GraphEQA: Using 3D Semantic Scene Graphs for Real-time Embodied Question Answering

Saumya Saxena[1*✉], Blake Buchanan[2*], Chris Paxton[3], Bingqing Chen[4], Narunas Vaskevicius[4], Luigi Palmieri[4], Jonathan Francis[1,4], Oliver Kroemer[1]

[1]Carnegie Mellon University [2]Neya Systems [3]Hello Robot Inc. [4]Bosch Center for AI

{saumyas,okroemer}@andrew.cmu.edu, bbuchanan@neyarobotics.com, cpaxton@hello-robot.com, {bingqing.chen,jon.francis}@us.bosch.com, {narunas.vaskevicius,luigi.palmieri}@de.bosch.com [†]

Abstract

In Embodied Question Answering (EQA), agents must explore and develop a semantic understanding of an unseen environment in order to answer a situated question with confidence. This remains a challenging problem in robotics, due to the difficulties in obtaining useful semantic representations, updating these representations online, and leveraging prior world knowledge for efficient exploration and planning. Aiming to address these limitations, we propose GraphEQA, a novel approach that utilizes real-time 3D metric-semantic scene graphs (3DSGs) and task relevant images as multi-modal memory for grounding Vision-Language Models (VLMs) to perform EQA tasks in unseen environments. We employ a hierarchical planning approach that exploits the hierarchical nature of 3DSGs for structured planning and semantic-guided exploration. Through experiments in simulation on the HM-EQA dataset and in the real world in home and office environments, we demonstrate that our method outperforms key baselines by completing EQA tasks with higher success rates and fewer planning steps. Videos and code are available on our website.

1. Introduction

Embodied Question Answering (EQA) [9] is a challenging task in robotics, wherein an agent is required to explore and understand a previously unseen environment sufficiently well, in order to answer an embodied question in natural language. Accomplishing this task efficiently requires agents to rely on both commonsense knowledge of human environments as well as ground its exploration strategy in the current environment context. For example, to answer the question "How many chairs are there at the dining table?", the agent might rely on commonsense knowledge to understand that dining tables are often associated with dining rooms and dining rooms are usually near the kitchen towards the back of a home. A reasonable navigation strategy would involve navigating to the back of the house to locate a kitchen. To ground this search in the current environment, however, requires the agent to continually maintain an understanding of where it is, memory of where it has been, and what further exploratory actions will lead it to relevant regions. Finally, the agent needs to observe the target object(s) and perform visual grounding, in order to reason about the number of chairs around the dining table, and confidently answer the question correctly.

Maintaining a concise and effective memory and using it to take actions in the environment is critical for solving EQA tasks. Prior works have demonstrated the impressive commonsense reasoning capabilities of Vision Language Models (VLMs) as planning agents, while leveraging a semantic map for retrieval [13] or semantic exploration [36]. In such approaches, the VLMs are not grounded in the current environment and the tasks of commonsense reasoning and context-based decision-making are disconnected. Recent works [3, 35, 41, 50, 52] focus on maintaining memory modules that can be queried by VLM agents for grounded planning. To construct a semantically rich memory, prior works either maintain a large set of images (not compact) [27, 52] or have to perform an expensive offline processing step to obtain a compact representation [3, 50, 51]. Thus, such semantic memory modules are either semantically rich [13, 46, 50], compact [35, 41], *or* online [41]—but not all at the same time.

To address these limitations, we propose GraphEQA, an

*Equal contribution. ✉Correspondence.

†This work was in part supported by the National Science Foundation under Grant No. CMMI-1925130 and in part by the EU Horizon 2020 research and innovation program under grant agreement No. 101017274 (DARKO). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

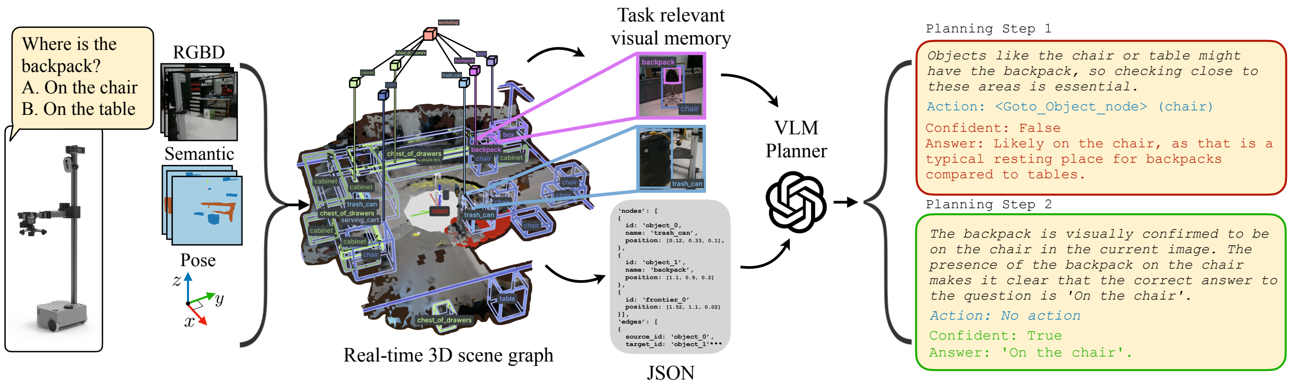


Figure 1. Overview of GraphEQA: A novel approach for utilizing real-time 3D metric-semantic hierarchical scene graphs and task-relevant images as multimodal memory for grounding vision-language based planners to solve embodied question answering tasks in unseen environments.

approach for building an online, compact, multimodal semantic memory that combines global, semantically-sparse, and task-agnostic information from real-time 3D scene graphs [18] with the local and semantically-rich information from task-relevant images [51]. GraphEQA uses this multimodal representation for grounding vision-language planners to solve EQA tasks in unseen environments. Specifically, we utilize a recent spatial perception system [18] that incrementally creates a real-time 3D metric-semantic hierarchical scene graph (3DSG), given sequential egocentric image frames. We further augment this scene graph with semantic room labels and semantically-enriched frontiers. We also maintain a task-relevant visual memory that keeps track of task-relevant images as the robot explores the environment. Finally, we employ a hierarchical planning approach that utilizes the hierarchical nature of scene graphs and semantically relevant frontiers for structured planning and exploration in an unseen environment before using the multimodal memory to answer the embodied question with high confidence.

We demonstrate that given our real-time multimodal memory and hierarchical planning approach, the agent is able to accomplish long-horizon tasks with significantly fewer VLM planning steps, explores explainable task-relevant frontiers, and succeeds in EQA tasks with a higher rate than previous approaches. We demonstrate our results on the HM-EQA dataset [36] in the Habitat simulation environment [53] and also in the real world using the Hello Robot Stretch mobile manipulator in two indoor scenes.

The key contributions of this work are as follows:

- We present GraphEQA, a novel approach for using real-time 3D metric-semantic hierarchical scene graphs and task-relevant images as multimodal memory, for grounding VLMs to solve EQA tasks in unseen environments.
- We propose an approach to enrich 3DSGs with 1) semantically enriched frontiers and 2) semantic room labels.
- We propose a hierarchical VLM-based planning approach that exploits the hierarchical nature of the enriched 3DSG for structured exploration and planning.
- We provide extensive experimental results, both in simulation on the HM-EQA dataset [36] and in the real-world in two indoor environments, home and office, using the Hello Robot Stretch RE2 mobile manipulator.

2. Related Work

3D semantic scene graph representations for planning: 3D semantic scene graphs [4, 12, 22, 37, 45] have recently emerged as a compact, efficient, and semantically rich representation for indoor environments, leading to recent developments in both offline [6, 13, 24, 46] and online scene graph generation methods [18, 26, 49]. Offline methods [13, 46] focus on building open-vocabulary semantic scene graphs, by semantically enriching the nodes and edges of a scene graph (affordances and relationships) using open-vocabulary vision-language models [23, 32], and utilizing these scene graphs for retrieval-based downstream language-guided tasks.

Online methods [18, 26] are critical for real-time deployment of embodied agents in unseen environments for tasks such as mobile manipulation or embodied question answering (EQA). However, these methods rely on closed set semantics [18] or are restricted to a predefined set of tasks [26] limiting their applicability to tasks that do not require complex open-world reasoning. Our proposed approach finds a middle ground between these offline and online approaches, by maintaining a *multimodal* memory, wherein a 3D scene graph is constructed *online* with closed-set semantics [18], and is used by a VLM agent to actively guide the agent towards task relevant areas where it can capture and store semantically-rich images [51]. This multimodal memory can then be used to solve open-world EQA tasks.

Recent works have explored the use of textual representations of 3D scene graphs for grounding VLM-based planners [1, 14, 33, 35]. These methods use 3DSGs for object search or rearrangement tasks that can be completed by relying solely on well-constructed scene graphs, but do not consider EQA tasks that require more complex semantic scene understanding.

Vision Language Models for 3D Scene Understanding and Planning: Several previous works leverage the commonsense reasoning capabilities of foundation models for long-horizon planning [2, 16, 17]. However, these methods are not grounded in the context of the current environment and additional tools are required to translate the LLM plan to executable actions [8, 16, 25, 42]. Previous works have explored the use of pre-trained vision language models for building dense queryable open-vocabulary 3D semantic representations using explicit pixel-level [10, 15, 19, 30, 56] or implicit neural [21, 38, 44] representations. These maps are built offline and then used for downstream retrieval-based tasks. However, such dense representations are not scalable to large environments and cannot be used to ground VLM-based planners.

