2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Relationship-Aware Hierarchical 3D Scene Graph for Task Reasoning

Albert Gassol Puigjaner, Angelos Zacharia, Kostas Alexis

***Abstract*— Representing and understanding 3D environments in a structured manner is crucial for autonomous agents to navigate and reason about their surroundings. While traditional Simultaneous Localization and Mapping (SLAM) methods generate metric reconstructions and can be extended to metric-semantic mapping, they lack a higher level of abstraction and relational reasoning. To address this gap, 3D scene graphs have emerged as a powerful representation for capturing hierarchical structures and object relationships. In this work, we propose an enhanced hierarchical 3D scene graph that integrates open-vocabulary features across multiple abstraction levels and supports object-relational reasoning. Our approach leverages a Vision Language Model (VLM) to infer semantic relationships. Notably, we introduce a task reasoning module that combines Large Language Models (LLM) and a VLM to interpret the scene graph's semantic and relational information, enabling agents to reason about tasks and interact with their environment more intelligently. We validate our method by deploying it on a quadruped robot in multiple environments and tasks, highlighting its ability to reason about them.**

I. INTRODUCTION

A central challenge in spatial perception for robotics is constructing 3D representations that are both structured and semantically meaningful. Humans naturally perceive and manipulate scenes by recognizing objects, their properties, and relationships, including hierarchical structures such as rooms within floors or buildings. For autonomous agents, this requires scalable, online representations that support multiple levels of abstraction and reasoning about object relationships.

Traditional Simultaneous Localization and Mapping (SLAM) methods reconstruct metric maps from sensors like cameras [1], LiDARs [2], radars [3], or Inertial Measurement Units (IMU) [4], and can be extended to closed- [5] or open-vocabulary [6] metric-semantic maps using vision foundation models [7], [8]. However, these approaches lack a higher level of abstraction and object reasoning.

3D scene graphs [5], [9], [10], [11], [12], [13] address this gap by capturing hierarchical and relational information. Hierarchical models [9], [10], [11] represent indoor scenes at multiple abstraction levels (*e.g.,* objects, rooms, buildings) while also encoding geometrical inter- and intra-layer relationships (*e.g.,* an object being inside a room). Other works [12], [13] generate object-level scene graphs from RGB point clouds, predicting geometric, comparative, and semantic relationships, while recent methods [14] leverage a Vision Language Model (VLM) to introduce open-vocabulary object relationships. More recently, incremental approaches [15], [16] construct open-vocabulary scene graphs, without explicit object-level relationships.

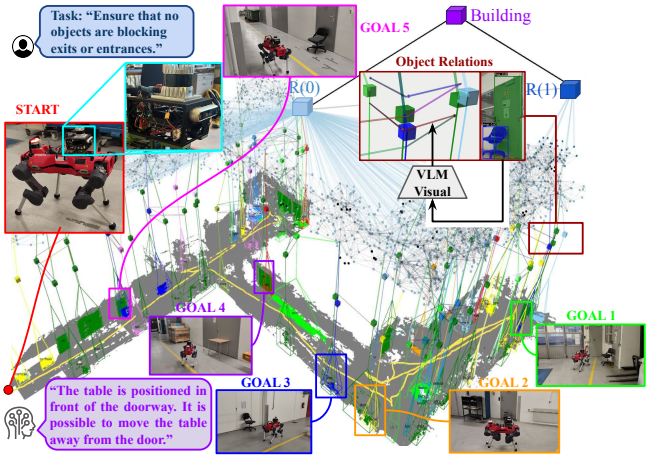


Fig. 1: Task reasoning example. We deploy REASONINGGRAPH on a quadruped robot, which incrementally builds an open-vocabulary, relationship-aware hierarchical scene graph of the environment during autonomous exploration. Leveraging open-vocabulary and object-relational embeddings, REASONINGGRAPH identifies task-relevant objects and reasons about their interactions. In this example, it identifies all the objects (chairs, a table, and a trash can) that are blocking the exits.

In this work, we propose REASONINGGRAPH, a framework for incrementally constructing a reasoning-enhanced hierarchical 3D scene graph that integrates open-vocabulary features across multiple levels of abstraction and supports object-relational reasoning. Additionally, we introduce a task reasoning module that, given a task that may require object-interaction reasoning (*e.g.,* "prepare the room for a meeting", "ensure that exits are not blocked"), leverages the semantic and relational information in our graph to decompose the task into subtasks, identify the relevant objects, and evaluate which subtasks need to be executed. Our contributions are:

- We extend hierarchical 3D scene graphs with open-vocabulary features across multiple abstraction levels.
- We leverage a VLM to infer object relationship features for a richer context-aware representation.
- We propose a reasoning module that combines Large Language Models (LLM) and a VLM to process natural language tasks, predict relevant objects, and assess object-interaction feasibility.
- We quantitatively evaluate REASONINGGRAPH's ability to encode open-vocabulary objects, showing competitive performance against strong baselines.
- We demonstrate the benefit of incorporating object relations together with our reasoning module to reason about complex tasks. Furthermore, we deploy our method on a quadruped robot, demonstrating its ability to build the scene graph online and reason about tasks.

Autonomous Robots Lab, Norwegian University of Science and Technology (NTNU), Trondheim, Norway, albert.g.puigjaner@ntnu.no
This work was supported by the European Commission Horizon Europe grant SYNERGISE (EC 101121321).

In the remainder of this paper, we first review related literature (Section II) and then define the problem addressed (Section III). We proceed with a detailed description of the proposed method (Section IV), followed by its performance evaluation (Section V). Conclusions are drawn in Section VI.

II. RELATED WORK

Metric-Semantic Representations. Closed-vocabulary semantic SLAM methods aim to build semantically annotated 3D maps of the environment. These approaches typically rely on semantic and panoptic segmentation networks [17], [18] to enhance the 3D representation [5], [19]. With the introduction of vision foundation models such as CLIP [7] and SAM [8], recent works focus on constructing 3D representations enriched with open-vocabulary features that can be easily queried and/or clustered [6]. These methods extract open-vocabulary embeddings from 2D images using vision foundation models and project them into 3D space. However, because they assign features at the point level, they require significant memory and do not scale efficiently. Additionally, these methods lack abstraction, hierarchical structure, and object-level reasoning, limiting their ability to support a higher level of scene understanding.

3D Scene Graphs. Early works [5], [9] introduced 3D scene graphs to model indoor multi-level abstractions, enabling spatial reasoning across agent poses, objects, rooms, and buildings. These methods also established inter- and intra-layer relationships, such as object containment and spatial proximity, to provide a richer structural understanding of the scene. Subsequent works [10], [11] adopted this hierarchical framework and extended it to incrementally build 3D scene graphs in real time. Meanwhile, object-level scene graphs [12], [13] have been proposed to infer geometric, comparative, and semantic relationships from RGB point clouds, further improving contextual reasoning. More recently, VLMs have been leveraged to introduce open-vocabulary relationships, enabling flexible and adaptable semantics in scene graphs. Koch *et al.* [14] propose to distill the knowledge of the visual encoder of InstructBLIP [20] into a Graph Convolutional Neural Network (GCNN) , while Chen *et al.* [21] adopt a similar approach with CLIP [7]. However, these methods typically construct object-level scene graphs offline, requiring a complete point cloud of the scene. Despite these advancements, existing approaches still suffer from several limitations. Namely, they either (a) lack open-vocabulary and relationship reasoning, (b) fail to incorporate hierarchical representations, or (c) are unsuitable for online scene graph construction.

