods achieve higher accuracy for rooms with fewer bns. These rooms are typically smaller and have only one entrance, resulting in spatially clustered border nodes. In such cases, features considering the border nodes' semantics lead to improved classification performance. The training accuracy of the GNN is 67.94%, while that of the kNN is 60.50%. Note that the training and validation accuracies reported in Table I correspond to the full set of rooms, irrespective of the number of border nodes.

C. Path-Planning on uHumans2 Office Scene

In this section, we perform multiple path-planning scenarios in the office 3DSG shown in Fig. 1. The robot must navigate from a start location to a goal while satisfying the following safety constraints, prioritized in descending order: (i) avoid computers, and (ii) avoid entering room R(0), a typically crowded area. To enforce these constraints, we assign semantic labels to `place` nodes. Specifically, we set $\phi_V^0(v) = 3$ for all $v \in C$, where C denotes the set of `place` nodes within a radius $r = 3$m of any computer in the scene. Formally, $C = \{v \in G^0 : \|x(v) - x(o)\|_2 \leq r, \ \forall o \in O\}$ where $x(\cdot)$ denotes the location of a `place` node v or `object` o, and O is the set of all computers in the 3DSG. We also assign $\phi_V^0(v) = 2$ for all $v \in$ R(0), thereby penalizing passing through this room. The rest of the nodes have $\phi_V^0(v) = 1$.

First, we evaluate the computational efficiency of the algorithm by performing 1,000 runs with fixed start and goal. The starting node v_s is located in room R(2), and the goal v_g is in room R(1). Fig. 5(a) shows the `place` layer, where nodes are color-coded according to their semantic class. The computed path of COA* in the `place` layer is shown in Fig. 5(b). Figs. 5(c)–(d) illustrate the results of HCOA* in the `room` and `place` layers. The figures also show the expanded nodes for each algorithm. In particular, COA* expanded nodes in two additional rooms compared to HCOA*.

The exact number of expanded nodes is provided in Table II, where HCOA* demonstrates a 25% reduction compared to COA*. Table II also presents the computational time of the algorithms. We observe that HCOA*-MC achieves the best performance in terms of computational efficiency, reducing the execution time by 14% compared to COA*. However, HCOA*-GNN has the highest computational time, as GNN inference can be more time-consuming than graph search in small graphs (the entire `place` subgraph contains only 1,314 nodes). Additionally, all classification methods classify R(0) as class 2 and R(2) as class 3. However, while GNN and kNN assign class 3 to R(3), MC classifies it as class 1. Although this misclassification does not impact the final path in the current scenario, it could lead to suboptimal solutions in more complex room graphs, as the robustness of the algorithm is highly dependent on the structure of the room graph.

Finally, all four approaches successfully compute the optimal path in the `place` layer, as shown in Fig. 5. It is important to note that while the first assumption of Proposition 2 holds, the second assumption does not. However, this does not affect the results in this case, as the optimal path in the `place` layer does not require deviations into rooms outside the unique path in the `room` layer to achieve a lower-cost trajectory.

To further evaluate the suboptimality of HCOA*, we conducted 500 scenarios with varied start and goal locations. HCOA* successfully found the optimal path in 96.95% of the cases. This outcome is highly dependent on the representation of the `room` layer; modeling a room solely by its centroid can lead to suboptimal paths, particularly in elongated or non-orthogonal rooms (e.g., R(0)) where the centroid may not accurately reflect spatial connectivity (see Fig. 3).

We also report results from the MA* implementation. When $\alpha = 2$, performance drops to 80.38% (comparing to COA*), while $\alpha = 10$ yields 99.84%. For larger values, accuracy approaches 100%, but for $\alpha > 10^5$, it degrades (e.g., 95.25%), indicating numerical instability. These findings show that MA* requires careful tuning of α based on the graph size, Euclidean distances, and the number of semantic classes, highlighting the benefit of using a total order as in COA* and HCOA*.

D. Path-Planning on uHumans2 Subway Scene

We performed similar simulations to those in Section VI-C on the two-floor subway 3DSG, which consists of 2,732 `place` nodes, shown in Fig. 6(a). The safety constraints are prioritized in descending order: (i) avoid room R(5), where stair repairs are in progress, and (ii) avoid room R(0), which is crowded. Accordingly, we assign semantic classes as follows: $\phi_V^0(v) = 3$ for all $v \in$ R(5), $\phi_V^0(v) = 2$ for all $v \in$ R(0), and $\phi_V^0(v) = 1$ for all other nodes.

To evaluate the computational efficiency of the algorithms, we performed 1,000 runs with fixed start and goal locations. The starting node v_s is located in room R(10), and the goal node v_g is in room R(4). Fig. 6(b) shows the `place` layer, while Figs. 5(b)–(d) depict the computed paths of COA* and HCOA*, along with the corresponding expanded nodes.

Table III summarizes the results. HCOA* achieves a 59% reduction in the number of expanded nodes compared to COA*. In terms of computational time, HCOA*-MC and HCOA*-kNN yield reductions of 49% and 15%, respectively.