Recent advancements in grounding VLM-based planners using videos [52, 57] are promising, but struggle with scalability for long-horizon tasks in large environments. VLMs have been used as planners while leveraging semantic maps for retrieval [7, 13, 46] or semantic exploration [28, 36, 39, 54, 58], however such approaches disconnect context-based decision-making and commonsense reasoning. Offline methods that build topological maps [40], keyframe selections [51], 3D semantic graphs [6, 13, 24, 31, 46] and experience summaries [3, 5, 48, 50], are unsuitable for real-time deployment in novel settings. Online semantic scene graphs, while real-time, are limited by closed-set semantics. Our approach introduces an online, compact, and semantically rich multimodal memory to effectively ground VLM planners for EQA tasks.

Embodied Question Answering: Embodied Question Answering [9, 11, 47] has emerged as a challenging paradigm for testing robotic task planning systems on their ability to incrementally build a semantic understanding of an environment in order to correctly answer an embodied question with confidence. [36] builds an explicit task-specific 2D semantic map of the environment to guide exploration, [3, 50] build offline experience modules that the LLM can query, [27] uses video memory to answer embodied questions using long-context VLMs. We focus on building agents that do not disconnect the semantic memory from the planner by grounding the planner in a compact scene representation for solving EQA tasks online.

3. Preliminaries

3.1. Hierarchical 3D Metric-semantic Scene Graphs

3D metric-semantic scene graphs (3DSGs) provide a structured, layered representation of environments, encoding spatial, semantic, and relational information [4, 22, 37]. Recent works, Clio [26] and Hydra [18], introduce an efficient real-time framework for incremental construction of metric and semantic SG layers encoding low-level geometries as well as high-level semantics at multiple levels of abstraction including objects, regions, rooms, buildings, etc.

Hydra 3DSGs are comprised of the following layers: *Layer 1 (bottom)*: a metric-semantic 3D mesh, *Layer 2*: objects and agent, *Layer 3*: regions or places, *Layer 4*: rooms, and *Layer 5 (top)*: building. Intra-layer edges between nodes denote ’traversability’, while inter-layer edges denote ’belonging’. For example, an edge between regions in Layer 3 denotes traversability between these regions and an edge between an object and a room denotes that the object is located in that room. 3DSGs are constructed using RGB and depth images from the robot’s camera, camera pose and camera intrinsics. Using off-the-shelf image segmentation models [59], the object nodes are assigned semantic object labels.

3.2. 2D Occupancy Mapping and Frontier Detection

3D voxel-based occupancy maps are a simple and effective way for storing explored, occupied and unexplored regions of an environment for planning and navigation. As the robot explores, using depth data and camera intrinsics, occupancy of the voxels is updated using Volumetric Truncated Signed Distance Function (TSDF) fusion. TSDF integrates depth observations to update voxels as occupied or free, while areas beyond a certain threshold are marked unexplored. Typically, the 3D occupancy map is projected into 2D, where frontiersboundaries between explored and unexplored regionsare identified to guide further exploration. We employ this approach in our method for identifying frontiers, clustering them and adding them to the scene graph.

3.3. HM-EQA Dataset

The Habitat-Matterport Embodied Question Answering (HM-EQA) dataset introduced by Yadev et al. [53] is based in the Habitat-Matterport 3D Research Dataset (HM3D) of photo-realistic, diverse indoor 3D scans [34]. The dataset is composed of 500 multiple choice questions from 267 different scenes which fall in the following categories: identification, counting, existence, state, and location. We use this dataset to benchmark our results against baselines in simulation. An example question from the HM-EQA dataset is shown below. This particular example is of type “location”.

Question: *I need to find light blue pillow. Where is it currently located?*
A) In the bedroom B) In the bathroom
C) In the hallway D) In the home office space
Answer: **A**

4. Proposed Approach

An overview of our proposed approach is shown in Fig. 2. We consider the task of embodied question answering, wherein an embodied agent is required to explore an indoor environment until it can answer a multiple choice question with high confidence. As the agent explores the environment, it incrementally constructs a 3D scene graph using Hydra [18] (Sec. 3.1) as well as a 2D occupancy map for frontier selection (Sec. 3.2) in real time. This scene graph is enriched with semantic information to facilitate hierarchical planning and semantic-guided frontier exploration, details of which are provided in Sec. 4.1. While navigating in the environment, the agent also maintains in memory a small set of task-relevant images. Details on how this visual memory is constructed are provided in Sec. 4.2. Finally, a hierarchical VLM-based planner utilizes the enriched scene graph and the task-relevant visual memory, to explore an unseen environment until it can answer an embodied question with high confidence. At every planning step, the planner outputs a high level action, which is executed in the environment while the scene graph, visual memory, and frontiers are all updated in real time. The planner architecture is explained in detail in Sec. 4.3.

4.1. Scene Graph Construction and Enrichment

We use Hydra [18] to construct a layered metric-semantic scene graph as mentioned in Sec. 3.1. We also maintain a 2D occupancy map of the environment depicting the explored, occupied, and unexplored navigable regions of the environment as mentioned in Sec. 3.2. We perform room and frontier enrichment steps to enable semantic-guided exploration and hierarchical planning.

Room enrichment: Room nodes in Hydra's 3DSG are assigned generic labels such as R0, R1, etc. To enrich them with semantic information, we prompt an LLM to assign semantic labels to each of the room nodes. We use a simple prompt "Which room are these objects <object list> most likely located in?" where <object list> is the list of all objects located in a certain room in scene graph. The output of the LLM is used to update the room labels.

Frontier enrichment: Frontiers are boundaries between explored and unexplored regions in a 2D occupancy map, and are useful for guiding greedy exploration in unseen environments. However, they do not encode semantic information on whether exploring a certain frontier will provide useful information for answering an embodied question. To enrich our 3DSG with semantic information that can enable task-relevant exploration, we extract frontier nodes from the 2D occupancy map, cluster them, and add them as independent nodes to the scene graph. Next, we find top-k object nodes nearest to each clustered frontier node (within a maximum distance). We add edges to the scene graph connecting each frontier node to its top-k object neighbors. This semantic information can now be utilized by a VLM-based planner to select the most semantically-relevant frontier to explore next. For example, when searching for a kettle, it might be useful to choose a frontier node which is near to a fridge, kitchen counter, or stove. For general exploration, it could be useful to choose frontier nodes near doors. We use $k = 3$ and maximum distance $= 2m$ in our experiments but can be varied based on the environment.

4.2. Task-relevant Visual Memory

At every planning step, the agent needs to execute a high level action in the environment (more details in Sec. 4.3). During action execution, images are stored in a buffer at a specified sampling frequency to avoid multiple similar repeated images. Images from this buffer, along with keywords from the question/task, are then processed using SigLIP [55] to obtain the text-image relevancy score for each image. Using this score, only the top-k most relevant images are maintained in the buffer and the rest are discarded. We use $k = 2$ in this work. We append these top-k images together, along with the current view of the agent, and use it as the visual input to the VLM planner at the next planning step as shown in Fig. 2

4.3. Hierarchical Vision-Language Planner

Our planner architecture is shown in Fig. 3. We use GPT-4o [29] for our VLM-based planner. Below we describe in detail its key components:

Inputs: At every planning step t, the VLM planner takes as input a multiple choice question $q \in Q$, the choices $a \in \mathcal{A}$, the enriched scene graph $\mathcal{S}_t^e$, and the task-relevant visual memory $\{\mathcal{I}_k\}_{k=1}^K$, where K is the number of images in memory. Additionally, we provide the planner with a structured history $\mathcal{H}_t$ and the agent's current state X_t. X_t is defined in the following format: `"The agent is currently at node <agent_node_id> at position <agent_position> in room <room_id> <room_name>"`, where information in '<·>' is populated from $\mathcal{S}_t^e$.