Open-Vocabulary 3D Scene Graphs. Recent works [15], [16] have explored open-vocabulary scene graphs to enable language-grounded navigation. ConceptGraphs [15] incrementally constructs object-level open-vocabulary 3D scene graphs by clustering 3D-projected CLIP embeddings. It further predicts object relationships by prompting GPT-4 with geometric information and summarized image captions of objects. Additionally, it enables language-grounded object search by querying GPT-4 with the objects' geometric data and captions. However, this approach is limited to small-scale scenes and lacks a hierarchical structure in its representation.

Building upon similar ideas, HOV-SG [16] introduces a hierarchical open-vocabulary 3D scene graph, organized into building, floors, rooms, objects, and a navigational graph. In addition to attaching CLIP embeddings to detected objects, where object embeddings are projected into 3D using depth images and averaged across multiple views, HOV-SG also extends open-vocabulary features to floors and rooms within its hierarchy. During object search, GPT-4 parses natural language queries into structured attributes: the room, floor, and object mentioned in the query. These attributes are then matched within the hierarchy by computing cosine similarities with the open-vocabulary embeddings. While this method integrates both hierarchical and open-vocabulary representations, it lacks relationship reasoning, which could further improve language-grounded task reasoning by capturing relevant interactions between objects.

III. PROBLEM STATEMENT

A hierarchical 3D scene graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{E} \rangle$, consisting of N hierarchical layers, is defined by a set of nodes $\mathcal{N} = \{\mathcal{N}_{L_n}\}_{n=1}^{N}$, which contains all nodes across the layers, and a set of edges $\mathcal{E}$ representing geometric or semantic relationships between them. Layers are organized from bottom to top, with each successive layer representing entities at a higher level of abstraction. Each node has geometric properties, such as position or centroid, and may include semantic and/or open-vocabulary attributes. Each layer L_n includes intra-layer edges $\mathcal{E}_{L_n}^{L_n}$, which encode relational information. Additionally, inter-layer edges $\mathcal{E}_{L_n}^{L_{n+1}}$ connect nodes between layers. Formally, the sets of nodes and edges of the graph are defined as:

$$\mathcal{N} = \bigcup_{n=1}^{N} \mathcal{N}_{L_n}, \tag{1}$$

$$\mathcal{E} = \left(\bigcup_{n=1}^{N-1} \left(\mathcal{E}_{L_n}^{L_n} \cup \mathcal{E}_{L_n}^{L_{n+1}} \right) \right) \cup \mathcal{E}_{L_N}^{L_N}. \tag{2}$$

Objective 1: Given a sequence of $M \in \mathbb{N}$ RGB-D frames $\mathcal{I} = \{I_i\}_{i=1}^{M}$, where each frame $I_i = \{I_i^{\text{RGB}}, I_i^{\text{Depth}}\}$, along with corresponding odometry estimates $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{M}$ of an indoor scene, the goal is to incrementally construct $\mathcal{G}$. Additionally, we aim to progressively enrich $\mathcal{G}$ with open-vocabulary semantics and relational information.

Objective 2: Given the scene graph $\mathcal{G}$, the objective is to leverage its open-vocabulary and relational representation to reason about tasks involving object interaction or search, by identifying relevant objects and determining whether interactions among them are needed to accomplish the task.

IV. METHOD

In this section, we briefly introduce our scene graph definition and its construction in Section IV-A, followed by a detailed discussion of open-vocabulary features and relations in Section IV-B. Finally, we present our reasoning module in Section IV-C, which leverages the scene graph information

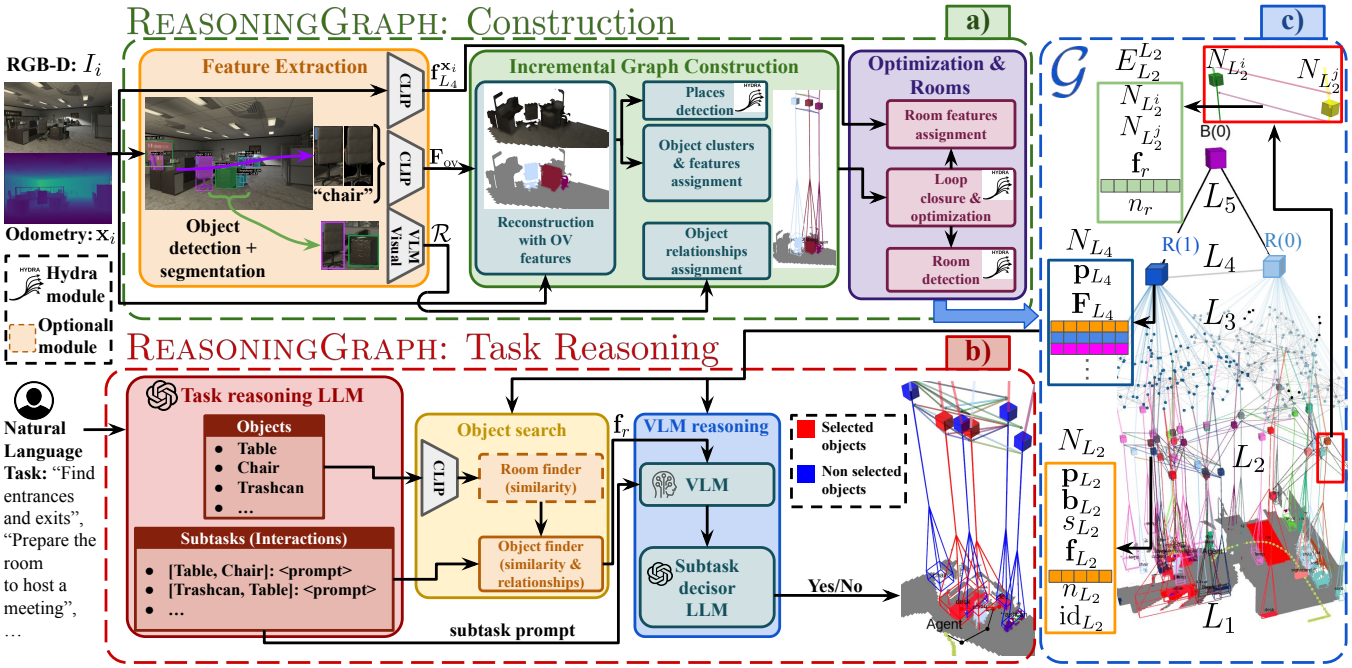


Fig. 2: REASONINGGRAPH overview. **a)** REASONINGGRAPH incrementally builds a hierarchical 3D scene graph $\mathcal{G}$ (**c**)) from RGB-D frames and poses, using an open-vocabulary detector [22] and CLIP [7] embeddings for object representation. Object relations are derived from a VLM [23] visual encoder, while Hydra [10] reconstructs the semantic mesh (L_1), clusters objects (L_2), and detects places and rooms (L_3, L_4). Open-vocabulary features and relations are then assigned to $\mathcal{G}$. **b)** The task reasoning module leverages two LLMs and a VLM. Given a task, the LLM identifies relevant objects and formulates subtasks needing evaluation. These subtasks are evaluated for feasibility by the VLM, with CLIP similarity used for object retrieval.

to reason about a given task. The overall design of our method is illustrated in Fig. 2.