We also assessed the suboptimality of the proposed algorithm by running 500 scenarios with varied start and goal locations. HCOA* successfully finds the optimal path in 70.56% of the cases. This decrease compared to the office environment is attributed to the increased complexity introduced by the two-floor

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

layout and the irregular geometry of the rooms. For comparison, the results of MA* show that with $\alpha = 2$, the performance is 72.42%, while $\alpha = 10$ yields 99.16%. For larger values, accuracy approaches 100%, but degrades for $\alpha > 10^5$ again indicating numerical instability.

VII. Conclusion

In this letter, we have addressed the problem of robot navigation in 3D geometric/semantic environments by leveraging the environment's hierarchy for efficient path-planning. We have introduced Hierarchical Class-ordered A* (HCOA*), an algorithm that exploits a total order over semantic classes to guide the search process while significantly reducing computational effort. To classify higher-layer nodes, we proposed three methods: a Majority-Class method, a k-Nearest Neighbors method, and a Graph Neural Network method. Through simulations on two 3D Scene Graphs, we showed that HCOA* effectively reduces computational cost compared to other state-of-the-art methods. Specifically, our results show that HCOA* reduces the computational time of navigation by up to 50%, while maintaining near-optimal performance.

Future work will address the limitations of using a total order over semantic classes for navigation. While total ordering offers a simple and interpretable mechanism for prioritizing semantic categories, it is insufficient for capturing more complex or temporally dependent constraints (e.g., avoid kitchen after visiting bathroom). To handle such cases, we plan to incorporate temporal logic [22] and explore partial ordering schemes. These enhancements will also necessitate extending our current path-planning framework to support online replanning capabilities. To this end, we aim to develop an online path-planning framework based on HCOA*, enabling adaptive decision-making in real-time, dynamic scenarios.

References

[1] I. Armeni et al., "3D scene graph: A structure for unified semantics, 3D space, and camera," in *Proc. IEEE Int. Conf. Comput. Vis.*, Seoul, Korea, 2019, pp. 5663–5672.

[2] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3D scene graph construction and optimization," in *Proc. Robot.: Sci. Syst.*, New York, NY, USA, 2022 .

[3] N. Hughes et al., "Foundations of spatial perception for robotics: Hierarchical representations and real-time systems," *Int. J. Robot. Res.*, vol. 43, no. 10, pp. 1457–1505, Feb. 2024.

[4] A. Botea, M. Müller, and J. Schaeffer, "Near optimal hierarchical path-finding (HPA*)," *J. Game Develop.*, vol. 1, pp. 1–30, Jan. 2004.

[5] J. A. Fernandez and J. Gonzalez, "Hierarchical graph search for mobile robot path planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, Leuven, Belgium, 1998, vol. 1, pp. 656–661.

[6] C. W. Warren, "Fast path planning using modified A* method," in *Proc. IEEE Int. Conf. Robot. Automat.*, Atlanta, GA, USA, 1993, vol. 2, pp. 662–667.

[7] P. Kremer, H. Bavle, J. L. Sanchez-Lopez, and H. Voos, "S-Nav:Semantic-Geometric Planning for Mobile Robots," 2023, *arXiv:2307.01613*.

[8] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. TSSC-4, no. 2, pp. 100–107, Jul. 1968.

[9] Q. Serdel, J. Marzat, and J. Moras, "SMaNa: Semantic mapping and navigation architecture for autonomous robots," in *Proc. Int. Conf. Informat. Control Autom. Robot.*, Rome, Italy, 2023, pp. 453–464.

[10] D. Wooden and M. Egerstedt, "On finding globally optimal paths through weighted colored graphs," in *Proc. IEEE Conf. Decis. Control*, San Diego, CA, USA, 2006, pp. 1948–1953.

[11] J. Lim and P. Tsiotras, "A generalized A* algorithm for finding globally optimal paths in weighted colored graphs," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Xi'an, China, May-Jun. 2021, pp. 7503–7509.

[12] J. Lim, O. Salzman, and P. Tsiotras, "Class-ordered LPA*: An incremental-search algorithm for weighted colored graphs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Prague, Czech Republic, 2021, pp. 6907–6913.

[13] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning a*," *Artif. Intell.*, vol. 155, no. 1, pp. 93–146, 2004.

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

[15] F. Ngom, H. Y. Zhang, L. Zhang, K. Godary-Dejean, and M. Huchard, "IntelliMove: Enhancing robotic planning with semantic mapping," in *Proc. Towards Auton. Robotic Syst.*, London, U.K., 2024, pp. 72–83.

[16] A. Ray, C. Bradley, L. Carlone, and N. Roy, "Task and motion planning in hierarchical 3D scene graphs," 2024, arXiv:2403.08094.

[17] A. Kurenkov, R. Mart'in-Mart'in, J. Ichnowski, K. Goldberg, and S. Savarese, "Semantic and geometric modeling with neural message passing in 3D scene graphs for hierarchical mechanical search," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 11227–11233.

[18] R. Talak, S. Hu, L. Peng, and L. Carlone, "Neural trees for learning on graphs," in *Proc. Conf. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 26395–26408.

[19] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. TIT-13, no. 1, pp. 21–27, Jan. 1967.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, Toulon, France, Apr. 2017, pp. 24–26.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, San Diego, CA, USA, May 2015, pp. 07–09.

[22] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Trans. Robot.*, vol. 25, no. 6, pp. 1370–1381, Dec. 2009.