Outputs: Given the inputs, the planner is required to output an answer $a_t \in \mathcal{A}$ to the multiple choice question, a boolean value $c_t \in \{\text{True}, \text{False}\}$ representing whether it is confident in answering the question, its current numeric confidence level $p_t^c \in [0, 1]$, and the next action u_t that the agent should take. We require the planner to also output the reasoning behind the choice of action

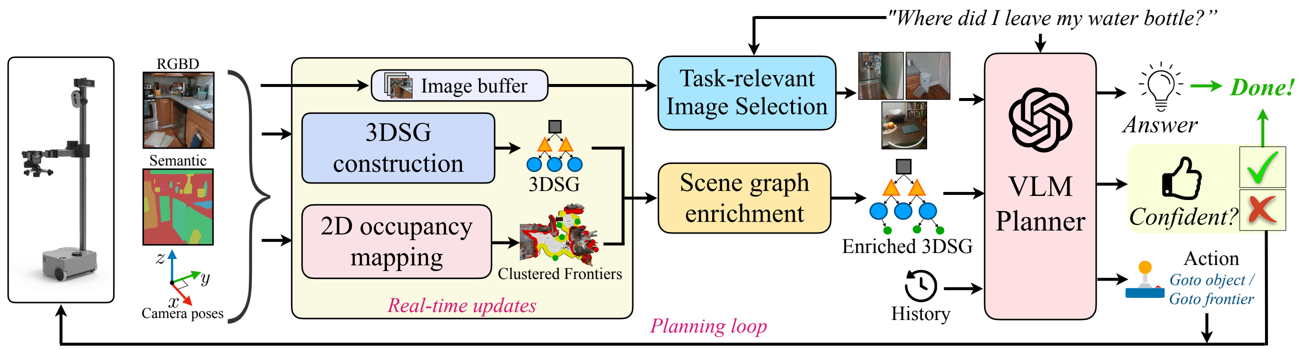


Figure 2. Overall GraphEQA architecture. As the agent explores the environment, it used its sensor data (RGBD images, semantic map, camera poses and intrinsics) to construct a 3D metric-semantic hierarchical scene graph (3DSG) as well as a 2D occupancy map for frontier selection in real time. The constructed 3DSG is enriched as discussed in Sec. 4.1. From the set of images collected during each trajectory execution, a task-relevant subset is selected, called the task-relevant visual memory (Sec. 4.2). A VLM-based planner (Sec. 4.3) takes as input the enriched scene graph, task-relevant visual memory, a history of past states and actions, and the embodied question and outputs the answer, its confidence in the selected answer, and the next step it needs to take in the environment. If the VLM agent is confident in its answer, the episode is terminated, else the proposed action is executed in the environment and the process repeats.

and its confidence in the answer. Finally, the planner is required to plan the next few steps, selecting from two high level action types: `<Goto_Object_node>(object_id)` and `<Goto_Frontier_node>(frontier_id)`, where `object_id` and `frontier_id` are selected from $\mathcal{S}_t^e$. Selecting an object node enables further visual examination of relevant visited areas. Selecting a frontier node enables visitation of unexplored areas. To prevent the planner from hallucinating non-existent nodes, we employ the structured output capabilities of GPT-4o. Finally, the planner is required to output a brief description of the current scene graph as well as a brief description of the input images. We track and update the history $\mathcal{H}_t$ at each time t such that $\mathcal{H}_{t+1} = X_t + a_t + c_t + p_t^c + u_t + \mathcal{H}_t$.

Hierarchical planner and frontier exploration: For actions of the type `<Goto_Object_node>(object_id)`, we enforce a hierarchical planning structure by requiring the planner to first reason about which room to go to by selecting a room node, then a region node (within the selected room) and finally the object node to go to. This planning structure reflects the hierarchical structure of the 3D scene graph, and forces the planner to reason about the hierarchical semantics of the scene to explore and answer the embodied questions. For actions of the type `<Goto_Frontier_node>(frontier_id)`, we require the planner to provide the reasoning behind the choice of frontier node by referring to the object nodes connected to the selected frontier by edges in the scene graph. This enforces semantic reasoning in the frontier selection process such that frontiers chosen are task-relevant and are information gathering, e.g., near doors.

Termination condition: A planning episode is terminated when the planner outputs c_t = True or $p_t^c > 0.9$, i.e., when it is confident in answering the question. The episode is also terminated if $t > T_{max}$, when the maximum allowed planning steps have been reached.

Prompt description: Here we share some key aspects of our prompting scheme. We provide the planner with a system-level prompt detailing how to understand the scene graph structure. We explain the criteria behind choosing the actions: hierarchically for object nodes and task-relevant or information gathering for frontier nodes. We explain that the scene graph can be imperfect/incomplete and that the planner should always seek visual confirmation before answering the question with confidence while employing the scene graph as a semantic map for examining and exploring the scene. Finally, we prompt the VLM to provide a brief description of the scene graph and the input images, focusing on elements in the scene that are relevant to the current task. The complete prompt is available in the appendix 8.2.

5. Experimental Setup

We identify the key research questions this work aims to evaluate:

- **Q1**: Do hierarchical 3D scene graphs provide an effective **metric-semantic memory** for solving **embodied question answering** tasks?
- **Q2**: How does the **real-time** nature of GraphEQA compare to offline approaches that provide the planner with full-state scene graphs? Specifically, we aim to evaluate if GraphEQA can utilize incrementally constructed state information to solve EQA tasks and terminate based on **confidence**, without needing to acquire full state information.
- **Q3**: Can GraphEQA effectively combine and reason about the **high-level, semantically-sparse and task-agnostic** information offered by **scene graphs** and the **local, semantically-rich and task-relevant** from the **visual memory** to actively take information gathering actions until it can confidently answer an embodied question?

5.1. Baselines and Ablations

To evaluate our method and answer the above research questions, we compare our method against several baselines and focus on

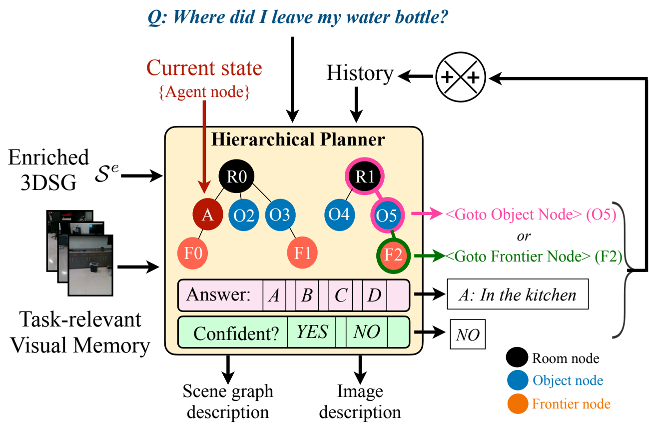


Figure 3. VLM Planner Architecture. The Hierarchical Vision-Language planner takes as input the question, enriched scene graph, task-relevant visual memory, current state of the robot (position and room) and a history of past states, actions, answers and confidence values. The planner chooses the next <Goto_Object_node> action hierarchically by first selecting the room node and then the object node. The <Goto_Frontier_node> action is chosen based on the object nodes connected to the frontier via edges in the scene graph. The planner is asked to output a brief reasoning behind choosing each action. The planner also outputs an answer, confidence in its answer, reasoning behind the answer and confidence, the next action, a brief description of the scene graph and the visual memory.

methods that employ VLM-based planners for solving **EQA** or object goal navigation tasks. We compare against a strong baseline, **Explore-EQA** [36], which calibrates Prismatic-VLM [20] to answer embodied questions confidently while maintaining a 2D semantic memory and using prompted images to guide exploration. Furthermore, to enable a more direct comparison to GraphEQA, we also compare against a version of Explore-EQA which uses GPT-4o as the VLM, a more powerful VLM as compared to Prismatic-VLM. We call this baseline **Explore-EQA-GPT4o**. We use the above baselines to answer **Q1** and **Q3**.

We also compare GraphEQA against a modified version of **SayPlan** [35] which we call **SayPlanVision**. Similar to SayPlan, SayPlanVision first constructs a scene graph of the whole scene offline and then uses this scene graph for planning. Similar to GraphEQA, SayPlanVision also has access to the task-relevant visual memory at every planning step. This baseline has the advantage of having access to the whole scene graph at every planning step, however has the disadvantage of requiring an expensive offline scene graph generation step. We use this baseline to evaluate the effectiveness of our real-time approach and answer **Q2**.

To further evaluate our method for **Q3**, we consider two ablations: **GraphEQA-SG** and **GraphEQA-Vis**. In **GraphEQA-SG**, the planner only has access to the real-time 3DSG and does not have access to images. In **GraphEQA-Vis**, the planner only takes the visual memory as input and exploration is done via random frontier-based exploration. These baselines will help us evaluate the benefits of using multi-modality in GraphEQA. Finally, we perform some additional ablations to evaluate the utility of different components of our method: **GraphEQA-NoEnrich**, which does not use frontier enrichment as mentioned in Sec. 4.1; and **GraphEQA-CurrView**, which uses only the current view as input to the VLM and does not choose additional task-relevant keyframes as mentioned in Sec. 4.2.

Table 1. Comparison to baselines (Simulation): Success rate (%), number of planning steps and L_τ the trajectory length. Methods with a † indicate our implementations of that particular baseline.

Method	Succ. Rate (%)	#Planning steps	L_τ (m)
Explore-EQA [36]	51.7	18.7	38.1
Explore-EQA-GPT4o†	46.4	3.4	6.3
SayPlanVision†	54.8	2.6	5.3
GraphEQA	**63.5**	5.1	12.6

Table 2. Ablations (Simulation): Success rate (%), number of planning steps and L_τ the trajectory length.

Ablation	Succ. Rate (%)	#Planning steps	L_τ (m)
GraphEQA-SG	13.6	8.8	33.0
GraphEQA-Vis	45.7	1.0	3.9
GraphEQA-NoEnrich	59.5	5.1	11.1
GraphEQA-CurrView	53.1	5.7	12.2
GraphEQA	**63.5**	5.1	12.6

Metrics: To compare the performance of our method against the baselines and ablations discussed in Sec. 5.1, we use the following three metrics: 1) Success Rate (%): an episode is considered a success if the agent answers the embodied question correctly with high confidence. Success rate is defined as the number of successful episodes divided by the total number of episodes; 2) # Planning steps: For successful episodes, we record the total number of VLM planning steps; 3) Trajectory length (meters): For successful episodes, we record the total length of the path traveled by the robot. We also report the success rates across the different question types in the EQA dataset (see Sec. 5.2, below).