A. Hierarchical Scene Graph Definition and Construction

Following Hydra [10], [11], we define a five-layer hierarchy ($N = 5$), illustrated in Fig. 2c:

- **Metric-Semantic Mesh Layer** L_1: Each node is $N_{L_1} = \{\mathbf{v}, \mathbf{c}, s\} \in \mathcal{N}_{L_1}$, where $\mathbf{v} \in \mathbb{R}^3$ is a mesh vertex, $\mathbf{c} \in [0, 255]^3$ its color, and $s \in \mathbb{N}$ its semantic label. If a vertex belongs to an object, we add a graph edge $E_{L_1}^{L_2} = \{N_{L_1}, N_{L_2}\} \in \mathcal{E}_{L_1}^{L_2}$.
- **Object Layer** L_2: As shown in Fig. 2c, each object node is $N_{L_2} = \{\mathbf{p}_{L_2}, \mathbf{b}_{L_2}, s_{L_2}, \mathbf{f}_{L_2}, n_{L_2}, \mathrm{id}_{L_2}\} \in \mathcal{N}_{L_2}$, where $\mathbf{p}_{L_2} \in \mathbb{R}^3$ is the object centroid, $\mathbf{b}_{L_2} \in \mathbb{R}^6$ its bounding box, $s_{L_2} \in \mathbb{N}$ its label, $\mathbf{f}_{L_2} \in \mathbb{R}^{|\mathbf{f}_{L_2}|}$ an open-vocabulary feature, $n_{L_2} \in \mathbb{N}$ the feature update count, and $\mathrm{id}_{L_2} \in \mathbb{N}$ a variable identifier used for object-level relationships assignment. Objects connect to their nearest place via $E_{L_2}^{L_3} = \{N_{L_2}, N_{L_3}\} \in \mathcal{E}_{L_2}^{L_3}$. Object-level semantic relations (*e.g.,* "a glass is on a table") between two objects ($N_{L_2}^i$ and $N_{L_2}^j$) are captured with $E_{L_2}^{L_2^j} = \{N_{L_2}^i, N_{L_2}^j, \mathbf{f}_r, n_r\} \in \mathcal{E}_{L_2}^{L_2}$, where $\mathbf{f}_r \in \mathbb{R}^{|\mathbf{f}_r|}$ encodes the relation and $n_r \in \mathbb{N}$ counts its updates. Such relations can be visualized in Fig. 2c.
- **Place Layer** L_3: Each node is $N_{L_3} = \{\mathbf{p}_{L_3}\} \in \mathcal{N}_{L_3}$, with $\mathbf{p}_{L_3} \in \mathbb{R}^3$ as centroid. Places connect to their rooms via the graph edge $E_{L_3}^{L_4} = \{N_{L_3}, N_{L_4}\} \in \mathcal{E}_{L_3}^{L_4}$.
- **Room Layer** L_4: Each node is $N_{L_4} = \{\mathbf{p}_{L_4}, \mathbf{F}_{L_4}\} \in \mathcal{N}_{L_4}$, where $\mathbf{p}_{L_4} \in \mathbb{R}^3$ is the room centroid and $\mathbf{F}_{L_4} = [\mathbf{f}_{L_4}^1, \ldots, \mathbf{f}_{L_4}^K]^T \in \mathbb{R}^{K \times |\mathbf{f}_{L_4}|}$ a set of K open-vocabulary feature clusters, with $\mathbf{f}_{L_4}^i \in \mathbb{R}^{|\mathbf{f}_{L_4}|}$ being the i-th open-vocabulary feature cluster of the set (see Fig. 2c for a visual representation of room nodes). Rooms connect to buildings via the edge $E_{L_4}^{L_5} = \{N_{L_4}, N_{L_5}\} \in \mathcal{E}_{L_4}^{L_5}$.
- **Building Layer** L_5: Each node is $N_{L_5} = \{\mathbf{p}_{L_5}\} \in \mathcal{N}_{L_5}$, with $\mathbf{p}_{L_5} \in \mathbb{R}^3$ as centroid.

Having defined the hierarchical scene graph, we now describe its construction (Fig. 2a) from sensor data. We employ Hydra [10] to reconstruct the scene and extract the hierarchical graph layers (L_1 - L_5). Hydra incrementally builds the mesh (L_1), object (L_2), and place (L_3) layers online, while the room layer (L_4), mesh refinement, and pose optimization are updated at a lower frequency. This produces the hierarchical structure, which serves as the basis for open-vocabulary features and relational reasoning enhancement.

The semantic mesh ($\mathcal{N}_{L_1}$) is constructed using Kimera [5] for semantic segmentation and a windowed Voxblox [24] to extract the Truncated Signed Distance Field (TSDF), Euclidean Signed Distance Field (ESDF), and mesh via marching cubes.

Objects ($\mathcal{N}_{L_2}$) are extracted via Euclidean clustering of vertices with the same semantic label, and overlapping objects of the same class are incrementally fused. Places ($\mathcal{N}_{L_3}$) are obtained by sparsifying a Generalized Voronoi Diagram (GVD) derived from the ESDF, connecting the resulting voxels to form a graph. Rooms ($\mathcal{N}_{L_4}$) are detected by dilating the voxel map and pruning the corresponding subgraph of places, such that connected place nodes correspond to rooms.

B. Open-Vocabulary and Reasoning Enhancement

We enhance the graph $\mathcal{G}$ by attaching open-vocabulary features to objects ($\mathcal{N}_{L_2}$) and rooms ($\mathcal{N}_{L_4}$). Furthermore, we compute relational features between objects and incorporate

them as graph edges ($\mathcal{E}_{L_2}^{L_2}$). This enhancement can be visualized in Fig. 2c. Introducing such features requires several steps in our framework. We present these in Algorithms 1 to 4. Next, we provide details on how these features are computed and included in $\mathcal{G}$.