5.2. Experimental Domains

We evaluate GraphEQA and the baselines mentioned in Sec. 5.1, in simulation in the Habitat-Sim [43] environment on the HM3D-EQA dataset [36] and in the real-world in two different settings: home and office. For the real-world setup, we deploy and evaluate our proposed approach on the **Hello Robot Stretch Research Edition 2 (RE-2)** [1] mobile manipulation platform. All experiments are conducted on a desktop machine with two (2) NVIDIA TITAN RTX GPUs, 64GB of RAM, and an Intel i9-100900K CPU.

6. Experimental Results

6.1. Comparison to Baselines

Tab. 1 shows simulation results comparing **GraphEQA** to the baselines discussed in Sec. 5.1 on the HM-EQA dataset [36]. GraphEQA has higher success rate as compared to **Explore-EQA** and **Explore-EQA-GPT4o**. Compared to **Explore-EQA** our method completes the task in significantly lower planning steps and navigates the environment more efficiently (lower trajectory

[1] GraphEQA leverages the open-source codebase for Hello Robot's stretch_ai at `https://github.com/hello-robot/stretch_ai`

length). **Explore-EQA-GPT4o** has lower success rates and qualitative results show that it tends to be overconfident and terminates the episode early. GraphEQA outperforms **SayPlanVision** even though SayPlanVision has access to the complete scene graph. However, for successful episodes, the average planning steps in SayPlanVision are lower than in GraphEQA since the agent does not need to explore the environment to build the scene graph. We provide a more detailed discussion of these results aligned with our research questions (Sec. 5) in the following Sec. 6.2.

6.2. Discussion

Q1: We observe from Tab. 1 that **GraphEQA** has higher success rate as compared to **Explore-EQA** and **Explore-EQA-GPT4o**, without the need to build an explicit 2D semantic task-specific memory. This demonstrates the capability of 3DSGs to provide an effective metric-semantic memory for EQA tasks. We also observe that GraphEQA requires a significantly lower number of planning steps as compared to **Explore-EQA**. This is because, unlike **Explore-EQA**, GraphEQA does not entirely rely on images as input to the VLM planner for building the semantic memory as well as planning, constraining the planner to choose from only regions that are visible in the current image. GraphEQA, on the other hand, can use the hierarchical structure of the scene graph as well as semantically-enriched frontier nodes to plan across the entire explored space. We observe that **Explore-EQA-GPT4o** has lower success rate but also has significantly lower planning steps as compared to GraphEQA. Quantitatively, we observe that Explore-EQA-GPT4o is overconfident and terminates the episode early. We hypothesize that, since the VLM is uncalibrated, without access to a scene graph to ground the planner in the current environment, Explore-EQA-GPT4o tries to answer the question using commonsense reasoning and does not understand that it needs to explore more. Additional error analysis (Sec. 6.4) reveals that Explore-EQA-GPT4o has significantly high percentage of *false positives*, i.e., questions are answered successfully using commonsense reasoning/guessing, without grounding in the answer in the current scene. For more details, see Sec. 6.4. This provides additional evidence of the effectiveness of 3DSGs in enabling *semantic exploration* by grounding the planner in the current environment.

Qualitatively, we observe that actions chosen by the planner, <Goto_Object_node>(object_id) and <Goto_Frontier_node>(frontier_id), are task-specific and explainable. For more qualitative results please refer to App. 8.5.

Q2: We observe from Tab. 1 that GraphEQA performs better than **SayPlanVision** which has access to the complete scene graph. This is a surprising result since it is expected that, given full scene graph information, SayPlanVision would outperform GraphEQA across all metrics. However, from a qualitative analysis of the results for **SayPlanVision**, we observe that, given access to the complete scene graph, the agent is overconfident about its choice of object node actions, leading to shorter trajectory lengths in successful cases, but also to increased failure cases. This points to an interesting advantage of our real-time approach — *incrementally* building memory by exploring task-relevant regions and maintaining a more concise representation benefits EQA tasks.

Q3: We observe from Tab. 2 that in the absence of images, and using only the 3DSG as input to the VLM planner, the **GraphEQA-SG** ablation performs very poorly. This demonstrates that a semantic scene graph constructed using closed-set semantics and without any task-specific semantic enrichment, will provide an incomplete and insufficient understanding of the environment, whereas open-set semantics and task-specific enrichment are crucial for solving EQA tasks. Furthermore, we note that the performance of the vision-only ablation **GraphEQA-Vis** also suffers: this is because the agent takes random exploratory actions in the environment, with no semantic understanding of the overall scene. Additionally, without the presence of a scene graph to ground the agent in the current environment, the agent exhibits overconfidence (very few planning steps) and tries to answer the question solely based on the current image (similar behavior as observed in **Explore-EQA-GPT4o**). **GraphEQA** outperforms all ablations, providing clear evidence of the utility of a multimodal approach for combining global spatial and semantic information from 3D scene graphs with local but rich semantic information from images, for solving challenging EQA tasks.

Figure 4. An example of a false positive case. The image on the left is the image that can be used to answer the question correctly. The image on the right is the image used by an agent to 'guess' the answer correctly with high confidence without grounding the answer in the current environment.

6.3. Additional Ablations:

Here we analyze two additional ablations, **GraphEQA-NoEnrich** and **GraphEQA-CurrView**. We observe that GraphEQA-NoEnrich performs slightly worse than GraphEQA which demonstrates that enriching the scene graph with additional semantic information in the form of edges between frontiers and nearest objects, as discussed in Sec. 4.1, lends itself to semantically informed exploration. We observe that the performance drop is worse in the case of GraphEQA-CurrView, where we do not use task-relevant visual memory, but only the current view of the agent. This demonstrates that task-relevant visual memory is very useful in long-horizon tasks where the current view of the robot might not be the best view for answering an embodied question.

6.4. Error Analysis of Competing Baselines

Given the nature of the EQA tasks, it is possible that some of the questions are answered successfully using only commonsense reasoning/guessing, without grounding the answer in the current environment. We consider these cases as ***false positives***. An example of a false positive is shown in Fig. 4. Furthermore, we also notice ***false negatives***, where the answer was marked incorrect given

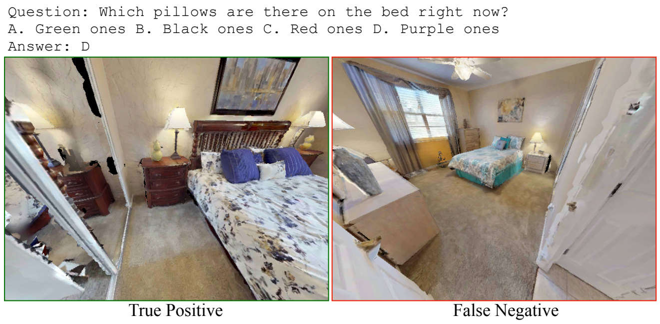


Figure 5. An example of a false negative case. The question inquires about the color of the pillow on the bed. The question is ambiguous. On the left is the image that corresponds to the answer in the dataset i.e. purple pillows. On the right is an image that the agent encounters during exploration and answers that the pillows are 'green' with high confidence. Given the image, the answer is correct but is deemed incorrect in the dataset.

Table 3. Error analysis (Simulation): Percentage %

	GraphEQA	Explore-EQA	Explore-EQA-GPT4o
True positive	58.18	31.58	22.81
True negative	31.82	44.74	46.49
False positive	6.36	16.67	24.56
False negative	3.64	7.02	6.14

the answer in the data set, although given the current image and scene graph, the answer seemed appropriate. Such cases exist due to ambiguities in the dataset. An example of a false negative is shown in Fig. 5. To get an estimate of the number of false positives and false negatives in our baselines, we uniformly sample a set of 114 questions from the HM-EQA dataset and manually label the results across the four categories: True Positives, True Negatives, False Positives and False Negatives. The results are shown in Tab. 3 where each number is a percentage of the total number of questions considered (114).

From Tab. 3, we observe that, GraphEQA has the least number of false positives and false negatives hence the success rates are more reliable. We note that **Explore-EQA-GPT4o** has a considerable percentage of false positives, i.e. questions are answered correctly based on guessing without grounding the answer in the current environment. This explains why **Explore-EQA-GPT4o** has success rate comparable to **Explore-EQA-GPT4o**, even with considerably fewer planning steps (Tab. 1) – answers questions based on commonsense reasoning with high confidence and terminates early. This provides further evidence that GraphEQA effectively grounds GPT-4o in the current environment, is not overconfident based on commonsense reasoning and explores the environment until it can answer the question based on evidence.

Baseline Performance on Task Categories: Tab. 4 shows the performance of baselines and GraphEQA across the different task categories in the HM-EQA dataset. GraphEQA outperforms all other methods across all task categories, but is particularly more performant in comparison when considering Counting and State tasks. It is worth noting that the Counting and State categories are among the most challenging.