Object Features. In Algorithm 1 (Line 2), we begin by detecting object bounding boxes, segmentation masks and semantic labels ($\mathcal{B}$, $I^{\text{seg}}, \mathcal{S}$) from the input RGB-D frame $I = \{I^{\text{RGB}}, I^{\text{Depth}}\}$ (frame index dropped for simplicity), using an open-set detection and segmentation method such as YOLOe [22]. For each detected object $i \in \{1, \dots, |\mathcal{B}|\}$, we generate two image crops: (a) a masked image $g_{\text{mask}}(I^{\text{RGB}}, I_i^{\text{seg}})$, where the object is isolated with a black background, and (b) a bounding-box crop $g_{\mathcal{B}}(I^{\text{RGB}}, \mathcal{B}_i)$. We then get averaged CLIP [7] embeddings:

$$\mathbf{f}_{\text{ov}}^i = \alpha_{\text{mask}} \mathbf{f}_{\text{CLIP}}(g_{\text{mask}}(I^{\text{RGB}}, I_i^{\text{seg}})) + \alpha_{\mathcal{B}} \mathbf{f}_{\text{CLIP}}(g_{\mathcal{B}}(I^{\text{RGB}}, \mathcal{B}_i)) + \alpha_s \mathbf{f}_{\text{CLIP}}(\mathcal{S}_i), \tag{3}$$

where $\mathbf{f}_{\text{CLIP}}(g_{\text{mask}}(I^{\text{RGB}}, I_i^{\text{seg}}))$ is the embedding of the masked object, $\mathbf{f}_{\text{CLIP}}(g_{\mathcal{B}}(I^{\text{RGB}}, \mathcal{B}_i))$ is the embedding of the cropped object defined by its bounding box and $\mathbf{f}_{\text{CLIP}}(\mathcal{S}_i)$ is the embedding of the object's semantic label. Following [16], we combine both cropped and masked embeddings since this enhances the robustness of the CLIP representation, while the semantic label embedding adds a complementary textual cue. The weights satisfy $\alpha_{\text{mask}} + \alpha_{\mathcal{B}} + \alpha_s = 1$. Finally, the open-vocabulary features of all objects are collected into a vector of feature vectors, denoted as $\mathbf{F}_{\text{ov}}$ (Line 6). This process is illustrated in the Feature Extraction block of Fig. 2a.

Consequently, in Algorithm 2, object features are temporarily attached to the mesh ($\mathcal{N}_{L_1}$) when performing the 3D reconstruction with Voxblox [24] and marching cubes (Line 2). For each object C^i in the clustered mesh (Line 3), with $i \in \{1, \dots, N_C\}$ and $N_C \in \mathbb{N}$ being the number of clusters, we determine if its vertices contain open-vocabulary features. When they do, we average them to get $\mathbf{f}_{\text{mesh}}^i$ (Line 7). If C^i corresponds to a node of our graph ($N_{L_2^i}$), we average its open-vocabulary feature (Lines 10 and 11):

$$\mathbf{f}_{L_2^i} = \frac{n_{L_2^i} \mathbf{f}_{L_2^i} + \mathbf{f}_{\text{mesh}}^i}{n_{L_2^i} + 1}, \quad \mathbf{f}_{L_2^i}, n_{L_2^i} \in N_{L_2^i}. \tag{4}$$

Otherwise, we create a new N_{L_2} node from the cluster C^i and add it to the graph (Line 13). The detected open-vocabulary features are then removed from the mesh ($\mathcal{N}_{L_1}$) to reduce memory usage (Line 16).

Rooms Features. Similarly to objects, we extract open-vocabulary features for rooms. In the feature extraction step (Algorithm 1), our system continuously computes CLIP embeddings ($\mathbf{f}_{L_4}^{\mathbf{x}}$) of the full RGB frames (I^{RGB}) and associates each embedding with the agent's corresponding pose ($\mathbf{x}$) (Line 7). In the room feature assignment module (Algorithm 3; see also Optimization & Rooms in Fig. 2a), we associate all the currently computed full RGB embeddings $\{\mathbf{f}_{L_4}^{\mathbf{x}_0}, \dots, \mathbf{f}_{L_4}^{\mathbf{x}_n}\}$ at timestep n, with rooms based on spatial containment. For each detected room (Line 3), we collect all CLIP embeddings linked to poses that lie within that room's boundaries (Lines 5 and 6). Since CLIP embeddings of a room can vary significantly across viewpoints, we cluster the collected embeddings into K groups using K-Means (Line 9). This produces a set of open-vocabulary feature clusters $\mathbf{F}_{L_4^i}$ associated with the i-th room $N_{L_4^i}$. By clustering embeddings, we assume that images with a similar view frustum produce similar CLIP embeddings.

Algorithm 1 Features Extraction

1: **Input:** $\mathbf{x}$, I
2: $\mathcal{B}, I^{\text{seg}}, \mathcal{S} = \texttt{detect}(I)$ ▷ Object detection
3: **for** $i = 1, \dots, |\mathcal{B}|$ **do** ▷ Average CLIP for each object
4: $\quad \mathbf{f}_{\text{ov}}^i = \texttt{average}_{\text{CLIP}}(I^{\text{RGB}}, I_i^{\text{seg}}, \mathcal{B}_i, \mathcal{S}_i)$ ▷ Eq. (3)
5: **end for**
6: $\mathbf{F}_{\text{ov}} = \{\mathbf{f}_{\text{ov}}^0, \dots, \mathbf{f}_{\text{ov}}^{|\mathcal{B}|}\}$
7: $\mathbf{f}_{L_4}^{\mathbf{x}} = \mathbf{f}_{\text{CLIP}}(I^{\text{RGB}})$ ▷ CLIP of the full input for rooms
8: $\mathcal{R} = \{(i,j) : \mathbf{f}_{\text{VLM}}(I^{\text{RGB}}[\mathcal{B}_i \cup \mathcal{B}_j])\} \ \forall i,j \in \{1, \dots, |\mathcal{B}|\}, i \neq j$ ▷ VLM visual encoder to get object-relation features. Stored in a dictionary of object pairs
9: **return** I^{seg}, $\mathcal{S}$, $\mathbf{F}_{\text{ov}}$, $\mathbf{f}_{L_4}^{\mathbf{x}}$, $\mathcal{R}$

Algorithm 2 Object Features Assignment

1: **Input:** $\mathbf{x}$, I, I^{seg}, $\mathcal{S}$, $\mathbf{F}_{\text{ov}}$, $\mathcal{R}$
2: $\mathcal{N}_{L_1} = \texttt{reconstruct}_{\text{ov}}(\mathcal{N}_{L_1}, \mathbf{x}, I, I^{\text{seg}}, \mathbf{F}_{\text{ov}}, \mathcal{S}, \mathcal{R})$
3: $C = \{C^1, \dots, C^{N_C}\} = \texttt{cluster}(\mathcal{N}_{L_1})$
4: **for** $i = 1, \dots, N_C$ **do** ▷ Iterate over object clusters
5: $\quad \mathbf{f}_{\text{mesh}}^i = \emptyset$, $n = 0$ ▷ Empty feature
6: $\quad$ **if** $\mathcal{N}_{L_1}.\texttt{has\_features}(C^i)$ **then** ▷ Fill feature
7: $\quad\quad \mathbf{f}_{\text{mesh}}^i = \texttt{average}(\mathcal{N}_{L_1}.\texttt{features}(C^i))$, $n = 1$
8: $\quad$ **end if**
9: $\quad$ **if** $N_{L_2^i} \in \mathcal{N}_{L_2}$ **then** ▷ Running average of features
10: $\quad\quad \mathbf{f}_{L_2^i} = \texttt{average}_{L_2}(N_{L_2^i}, \mathbf{f}_{\text{mesh}}^i)$ ▷ Eq. (4)
11: $\quad\quad n_{L_2^i} \mathrel{+}= 1$, $\text{id}_{L_2^i} = L_2^i.\text{id}$
12: $\quad$ **else** ▷ Add new object node to the graph
13: $\quad\quad \mathcal{N}_{L_2} \leftarrow \{C^i.\mathbf{p}_{L_2}, C^i.\mathbf{b}_{L_2}, C^i.s_{L_2}, \mathbf{f}_{\text{mesh}}^i, n, C^i.\text{id}\}$
14: $\quad$ **end if**
15: **end for**
16: $\mathcal{N}_{L_1}.\texttt{remove}(\mathbf{F}_{\text{ov}})$ ▷ Delete mesh features

Object-level Relationships. We enhance our graph representation by explicitly modeling relationships between objects, leveraging the expressive power of a VLM. After detecting objects of the input RGB-D frame (I) in our feature extraction module (see Fig. 2a and Algorithm 1), we extract visual features for each pair of detected objects using a VLM's visual encoder ($\mathbf{f}_{\text{VLM}}$) and store them in a dictionary ($\mathcal{R}$) whose keys are the detected objects' indices (Line 8). Our framework is designed to be compatible with any VLM, as all architectures include a visual encoder, a visual-language adaptor and an LLM. In our graph $\mathcal{G}$, we attach the output of the visual encoder, while the visual-language adaptor and LLM are used in the reasoning module (Section IV-C).