Table 4. Success Rate (%) in simulation for task categories in the HM-EQA dataset, for Identification, Counting, Existence, State, and Location categories. Reported in terms of category successes / total number of category EQA tasks. † indicates our implementation of that baseline.

Method	Ident.	Counting	Existence	State	Location
Explore-EQA	59.2	46.2	56.5	46.5	47.7
Explore-EQA-GPT4o†	32.5	44.2	56.4	42.3	40.8
SayPlanVision†	75	44.4	63.3	43.4	56
GraphEQA	**77.8**	**57.9**	**69**	**65.2**	**64**

Figure 6. Real world environments for deployment of GraphEQA on Hello Robot's Stretch RE2 platform. Left: Home. Right: Office.

6.5. Real-world Experiments

We deploy GraphEQA on **Hello Robot's Stretch RE2 platform** in two real indoor environments, *home* and *office*, as shown in Fig. 6. We design a set of 5 custom EQA questions for each environment, aligned with the task categories discussed in Sec. 3.3. We utilize Detic [59] to obtain semantic maps which, along with RGBD images, camera extrinstics, and instrinsics, are passed as input to Hydra [18] to obtain the 3D metric-semantic scene graph.
Home Environment The home environment is shown Fig. 6. Fig. 7 (a) shows results for the robot answering a '*state*' question `How many white cushions are there on the grey couch? A. One B. Two C. Three D. Four` In Fig. 7 (b), the robot is asked to answer the question `Is the front door, next to the staircase, open? A. Yes, C. No.` In both these scenarios, we note that the agent chooses semantically informed actions to explore the environment before answering the question with confidence. For additional videos of real-world experiments in the home environment, please refer to the website.
Office Environment The office environment is shown Fig. 6. Fig. 7 (c) shows the robot answering a '*existence*' question `Is my sweater on the blue couch? A. Yes B. No.` The agent moves along the lobby until the blue couch comes in view and then approaches the couch to get a better view before answering the question with confidence. Full list of tasks and qualitative results can be found in the Appendix Sec. 8.3.

7. Conclusions, Limitations, and Future Work

We present GraphEQA, an approach for solving embodied question answering tasks in unseen environments by grounding a

Figure 7. Experiments from deploying GraphEQA on the Hello Robot Stretch RE2 platform in a home environment (a,b) and in an office (c). Each set of images is from the head camera on the Stretch robot, and represents the top-k task-relevant images at each planning step. Provided under the images are the answers, confidence levels, and explanations output from the VLM planner.

vision-language based planner in the context of the current environment by providing as input textual representations of real-time 3D metric-semantic scene graphs and a task-relevant visual memory. The planner employs a semantically-guided exploration strategy for visiting semantically relevant frontiers and structurally exploring rooms and objects before using this mulitmodal memory for confidently answering the embodied question. A limitation of this approach is reliance on off-the-shelf segmentation and detection models for creating semantic maps required for 3DSG construction. The performance of our approach, hence, is directly impacted by the performance of the detection method used and the semantic categories in the scene graph are limited to the categories detected by the segmentation model. An exciting direction for future work includes enriching the scene graph online with task-relevant node and edge features using open-set models. Another limitation of our approach is that VLM-based planners can be overconfident or underconfident when answering embodied questions. Grounding calibrated models in real-time multi-modal memory is another important direction for future work.

References

[1] Christopher Agia, Krishna Murthy Jatavallabhula, Mohamed Khodeir, Ondrej Miksik, Vibhav Vineet, Mustafa Mukadam, Liam Paull, and Florian Shkurti. Taskography: Evaluating robot task planning over large 3d scene graphs. In *Conference on Robot Learning*, pages 46–58. PMLR, 2022. 2

[2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances, 2022. 3

[3] Abrar Anwar, John Welsh, Joydeep Biswas, Soha Pouya, and Yan Chang. Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation, 2024. 1, 3

[4] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R. Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera, 2019. 2, 3

[5] Samuel Bustamante Gomez, Markus Wendelin Knauer, Thun Jeremias, Stefan Schneyer, Bernhard Weber, and Freek

Stulp. Grounding embodied question-answering with state summaries from existing robot modules. In *RSS (Robotics: Science and Systems) conference 2024, Generative Modeling meets HRI Workshop*, 2024. 3

[6] Haonan Chang, Kowndinya Boyalakuntla, Shiyang Lu, Siwei Cai, Eric Jing, Shreesh Keskar, Shijie Geng, Adeeb Abbas, Lifeng Zhou, Kostas Bekris, and Abdeslam Boularias. Context-aware entity grounding with open-vocabulary 3d scene graphs, 2023. 2, 3

[7] Matthew Chang, Theophile Gervet, Mukul Khanna, Sriram Yenamandra, Dhruv Shah, So Yeon Min, Kavit Shah, Chris Paxton, Saurabh Gupta, Dhruv Batra, Roozbeh Mottaghi, Jitendra Malik, and Devendra Singh Chaplot. Goat: Go to any thing, 2023. 3

[8] Boyuan Chen, Fei Xia, Brian Ichter, Kanishka Rao, Keerthana Gopalakrishnan, Michael S Ryoo, Austin Stone, and Daniel Kappler. Open-vocabulary queryable scene representations for real world planning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11509–11522. IEEE, 2023. 3

[9] Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–10, 2018. 1, 3

[10] Runyu Ding, Jihan Yang, Chuhui Xue, Wenqing Zhang, Song Bai, and Xiaojuan Qi. Pla: Language-driven open-vocabulary 3d scene understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7010–7019, 2023. 3

[11] Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. Iqa: Visual question answering in interactive environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4089–4098, 2018. 3

[12] Elias Greve, Martin Büchner, Niclas Vödisch, Wolfram Burgard, and Abhinav Valada. Collaborative dynamic 3d scene graphs for automated driving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11118–11124. IEEE, 2024. 2

[13] Qiao Gu, Alihusein Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, Chuang Gan, Celso Miguel de Melo, Joshua B. Tenenbaum, Antonio Torralba, Florian Shkurti, and Liam Paull. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning, 2023. 1, 2, 3

[14] Daniel Honerkamp, Martin Bchner, Fabien Despinoy, Tim Welschehold, and Abhinav Valada. Language-grounded dynamic scene graphs for interactive object search with mobile manipulation. *IEEE Robotics and Automation Letters*, 9(10): 82988305, 2024. 2

[15] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10608–10615. IEEE, 2023. 3

[16] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents, 2022. 3

[17] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022. 3

[18] Nathan Hughes, Yun Chang, and Luca Carlone. Hydra: A real-time spatial perception system for 3d scene graph construction and optimization, 2022. 2, 3, 4, 8

[19] Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Omama, Tao Chen, Alaa Maalouf, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Keetha, et al. Conceptfusion: Open-set multimodal 3d mapping. *arXiv preprint arXiv:2302.07241*, 2023. 3

[20] Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. *arXiv preprint arXiv:2402.07865*, 2024. 6

[21] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19729–19739, 2023. 3

[22] Ue-Hwan Kim, Jin-Man Park, Taek-jin Song, and Jong-Hwan Kim. 3-d scene graph: A sparse and semantic representation of physical environments for intelligent agents. *IEEE Transactions on Cybernetics*, 50(12):49214933, 2020. 2, 3

[23] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 2

[24] Wenhao Li, Zhiyuan Yu, Qijin She, Zhinan Yu, Yuqing Lan, Chenyang Zhu, Ruizhen Hu, and Kai Xu. Llm-enhanced scene graph learning for household rearrangement, 2024. 2, 3

[25] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023. 3

[26] Dominic Maggio, Yun Chang, Nathan Hughes, Matthew Trang, Dan Griffith, Carlyn Dougherty, Eric Cristofalo, Lukas Schmid, and Luca Carlone. Clio: Real-time task-driven open-set 3d scene graphs. *IEEE Robotics and Automation Letters*, 9(10):8921–8928, 2024. 2, 3

[27] Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, et al. Openeqa: Embodied question answering in the era of foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16488–16498, 2024. 1, 3

[28] Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, Annie Xie, Danny Driess, Ayzaan Wahid, Zhuo Xu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint arXiv:2402.07872*, 2024. 3

[29] et al OpenAI Team. Gpt-4o system card, 2024. 4

[30] Songyou Peng, Kyle Genova, Chiyu Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser, et al. Openscene: 3d scene understanding with open vocabularies. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 815–824, 2023. 3

[31] Dicong Qiu, Wenzong Ma, Zhenfu Pan, Hui Xiong, and Junwei Liang. Open-vocabulary mobile manipulation in unseen dynamic environments with 3d semantic maps, 2024. 3

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 2

[33] Abhinav Rajvanshi, Karan Sikka, Xiao Lin, Bhoram Lee, Han-Pang Chiu, and Alvaro Velasquez. Saynav: Grounding large language models for dynamic planning to navigation in new environments, 2024. 2

[34] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 3

[35] Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning, 2023. 1, 2, 6

[36] Allen Z. Ren, Jaden Clark, Anushri Dixit, Masha Itkina, Anirudha Majumdar, and Dorsa Sadigh. Explore until confident: Efficient exploration for embodied question answering, 2024. 1, 2, 3, 6

[37] Antoni Rosinol, Andrew Violette, Marcus Abate, Nathan Hughes, Yun Chang, Jingnan Shi, Arjun Gupta, and Luca Carlone. Kimera: from slam to spatial perception with 3d dynamic scene graphs, 2021. 2, 3

[38] Nur Muhammad Mahi Shafiullah, Chris Paxton, Lerrel Pinto, Soumith Chintala, and Arthur Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory. *arXiv preprint arXiv:2210.05663*, 2022. 3

[39] Dhruv Shah, Michael Robert Equi, Błażej Osiński, Fei Xia, Brian Ichter, and Sergey Levine. Navigation with large language models: Semantic guesswork as a heuristic for planning. In *Conference on Robot Learning*, pages 2683–2699. PMLR, 2023. 3

[40] Dhruv Shah, Błażej Osiński, Sergey Levine, et al. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Conference on robot learning*, pages 492–504. PMLR, 2023. 3

[41] Rutav Shah, Albert Yu, Yifeng Zhu, Yuke Zhu, and Roberto Martn-Martn. Bumble: Unifying reasoning and acting with vision-language models for building-wide mobile manipulation, 2024. 1

[42] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3009, 2023. 3

[43] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 6

[44] Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural feature fusion fields: 3d distillation of self-supervised 2d image representations. In *2022 International Conference on 3D Vision (3DV)*, pages 443–453. IEEE, 2022. 3

[45] Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3961–3970, 2020. 2

[46] Abdelrhman Werby, Chenguang Huang, Martin Bchner, Abhinav Valada, and Wolfram Burgard. Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation. In *Robotics: Science and Systems XX*. Robotics: Science and Systems Foundation, 2024. 1, 2, 3

[47] Erik Wijmans, Samyak Datta, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. Embodied question answering in photorealistic environments with point cloud perception. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6659–6668, 2019. 3

[48] Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. Tidybot: Personalized robot assistance with large language models. *Autonomous Robots*, 47(8):1087–1102, 2023. 3

[49] Shun-Cheng Wu, Johanna Wald, Keisuke Tateno, Nassir Navab, and Federico Tombari. Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7515–7525, 2021. 2

[50] Quanting Xie, So Yeon Min, Tianyi Zhang, Kedi Xu, Aarav Bajaj, Ruslan Salakhutdinov, Matthew Johnson-Roberson, and Yonatan Bisk. Embodied-rag: General non-parametric embodied memory for retrieval and generation, 2024. 1, 3

[51] Runsen Xu, Zhiwei Huang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. Vlm-grounder: A vlm agent for zero-shot 3d visual grounding. *arXiv preprint arXiv:2410.13860*, 2024. 1, 2, 3

[52] Zhuo Xu, Hao-Tien Lewis Chiang, Zipeng Fu, Mithun George Jacob, Tingnan Zhang, Tsang-Wei Edward Lee, Wenhao Yu, Connor Schenck, David Rendleman, Dhruv Shah, et al. Mobility vla: Multimodal instruction navigation with long-context vlms and topological graphs. In *8th Annual Conference on Robot Learning*, 2024. 1, 3

[53] Karmesh Yadav, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, Theo Gervet, John Turner, Aaron Gokaslan, Noah Maestre, Angel Xuan Chang, Dhruv Batra, Manolis Savva, et al. Habitat-matterport 3d semantics dataset. *arXiv preprint arXiv:2210.05633*, 2022. 2, 3

[54] Naoki Yokoyama, Sehoon Ha, Dhruv Batra, Jiuguang Wang, and Bernadette Bucher. Vlfm: Vision-language frontier maps for zero-shot semantic navigation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 42–48. IEEE, 2024. 3

[55] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training, 2023. 4

[56] Junbo Zhang, Runpei Dong, and Kaisheng Ma. Clip-fo3d: Learning free open-world 3d scene representations from 2d dense clip. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2048–2059, 2023. 3

[57] Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. Navid: Video-based vlm plans the next step for vision-and-language navigation, 2024. 3

[58] Kaiwen Zhou, Kaizhi Zheng, Connor Pryor, Yilin Shen, Hongxia Jin, Lise Getoor, and Xin Eric Wang. Esc: Exploration with soft commonsense constraints for zero-shot object navigation. In *International Conference on Machine Learning*, pages 42829–42842. PMLR, 2023. 3

[59] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krhenbhl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision, 2022. 3, 8

GraphEQA: Using 3D Semantic Scene Graphs for Real-time Embodied Question Answering

Supplementary Material

8. Appendix

8.1. Habitat environment setup

The Habitat-Sim setup for our experiments is identical to the setup used in [36]. The camera sensor settings are as follows: image width = 640, image height = 480, camera height = 1.5m, camera tilt = -30 degrees, field of view = 120 degrees. For generating trajectories for the `<Goto_Object_node>(object_id)` and `<Goto_Frontier_node>(frontier_id)` actions, we find the shortest path between the current agent position and the desired object/frontier node location, on the obstacle-free voxel space of of the 2D occupancy map. We orient the agent such that camera always points towards the desired node location all along the trajectory. In case of the `<Goto_Object_node>(object_id)` action, this maximizes the number of views that capture the target object. In case of the `<Goto_Frontier_node>(frontier_id)` action, this makes the agent look outwards into the unexplored areas.

8.2. Prompting

8.2.1 GPT Prompt

The full prompt provided to GPT4o for GraphEQA is given in Sec. 8.5.1. In it we provide the scene graph description, description of the agent's current state, agent prompt, and just generally more descriptive text for more context.

8.2.2 Hierarchical Nature of 3DSGs and Planning

The portion of the prompt used to describe the scene graph in GraphEQA clarifies to the VLM how layers and nodes are organized in a 3DSG. We take advantage of this structure by requiring `<Goto_object_node_step>` to be organized such that the VLM first chooses a room (level 4) to navigate to, then choosing an object (level 2) in that room. This inherent structure and explanation of it in the prompt guides the VLM to choose actions that investigate objects that are semantically relevant to the question.

8.2.3 Structured Output

We employ the structured output capabilities of OpenAI's Python API to force a desired structure on what is output by GPT4o. Below is the `create_planner_response` function used in the implementation of GraphEQA.

```
def create_planner_response(frontier_node_list,
    room_node_list, region_node_list,
    object_node_list, Answer_options):

    class Goto_frontier_node_step(BaseModel):
        explanation_frontier: str
        frontier_id: frontier_node_list

    class Goto_object_node_step(BaseModel):
        explanation_room: str
        explanation_obj: str
        room_id: room_node_list
        object_id: object_node_list

    class Answer(BaseModel):
        explanation_ans: str
        answer: Answer_options
        explanation_conf: str
        confidence_level: float
        is_confident: bool

    class PlannerResponse(BaseModel):
        steps: List[Union[Goto_object_node_step,
            Goto_frontier_node_step]]
        answer: Answer
        image_description: str
        scene_graph_description: str

    return PlannerResponse
```

Code Listing 1. The `create_planner_response` function used to structure output from GTP4o.

The `create_planner_response` function takes as input enums for frontier nodes, room nodes, region nodes, object nodes, and the answer options for the particular question being answered by the VLM. These enums are used to populate the member variables of the `Goto_frontier_node_step`, `Goto_object_node_step`, and `Answer` classes, enforcing both type as well as the options available when calling the OpenAI API.

8.3. Additional Real-world experiments

Additional experiments in the office environment are shown in Fig. 9 and Fig. 10. They include both success and failure cases. For videos of additional experiments in the home environment, please refer to the website.

Figure 8. Task-relevant images selected by GraphEQA for the task *"Where is the backpack?"*.

8.3.1 Qualitative analysis

Workshop environment
GraphEQA was provided with the following multiple choice question:

> Question: *Where is the backpack?*
> A. On the chair
> B. On the table
> Answer: **A**

The agent first takes a <Goto_Object_node>(chair) step in the environment after an initial rotate-in-place mapping operation to populate the scene graph, choosing to investigate a chair found in the environment. An explanation for this choice is provided by the VLM: "Objects like the chair or table might have the backpack, so checking close to these areas is essential." The robot then begins navigating to the chair to determine if the backpack is located there. During execution of the trajectory toward the chair, GraphEQA leverages its task-relevant visual memory to score three images encountered on its way to the chair. These images are shown in Fig. 8.

After finishing the execution of this trajectory, GraphEQA answers the question with 'On the chair' and provides the following explanation of its answer: "The backpack is visually confirmed to be on the chair in the current image. The presence of the backpack on the chair makes it clear that the correct answer to the question is 'On the chair'."

> Question: *Is the door to the lobby open?*
> A. Yes. B. No.
> Answer: **A**

In this particular experiment, the agent takes three planning steps, two of which constitute <Goto_Object_node>(cabinet) steps, and one <Goto_Frontier_node>(frontier_id). For the step prior to the final action during which the agent answers the question correctly, the explanation of its current answer is "There is no direct observation of the lobby door in the current view or the scene graph. Since I've previously explored without finding the door, and considering I do not have visual confirmation, I cannot accurately determine if the door is open or closed." after which the agent takes the frontier step to explore, finding the lobby door and correctly answering the question.

We ask the agent a second question regarding the location of a blue water bottle, along with three multiple choice answers.