The input to $\mathbf{f}_{\text{VLM}}$ is the cropped region defined by the union of the two objects' bounding boxes. Within this region, the boxes are inpainted into the image using a unique color for each label (see Fig. 4) to explicitly inform the VLM about

Algorithm 3 Room Features Assignment

1: **Input:** $\{\mathbf{f}_{L_4}^{\mathbf{x}_0}, \dots, \mathbf{f}_{L_4}^{\mathbf{x}_n}\}$, $\{\mathbf{x}_0, \dots, \mathbf{x}_n\}$
2: `room_features` $= \{\}$
3: $\mathcal{N}_{L_4} =$ `detect_rooms`$(\mathcal{G})$
4: **for** $i = 0, \dots, n$ **do**
5: $\quad N_{L_4} =$ `find_room`$(\mathcal{N}_{L_4}, \mathbf{x}_i)$ ▷ Room containing $\mathbf{x}_i$
6: $\quad$ `room_features`$[N_{L_4}]$`.append`$(\mathbf{f}_{L_4}^{\mathbf{x}_i})$
7: **end for**
8: **for** $i = 0, \dots, |\mathcal{N}_{L_4}|$ **do** ▷ Room-feature clusters
9: $\quad \mathbf{F}_{L_4^i} =$ `KMeans`$($`room_features`$[N_{L_4^i}])$
10: **end for**

Algorithm 4 Object Relationships Assignment

1: **Input:** $\mathcal{R}$
2: **for** $i = 1, \dots, |\mathcal{N}_{L_2}|; j = 1, \dots, |\mathcal{N}_{L_2}|; j \neq i$ **do**
3: $\quad \mathbf{f}_{\text{VLM}}^{i,j} = \mathcal{R}[(\text{id}_{L_2^i}, \text{id}_{L_2^j})]$
4: $\quad$ **if** $\mathbf{f}_{\text{VLM}}^{i,j} == \emptyset$ **: continue** ▷ Add/Update relations
5: $\quad$ **if** $E_{L_2^i}^{L_2^j} \in \mathcal{E}_{L_2}^{L_2}$ **then** ▷ Update existing relation
6: $\quad\quad \mathbf{f}_r^{i,j} =$ `average`$_r(E_{L_2^i}^{L_2^j}, \mathbf{f}_{\text{VLM}}^{i,j})$ ▷ Eq. (5)
7: $\quad\quad n_r^{i,j} \mathrel{+}= 1$
8: $\quad$ **else** ▷ Add new relationship to the graph
9: $\quad\quad \mathcal{E}_{L_2}^{L_2} \leftarrow \{N_{L_2^i}, N_{L_2^j}, \mathbf{f}_{\text{VLM}}^{i,j}, 1\}$
10: $\quad$ **end if**
11: **end for**
12: $\mathcal{N}_{L_1}$`.remove(`$\mathcal{R}$`.keys())` ▷ Delete relation IDs

the important objects to reason about. These features can later be combined with a text prompt to describe relationships between objects in the graph as natural language.

When reconstructing the mesh (Line 2, Algorithm 2), we attach the detected object indices (called IDs) to the mesh, which are later used to assign an ID for each object node (Lines 11 and 13). In Algorithm 4, we use these IDs to assign or update object-level relationships. If an edge between the objects already exists in the graph, we update its relationship feature using the following running average (Lines 5 to 7):

$$\mathbf{f}_r^{i,j} = \frac{n_r^{i,j}\mathbf{f}_r^{i,j} + \mathbf{f}_{\text{VLM}}^{i,j}}{n_r^{i,j} + 1}, \quad \mathbf{f}_r^{i,j}, n_r^{i,j} \in E_{L_2^i}^{L_2^j}. \tag{5}$$

Otherwise, we create a new edge containing the visual relationship feature (Line 9). Finally, we remove the IDs from the mesh ($\mathcal{N}_{L_1}$), so that they do not interfere with the next iteration when assigning relationship features (Line 12).

C. Task Reasoning

In this section, we present a method that leverages the open-vocabulary semantics and object relationship information encoded in $\mathcal{G}$ to reason about tasks. The goal is to assess task feasibility and generate a high-level plan that can guide downstream planning and scene interaction methods. Given a task in natural language, our approach consists of a three-stage reasoning pipeline as shown in Fig. 2b.

Task Reasoning. We use an LLM to parse the input task. The LLM identifies a list of $N_{\text{LLM}} \in \mathbb{N}$ task-relevant objects and, if the task requires interactions between these objects, it also enumerates those interactions framed as subtasks. For each

Task Reasoning LLM System Prompt

You are given a natural language description of a task. Your goal is:
1. **Identify relevant objects** for the task.
2. Determine whether the task requires reasoning about **object interactions** (e.g., moving, placing, using, etc).
Your output must be in **JSON** format with:
objects: list of relevant objects.
interactions: prompts that formulate the objects interaction subtask (leave empty if the task only involves locating, detecting, or classifying objects).
Example 1. Prompt: "Find a place to sit down." Output: {"objects": ["sofa", "chair", "bed"], "interactions": {}}
Example 2. Prompt: "Save the food for later." Output: {"objects": ["food", "fridge", "freezer"], "interactions": {"1": {"objects": ["food", "fridge"], "prompt": "Check whether the food is stored properly; if not, can it be placed in the fridge?"}, "2": {"objects": ["food", "freezer"], "prompt": "Check whether the food is stored properly; if not, can it be placed in the freezer?"}}}
...

Subtask Decisor LLM System Prompt

You are a decision-making assistant. Given:
- **VLM's prompt:** a question or instruction about a task.
- **VLM's response:** the answer from the VLM.
Decide if the **task should be executed** by evaluating its **necessity, justification** and **feasibility** according to the **VLM**.
Respond with **"Yes"** if the task should be performed or **"No"** if it should not.

Fig. 3: System prompts for task reasoning and subtask decisor LLMs.

subtask, the LLM generates a natural language prompt to assess its feasibility. To ensure consistent and structured output, we design a system prompt, shown in Fig. 3, that guides the LLM to follow this reasoning process and return a single JSON file containing the relevant objects, their subtasks (if required), and corresponding prompts. We further guide the LLM by providing examples of tasks (Fig. 3 bottom part).