> Question: *Where is blue water bottle?*
> A. On the table
> B. On the cabinet
> C. On the floor
> Answer: **B**

After exploring the environment with one <Goto_Object_node>(cabinet) step, see Fig. 7, the agent successfully finds the water bottle and confirms its location, providing the following justification for its answer: "The image shows a cabinet with some objects on top, including a blue water bottle. There is also a computer monitor and various tools visible on the countertop."

Office environment We ask the following question in an office setting.

> Question: *Is my sweater on the blue couch?*
> A. Yes
> B. No Answer. **A**

The agent starts by taking a <Goto_Object_node>(couch) step, to explore the blue couch. The following VLM explanation of the object step clarifies GraphEQA is referring to the blue couch: explanation_obj='I need to locate the blue couch before I can determine if the sweater is on it or not.' object_id=<object_node_list.object_1: couch>

The low-level planner implementation on Hello Robot's stretch does not plan the entire path to the blue couch, however, resulting in several more <Goto_Object_node>(couch) steps before answering the question confidently after 11 steps.

8.4. Additional Quantitative results

8.4.1 Zero-shot reasoning for commonsense questions

We perform this evaluation to answer the following question: how many questions in the HM-EQA dataset can be answered *correctly* purely based on commonsense reasoning or guessing, without exploring the environment? We aim to perform this analysis to roughly estimate the percentage of false positives that can occur in the HM-EQA dataset using different VLMs. To answer this question we define the following three additional baselines: **EQAZeroShotGPT4o**, **EQAZeroShotPrismatic** and **EQAZeroShotGPT4oQuestionOnly**. **EQAZeroShotGPT4o** evaluates the zero-shot performance of GPT-4o when answering an EQA question using the initial image and the question. **EQAZeroShotPrismatic** evaluates the zero-shot performance of the calibrated Prismatic model from Explore-EQA [36] when answering an EQA question using the initial image and the question . **EQAZeroShotQuestionOnly** evaluates the zero-shot performance of GPT-4o when answering an EQA question using only the question. In all the above baselines, no exploration steps are taken. Prompts for the above baselines are identical to ones used

Table 5. Additional baselines (Simulation): Success rate (%)

Method	Succ. Rate (%)
Explore-EQA [36]	51.7
Explore-EQA-GPT4o	46.4
SayPlanVision	54.8
GraphEQA	**63.5**
EQAZeroShotGPT4o	17.2
EQAZeroShotPrismatic	1.8
EQAZeroShotGPT4oQuestionOnly	6.6

by Explore-EQA [36]. An episode is considered a success if the question is answered correctly and with high confidence (> 0.5).

Tab. 5 shows the simulation results for the baselines mentioned above compared to the baselines discussed in Sec. 5.1. We observe that, given only the question, **EQAZeroShotQuestionOnly** answers 6.6% of the questions correctly with high confidence. This can be attributed to the VLM answering questions based on commonsense reasoning or even just random guessing, and getting them correct. **EQAZeroShotPrismatic** answers only 1.8% of the questions correctly with a confidence > 0.5. This can be attributed to the fact that the Prismatic VLM is calibrated to avoid overconfident behavior in the absence of evidence, hence the zero-shot confidence values are low. **EQAZeroShotGPT4o** answers 17.2% of the questions correctly with high confidence. To evaluate whether these questions were answered based on actual evidence in the initial image or purely based on commonsense reasoning/guessing, we further qualitatively evaluated the successful cases. Among the 17.2% that **EQAZeroShotGPT4o** answers successfully, 8.8% were answered based on actual evidence in the initial image. These questions could be answered using the initial image. The remaining 8.4% were answered based on commonsense reasoning/guessing without any evidence from the environment. Thus, 8.4% is the rough estimate of the false positives that can occur in the HM-EQA dataset using GPT-4o based methods.

8.5. Qualitative results

8.5.1 Choosing object-nodes and frontiers semantically

In 8.5.1 we provide a qualitative examples of GraphEQA planner choosing to go near a 'couch' to better answer a question about living room lights. This shows that the planner chooses task-relevant objects to guide exploration. In 8.5.1 we provide a qualitative examples of GraphEQA planner choosing task-relevant frontier nodes and object nodes to guide exploration.

Qualitative results - choosing task-relevant object nodes

```
Question:: Is the living room light on?
A. (Do not choose this option)
B. (Do not choose this option)
C. No
D. Yes.
Answer: D
At t=0::
Agent state: The agent is currently at node agent1 at position [-2.928846836090088,
-1.5354466438293457, 4.291640281677246] at room node: room0 with name living room
Action: explanation room=I need to further investigate the living room to possibly find indicators
like light switches or bulbs. explanation obj=Going near the blue couch might provide a better
view of the surroundings, including light fixtures. room id=room 0: 'living room' object
id=object 13: 'couch'
Answer: explanation ans= I need to explore more areas related to lighting in the living room
before answering confidently. answer=Answer options A: '(Do not choose this option)' explanation
conf='The scene graph and current image do not provide enough information on lighting. Going near
specific objects like the couch could help.' confidence level=0.2 is confident=False
At t=1::
Agent state: The agent is currently at node agent7 at position [-2.2595670223236084,
-2.6618852615356445, 4.291640281677246] at room node: room 0 with name living room
Answer: explanation ans='The image clearly shows a light fixture on the ceiling that appears to be
on. This indicates that the living room light is indeed on.' answer=D 'Yes' explanation conf='The
light fixture being on is visible in the image, providing clear evidence that the living room light
is on.' confidence level=0.9 is confident=True
```

Qualitative example of GraphEQA planner choosing task-relevant object nodes for exploration.

Qualitative results - choosing task-relevant frontier nodes

```
Question::  Is the kitchen sink currently free of dishes?
A. (Do not choose this option)
B. (Do not choose this option)
C. Yes, it's empty
D. No, there are dishes.
Answer:  C
At t=0::
Agent state:  The agent is currently at node agent 1 at position [0.9115259647369385,
-0.7087370157241821, -1.5000251531600952] at room node:  room 0 with name bedroom
Action:  explanation frontier="The kitchen typically contains objects like chairs, tables, etc.
Frontier 4 is near objects that could indicate a transition to a dining or kitchen area, as it is
near the couch and chair, which might lead to a kitchen.  It's worth exploring this frontier to
find the kitchen." frontier id= frontier 4
Answer:  explanation ans="Currently, the kitchen is not visible in the scene.  The scene graph and
current image provide information about a bedroom, but not a kitchen.  Therefore, I can't answer
whether the kitchen sink is free of dishes." answer= A: '(Do not choose this option)' explanation
conf="I haven't yet gathered any information on the kitchen or the sink from my current location in
the bedroom, and no other regions of the building have been explored yet." confidence level=0.0 is
confident=False
At t=1::
Agent state:  The agent is currently at node agent 6 at position [2.5569498538970947,
2.228832721710205, -1.5000251531600952] at room node:  room 0 with name living room
Action:  explanation frontier="The frontier 2 is connected to objects like the table, cabinet,
fridge, and plates, which might lead to a kitchen area or provide a clue to the sink's status."
frontier id=frontier 2
Answer:  explanation ans='Currently, there is no visual confirmation of a sink in the image
or scene graph, and insufficient detail to determine if the kitchen sink is free of dishes.'
answer=A: '(Do not choose this option)' explanation conf="The scene graph and current image give
no direct view of a kitchen sink or its status, so it's uncertain whether it's free of dishes."
confidence level=0.2 is confident=False
At t=2::
Agent state:  The agent is currently at node agent 11 at position [2.6420648097991943,
-0.10468387603759766, -1.5000251531600952] at room node:  room 0 with name living room
Action:  explanation room='To determine if the kitchen sink is free of dishes, I need to transition
from the living room to the kitchen area, where the sink is likely located.  Frontiers associated
with kitchen-related objects can guide me there.'  explanation obj="To check the sink, moving
towards the 'plates' near object 41 might lead to the kitchen area." room id=room 0:  'living room'
object id=object 41:  'plates'
Answer:  explanation ans="Currently, I don't have enough information to directly answer if the
kitchen sink is free of dishes, based on the image or scene graph.  Further exploration is
needed to confirm the state of the sink." answer=A: '(Do not choose this option)' explanation
conf="Given my current location and the items visible, I can't see the sink or its contents.  I
need to navigate closer to where I suspect the sink might be, based on kitchen-related objects like
plates." confidence level=0.3 is confident=False
At t=3::
Agent state:The agent is currently at node agent 21 at position [1.7573769092559814,
-0.35986196994781494, -1.5000251531600952] at room node:  room 0 with name living room
Answer:  explanation ans='The image shows the kitchen, and the sink area is visible.  There
are no dishes in the sink, indicating it is empty.'  answer=C: "Yes, it's empty" explanation
conf='The image clearly shows the sink, and there are no dishes visible, providing clear evidence
to confidently answer the question.'  confidence level=1.0 is confident=True
```

Qualitative example of GraphEQA planner choosing task-relevant frontier and object nodes for exploration.