Object and Room Search. Given the list of N_{LLM} task-relevant objects from the task reasoning LLM, the next step is to locate these objects within $\mathcal{G}$. To achieve this, we first compute the N_{LLM} CLIP [7] embeddings of the provided object names. We then compare these embeddings to those of the objects already in the scene graph by computing the cosine similarity against their open-vocabulary features $\mathbf{f}_{L_2}$. An object is considered found if its similarity exceeds a predefined threshold. If the task reasoning LLM specifies subtasks between objects, we verify - after matching objects - that a relationship edge exists between the corresponding nodes in the graph. This relationship information is required for the subtask reasoning step performed by the VLM. While objects in $\mathcal{G}$ are associated with semantic labels, we rely on their CLIP embeddings for object search to mitigate label classification errors from the object detector.

Additionally, our method supports focusing on objects in specific rooms. To do this, we compute the cosine similarity between the CLIP embeddings of object names provided by the task reasoning LLM and the embeddings of objects in each of the detected rooms. For each room, we average its $K \times N_{\text{LLM}}$ cosine similarities and select rooms whose average exceeds a given threshold. This allows us to narrow down the object search and reasoning to relevant regions.

VLM Reasoning. If the task requires reasoning about interactions between objects, and the relevant objects have already been located in $\mathcal{G}$, we use a VLM to evaluate whether each subtask is both necessary and feasible. We combine the subtasks generated by the task reasoning LLM with the object-level relationship features ($\mathbf{f}_r$) stored in $\mathcal{G}$. In addition to the subtask generated by the task reasoning LLM, we

	HM3DSem [25]								Replica [26]							
	Acc_5	Acc_{10}	Acc_{25}	Acc_{100}	Acc_{250}	Acc_{500}	AUC_k^{Acc}	#objects	Acc_5	Acc_{10}	Acc_{25}	Acc_{100}	Acc_{250}	Acc_{500}	AUC_k^{Acc}	#objects
ConceptGraphs [15]	**18.5**	**24.70**	**38.05**	**60.29**	**76.58**	**85.98**	**87.27**	**237.78**	**20.70**	**30.02**	**41.72**	**57.47**	**72.85**	**79.94**	**80.50**	**40.5**
HOV-SG [16]	16.09	20.98	32.62	56.78	71.79	81.98	84.46	**405.33**	3.66	6.56	11.26	24.00	43.65	63.27	66.82	**72.63**
REASONINGGRAPH (Ours)	**37.72**	**45.18**	**56.52**	**74.47**	**85.51**	**92.95**	**92.49**	88.33	**53.99**	**58.23**	**63.16**	**73.74**	**87.29**	**88.80**	**87.23**	13.25

TABLE I: Object retrieval quantitative results. **Best result**, **Second-best result**. REASONINGGRAPH outperforms baselines across metrics and datasets. Note that ConceptGraphs [15] and HOV-SG [16] operate on fully open-vocabulary segments, computed by clustering the embeddings of their point clouds.

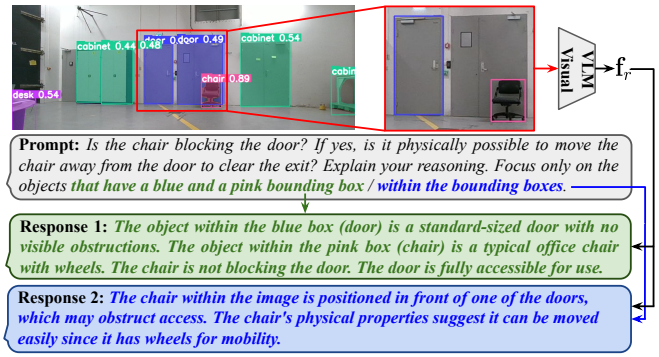


Fig. 4: Guiding the VLM with bounding boxes colors. The VLM is provided with an image with a pair of objects and their inpainted bounding boxes, along with the task reasoning LLM subtask prompt. In this example, the task is to reason about a door and a chair. Without including the bounding box colors in the prompt (blue), the VLM focuses on the wrong object, reasoning about the door behind the chair. When the color information is included (green), the VLM is guided to attend to the relevant objects.

append an additional prompt to guide the VLM focus on the correct objects in the image. Specifically, this hand-crafted prompt is: *Focus only on the objects that have a X and Y bounding box*, where *X* and *Y* are the colors of the bounding boxes, inferred from the objects' labels s_{L_2}. For this to work, each label in the set has a unique color associated with it. As shown in Fig. 4, providing this additional information helps the VLM attend to the objects of interest. In contrast, if the prompt only specifies generic bounding-boxed objects, the VLM may struggle to focus on the intended objects.

Finally, a second LLM interprets the VLM's output to decide whether the subtasks should be executed. This LLM is guided by a dedicated system prompt (see Fig. 3).

V. EXPERIMENTS

In this section, we first present the implementation details of our method (Section V-A). We then evaluate REASONINGGRAPH's ability to construct scene graphs enriched with object-level open-vocabulary features (Section V-B). Next, we assess the task reasoning module in diverse real-world scenarios, using a quadruped robot and data collected by a human operator (Section V-C). Finally, we report the runtime performance of our method (Section V-D).

A. Implementation Details

We employ YOLOe [22] as our object detector, which outputs both bounding boxes and segmentation masks. For open-vocabulary feature extraction, we use CLIP [7], specifically OpenAI's ViT-L/14 model, chosen for its strong generalization across image classification tasks. We select Deepseek VL2 [23] as our VLM due to its state-of-the-art (SOTA) performance. In addition, we use OpenAI's o3 LLM for task reasoning and parsing, and GPT-4o as subtask decisor LLM.

Our sensor and compute setup consists of a Realsense D455 RGB-D camera, an Ouster OS0 LiDAR, and a VectorNav VN100 IMU for odometry, integrated with an NVIDIA Jetson Orin AGX for onboard processing (see Fig. 1). On the Orin AGX, we run the feature extraction modules (YOLOe, CLIP, and the VL2 visual encoder) as well as scene graph construction online at 1 Hz, while the room detection, mesh refinement and pose optimization are run at 2 Hz. The LLMs and the language components of VL2 are hosted in the cloud and queried only when the task is given.

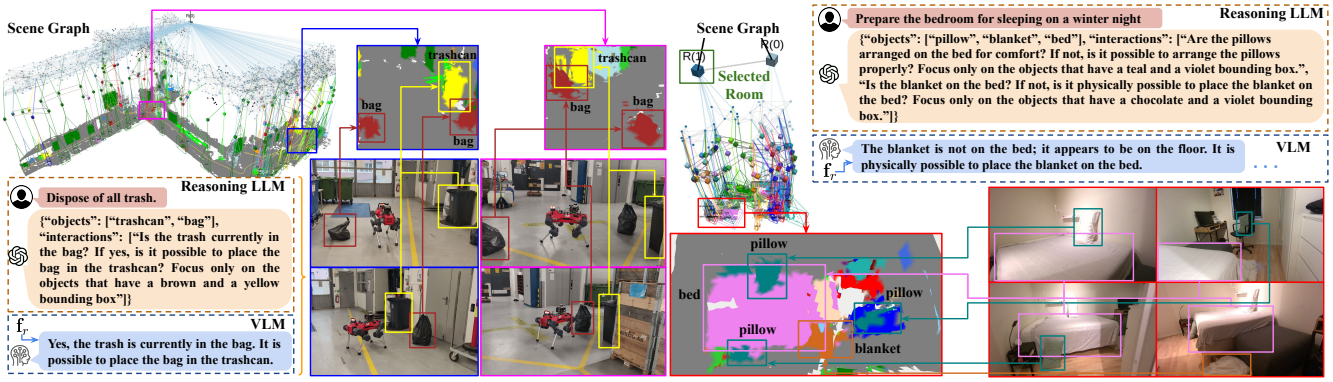
B. Open-Vocabulary Object Retrieval

To evaluate open-vocabulary object features, we compute the cosine similarity between each object feature and the dataset labels. We then rank the labels by similarity and measure the accuracy at the top-k ranked predictions (Acc_k). Following HOV-SG [16], we also compute the area under the top-k accuracy curve (AUC_k^{Acc}), which captures the alignment between predicted and ground-truth object categories across different values of k. We compare our approach against two strong open-vocabulary scene graph baselines, HOV-SG [16] and ConceptGraphs [15], on the Replica [26] and Habitat Semantics (HM3DSem) [25] datasets.

Both ConceptGraphs [15] and HOV-SG [16] construct scene graphs from fully open-vocabulary object segments, obtained by clustering CLIP embeddings on point clouds, which introduces background objects in their graphs. To ensure a fair comparison, we filter these out, as our method only includes foreground objects. Background objects generally degrade performance because their embeddings are more ambiguous due to a lack of texture and contamination from foreground objects. Specifically, we compute the cosine similarity between each detected object and a set of background categories (*e.g.,* "wall", "floor", "ceiling", "stairs"), and discard objects with high similarity. Results in Table I report top-k accuracies (Acc_k) and AUC_k^{Acc} scores for both our method and the baselines. REASONINGGRAPH achieves substantially higher performance, showing its ability to assign correct open-vocabulary features to objects. Note that baselines detect a larger number of objects, since their fully open-vocabulary design, which detects objects by clustering on the embedding space, can fragment a single object into multiple segments when CLIP embeddings vary across viewpoints of the same object. We nevertheless include these methods as baselines because they represent the current SOTA in open-vocabulary scene graph construction.

C. Real-World Task Reasoning Evaluation

To assess the reasoning capabilities of REASONINGGRAPH in handling complex tasks, we design a series of eval-



a) Quadruped performing **T1**. b) Operator performing **T3**.

Fig. 5: REASONINGGRAPH performing **T1** and **T3**. The scene graph is built during exploration, after which the LLM reasons about the task and the VLM evaluates subtasks using object relations ($\mathbf{f}_r$). In both individual experiments, a 100 SR% is achieved. In **T3**, we apply our room search method, achieving 100% accuracy in all 5 evaluations. We present one VLM reasoning example per task, although in practice the VLM reasons about each subtask.

uation tasks that require identifying objects within the graph and, in some cases, reasoning about their relations. Since no dedicated evaluation dataset exists, we choose to conduct real experiments, which not only provide the necessary evaluation data but also allow us to verify the robustness of our method in practice. Performance is measured using two metrics: the success ratio (SR%), defined as the percentage of correctly evaluated subtasks or successfully identified objects, and the number of false positives (FP), defined as the number of incorrectly evaluated subtasks or misidentified objects. For each task, we manually identify the ground truth objects or object pairs whose subtask should be positively evaluated. Specifically, we consider the following tasks:

- **T1: Dispose of all trash**: Identify filled trash bags near trash cans and determine if they can be thrown away.
- **T2: Ensure exits and entrances are not blocked**: Detect objects that may be blocking doorways.
- **T3: Prepare the bedroom for sleeping on a winter night**: Verify whether the pillows and blankets are placed appropriately on the bed.
- **T4: Prepare the meeting room for floor cleaning**: Determine whether chairs and other objects can be placed on desks to facilitate cleaning.
- **T5: Find a backpack, a fan, plants and trash cans.**

For **T1**, **T2**, and **T5**, REASONINGGRAPH is deployed on a quadruped robot equipped with the sensing and computing module introduced in Section V-A. The robot autonomously explores the environment using a graph-based path planning approach [27], while incrementally constructing the scene graph. The same planner [27] is used to navigate to detected objects or object pairs of positively evaluated subtasks. In contrast, for **T3** and **T4**, data is collected with the same sensing module, but carried by a human operator.

Once the scene graph is constructed, task reasoning begins. As detailed in Section IV-C, the task reasoning LLM identifies task-relevant objects, which are then located in the scene graph using CLIP-based cosine similarity. In **T5**, illustrated in Fig. 6, this retrieval step is sufficient, as no reasoning about object interactions is required. For the remaining tasks, the VLM is prompted with the object relationships ($\mathbf{f}_r$) and the subtask prompts generated by the task reasoning LLM.

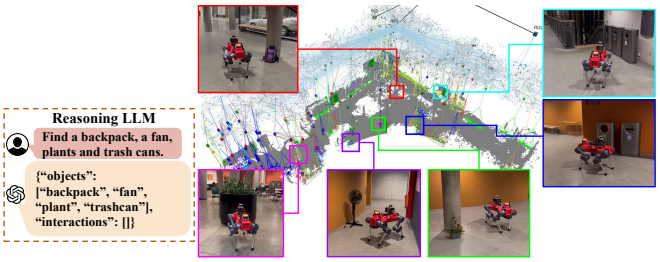


Fig. 6: Object search task (**T5**). After the scene graph is constructed during autonomous exploration, we ask REASONINGGRAPH to find a backpack, a fan, plants and trash cans. Our method is capable of finding all the objects.

Table II presents the success ratio and number of false positives with standard deviation across five evaluations for each task, comparing two VLMs: DeepSeek VL2 [23] and InstructBLIP [20]. REASONINGGRAPH (with VL2) achieves consistently high SR% (84-100) with low FP across all tasks. DeepSeek VL2 consistently outperforms InstructBLIP, achieving higher success ratios and fewer false positives, supporting its selection as the primary VLM in REASONINGGRAPH. #GT+ denotes the number of positive subtasks in each task. Fig. 5a illustrates **T1** on the quadruped robot, where the system reasons about placing four trash bags inside nearby trashcans, and Fig. 5b illustrates **T3**, involving arranging blankets and pillows on a bed. REASONINGGRAPH achieves high SR% in both examples.

To further assess the performance of the object-level relationships ($\mathbf{f}_r$), the VLM (VL2), and the subtask decisor LLM, we use the collected data from tasks **T1** to **T4** to re-evaluate all subtasks 100 times, accounting for the stochasticity of both models. As shown in Table III, the combined system achieves consistently high accuracy, defined as the percentage of correctly evaluated subtasks, across all tasks. The macro-averaged F_1 scores, which balance precision and recall, indicate strong performance on both positive and negative subtasks, *i.e.,* those that should and should not be executed. The pooled results highlight the robustness of our method, confirming its ability to reason about object interactions despite the stochasticity of the VLM and LLM.