Full VLM Prompt

Agent prompt: You are an excellent hierarchical graph planning agent. Your goal is to navigate an unseen environment to confidently answer a multiple-choice question about the environment. As you explore the environment, your sensors are building a scene graph representation (in json format) and you have access to that scene graph.
Scene graph description: A scene graph represents an indoor environment in a hierarchical tree structure consisting of nodes and edges/links. There are six types of nodes: building, rooms, visited areas, frontiers, objects, and agent in the environment. The tree structure is as follows: At the highest level 5 is a 'building' node. At level 4 are room nodes. There are links connecting the building node to each room node. At the lower level 3, are region and frontier nodes. 'region' node represent region of room that is already explored. Frontier nodes represent areas that are at the boundary of visited and unexplored areas. There are links from room nodes to corresponding region and frontier nodes depicted which room they are located in. At the lowest level 2 are object nodes and agent nodes. There is an edge from region node to each object node depicting which visited area of which room the object is located in. There are also links between frontier nodes and objects nodes, depicting the objects in the vicinity of a frontier node. Finally the agent node is where you are located in the environment. There is an edge between a region node and the agent node, depicting which visited area of which room the agent is located in.
Current state description: CURRENT STATE will give you the exact location of the agent in the scene graph by giving you the agent node id, location, room_id and room name.
General Description: Given the current state information, try to answer the question. Explain the reasoning for selecting the answer. Finally, report whether you are confident in answering the question. Explain the reasoning behind the confidence level of your answer. Rate your level of confidence. Provide a value between 0 and 1; 0 for not confident at all and 1 for absolutely certain. Do not use just commonsense knowledge to decide confidence. Choose TRUE, if you have explored enough and are certain about answering the question correctly and no further exploration will help you answer the question better. Choose 'FALSE', if you are uncertain of the answer and should explore more to ground your answer in the current environment. Clarification: This is not your confidence in choosing the next action, but your confidence in answering the question correctly. If you are unable to answer the question with high confidence, and need more information to answer the question, then you can take two kinds of steps in the environment: Goto_object_node_step or Goto_frontier_node_step You also have to choose the next action, one which will enable you to answer the question better. Goto_object_node_step: Navigates near a certain object in the scene graph. Choose this action to get a good view of the region around this object, if you think going near this object will help you answer the question better. Important to note, the scene contains incomplete information about the environment (objects maybe missing, relationships might be unclear), so it is useful to go near relevant objects to get a better view to answer the question. Use a scene graph as an imperfect guide to lead you to relevant regions to inspect. Choose the object in a hierarchical manner by first reasoning about which room you should goto to best answer the question, and then choose the specific object. Goto_frontier_node_step: If you think that using action ''Goto_object_node_step'' is not useful, in other words, if you think that going near any of the object nodes in the current scene graph will not provide you with any useful information to answer the question better, then choose this action. This action will navigate you to a frontier (unexplored) region of the environment and will provide you information about new objects/rooms not yet in the scene graph. It will expand the scene graph. Choose this frontier based on the objects connected this frontier, in other words, Goto the frontier near which you see objects that are useful for answering the question or seem useful as a good exploration direction. Explain reasoning for choosing this frontier, by listing the list of objects (<id> and <name>) connected to this frontier node via a link (refer to scene graph).
While choosing either of the above actions, play close attention to 'HISTORY' especially the previous 'Action's to see if you have taken the same action at previous time steps. Avoid taking the same actions you have taken before. Describe the CURRENT IMAGE. Pay special attention to features that can help answer the question or select future actions. Describe the SCENE GRAPH. Pay special attention to features that can help answer the question or select future actions.''

Prompt 1: The prompt used in the implementation of GraphEQA.

Additional Real-world Experiments in the office environment

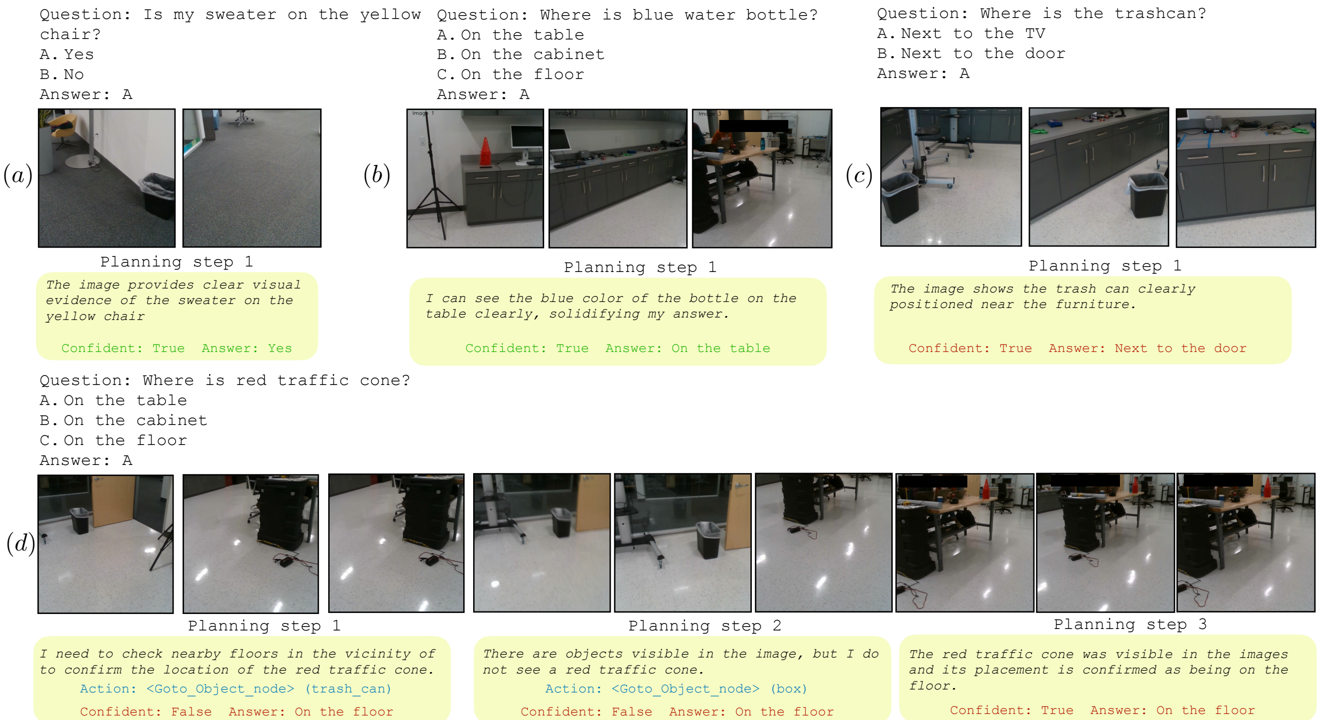
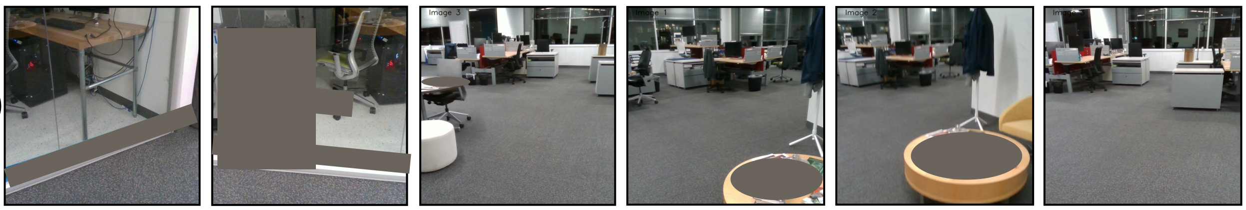


Figure 9. Additional experiments from deploying GraphEQA on the Hello Robot Stretch RE2 platform in a workshop environment (b, c, d) and in an office (a). (c) and (d) are failure cases. Each set of images is from the head camera on the Stretch robot, and represents the top-k task-relevant images at each planning step. Provided under the images are the answers, confidence levels, and explanations output from the VLM planner.

Question: Is there a bottle on one of the desks?
A. Yes
B. No
Answer: A

(e)



Planning step 1

To check for a bottle, I'll move towards the closest table.

Action: <Goto_Object_node> (table)

Confident: False Answer: No

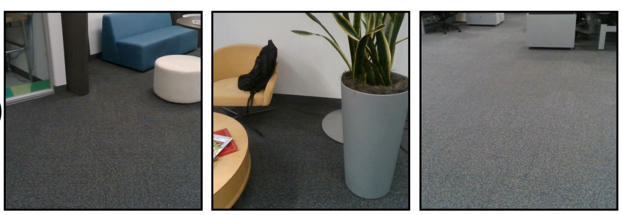
Planning step 2

Given that the image clearly shows multiple surfaces but does not include a recognizable bottle, there is no bottle.

Confident: True Answer: No

Question: Where is my backpack?
A. On the yellow chair
B. On the blue couch
Answer: A

(f)



Planning step 1

The image provides a clear view of the backpack's location on the chair.

Confident: True Answer: On the yellow chair

Figure 10. Additional experiments from deploying GraphEQA on the Hello Robot Stretch RE2 platform in an office environment (e, f). (e) is a failure case. Each set of images is from the head camera on the Stretch robot, and represents the top-k task-relevant images at each planning step. Provided under the images are the answers, confidence levels, and explanations output from the VLM planner.