D. Runtime Evaluation

To verify the feasibility of running our graph construction online, we measure the runtime of its main components,

	VLM	T1	T2	T3	T4	T5
SR%	VL2 [23]	90 ± 12.3	84 ± 15.0	90 ± 12.3	86 ± 7.1	100
	BLIP [20]	55 ± 24.5	48 ± 9.8	40 ± 12.2	53.3 ± 12.5	
FP	VL2 [23]	0.2 ± 0.4	0.8 ± 0.4	0.6 ± 0.5	0.8 ± 0.4	1.8 ± 0.4
	BLIP [20]	0.2 ± 0.4	1.8 ± 0.4	0.6 ± 0.49	1.0 ± 0.0	
	#GT+	4	5	4	6	7

TABLE II: Comparison of task reasoning performance using two VLMs (DeepSeek VL2 [23] and InstructBLIP [20]). **T1** - **T4** require relational reasoning, while **T5** does not. Each task is executed 5 times, and the results are averaged. Our method with VL2 achieves higher success ratios with fewer false positives, motivating its choice as the primary VLM.

Tasks	T1	T2	T3	T4	Pooled
Accuracy (%)	88.81	84.43	83.11	74.91	79.30
F_1 (%)	88.55	85.43	81.86	75.84	79.11
Average FP	0.27	1.04	0.69	1.12	0.78
# Positive subtasks	18	21	20	44	103
# Negative subtasks	3	75	7	11	96

TABLE III: VLM and subtask decisor LLM performance. We evaluate each subtask from **T1** - **T4** 100 times to assess the performance of object-level relationships and our reasoning module. This accounts for the stochasticity introduced by both the VLM and the decisor LLM.

as reported in Table IV. On average, YOLOe and CLIP inference require less than 160 ms per frame, while the VL2 visual encoder is the most expensive component, ranging from 214 - 278 ms. Scene graph construction remains lightweight, with runtimes between 145 - 276 ms. Overall, the method operates below one second per frame. This runtime supports online operation on the Orin AGX mounted on the quadruped robot, since its speeds allow running the method at 1 - 2 Hz without skipping important information.

VI. CONCLUSIONS

We propose REASONINGGRAPH, a framework for constructing hierarchical 3D scene graphs that incorporate open-vocabulary features and encode object-level relationships using a VLM. Our reasoning module further leverages both LLMs and a VLM to interpret complex tasks, identify task-relevant objects, and reason about their potential interactions. Extensive experiments show that REASONINGGRAPH outperforms existing open-vocabulary scene graph baselines in object retrieval and consistently solves diverse reasoning tasks with high success rates and low false positives. Deployment on a quadruped robot demonstrates the framework's capability for online, incremental scene understanding and task reasoning. These results demonstrate the potential of combining hierarchical representations with vision-language reasoning to achieve richer context-aware understanding.

Tasks	YOLOe [22]	CLIP [7]	VL2 Visual [23]	Graph
T1	44.9 ± 29.5	76.1 ± 56.6	214.8 ± 147	262.2 ± 105.5
T2	54.9 ± 43	97.4 ± 59	257.8 ± 132.4	231.1 ± 86.3
T3	70.7 ± 63.6	133.1 ± 95.5	277.8 ± 141.2	144.9 ± 91.8
T4	87.9 ± 33.8	157.9 ± 69.1	223.8 ± 145.3	213.5 ± 84.2
T5	54.8 ± 57.2	82.9 ± 66.9	247 ± 140.8	276 ± 92.2

TABLE IV: Timing statistics (mean ± standard deviation) in milliseconds.

REFERENCES

[1] J. L. Schönberger, E. Zheng *et al.*, "Pixelwise view selection for unstructured multi-view stereo," *European Conference on Computer Vision*, 2016.

[2] N. Khedekar, M. Kulkarni *et al.*, "Mimosa: A multi-modal slam framework for resilient autonomy against sensor degradation," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022.

[3] M. Nissov, N. Khedekar *et al.*, "Degradation resilient lidar-radar-inertial odometry," *IEEE International Conference on Robotics and Automation*, 2024.

[4] C. Forster, L. Carlone *et al.*, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, 2017.

[5] A. Rosinol, A. Violette *et al.*, "Kimera: From slam to spatial perception with 3d dynamic scene graphs," *The International Journal of Robotics Research*, 2021.

[6] S. Peng, K. Genova *et al.*, "Openscene: 3d scene understanding with open vocabularies," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[7] A. Radford, J. Kim *et al.*, "Learning transferable visual models from natural language supervision," *International Conference on Machine Learning*, 2021.

[8] A. Kirillov, E. Mintun *et al.*, "Segment anything," *IEEE/CVF International Conference on Computer Vision*, 2023.

[9] I. Armeni, Z.-Y. He *et al.*, "3d scene graph: A structure for unified semantics, 3d space, and camera," *IEEE/CVF International Conference on Computer Vision*, 2019.

[10] N. Hughes, Y. Chang *et al.*, "Hydra: A real-time spatial perception system for 3D scene graph construction and optimization," *Robotics: Science and Systems*, 2022.

[11] ——, "Foundations of spatial perception for robotics: Hierarchical representations and real-time systems," *The International Journal of Robotics Research*, 2024.

[12] J. Wald, H. Dhamo *et al.*, "Learning 3d semantic scene graphs from 3d indoor reconstructions," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[13] S.-C. Wu, J. Wald *et al.*, "SceneGraphFusion: Incremental 3D Scene Graph Prediction from RGB-D Sequences," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[14] S. Koch, N. Vaskevicius *et al.*, "Open3dsg: Open-vocabulary 3d scene graphs from point clouds with queryable objects and open-set relationships," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[15] Q. Gu, A. Kuwajerwala *et al.*, "Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning," *IEEE International Conference on Robotics and Automation*, 2024.

[16] A. Werby, C. Huang *et al.*, "Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation," *Robotics: Science and Systems*, 2024.

[17] Z. Zhang, H. Cai *et al.*, "Efficientvit-sam: Accelerated segment anything model without performance loss," *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2024.

[18] F. Li, H. Zhang *et al.*, " Mask DINO: Towards A Unified Transformer-based Framework for Object Detection and Segmentation ," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[19] G. Narita, T. Seno *et al.*, "Panopticfusion: Online volumetric semantic mapping at the level of stuff and things," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[20] W. Dai, J. Li *et al.*, "InstructBLIP: Towards general-purpose vision-language models with instruction tuning," *Conference on Neural Information Processing Systems*, 2023.

[21] L. Chen, X. Wang *et al.*, "Clip-driven open-vocabulary 3d scene graph generation via cross-modality contrastive learning," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[22] A. Wang, L. Liu *et al.*, "Yoloe: Real-time seeing anything," *IEEE/CVF International Conference on Computer Vision*, 2025.

[23] H. Lu, W. Liu *et al.*, "Towards real-world vision-language understanding," 2024.

[24] H. Oleynikova, Z. Taylor *et al.*, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

[25] K. Yadav, R. Ramrakhya *et al.*, "Habitat-matterport 3d semantics dataset," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[26] J. Straub, T. Whelan *et al.*, "The Replica dataset: A digital replica of indoor spaces," 2019.

[27] T. Dang, M. Tranzatto *et al.*, "Graph-based subterranean exploration path planning using aerial and legged robots," *Journal of Field Robotics*, 2020.