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

Saumya Saxena[1*✉] **Blake Buchanan**[2*] **Chris Paxton**[3] **Peiqi Liu**[4] **Bingqing Chen**[5]
Narunas Vaskevicius[5] **Luigi Palmieri**[5] **Jonathan Francis**[1,5] **Oliver Kroemer**[1]

[1]Carnegie Mellon University [2]Neya Systems [3]Agility Robotics [4]Hello Robot [5]Bosch Center for AI
{saumyas,okroemer}@andrew.cmu.edu, bbuchanan@neyarobotics.com, chris.paxton.cs@gmail.com,
pl2285@nyu.edu, {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 to answer a situated question with confidence. This problem remains challenging in robotics, due to the difficulties in obtaining useful semantic representations, updating these representations online, and leveraging prior world knowledge for efficient planning and exploration. 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 semantics-guided exploration. We evaluate GraphEQA in simulation on two benchmark datasets, HM-EQA and OpenEQA, and demonstrate that it outperforms key baselines by completing EQA tasks with higher success rates and fewer planning steps. We further demonstrate GraphEQA in multiple real-world home and office environments. Videos and code: website.

Keywords: Embodied Question Answering, Vision Language Models, Robot Planning, Real-time 3D Scene Graphs, Guided Exploration

1 Introduction

Embodied Question Answering (EQA) [1] is a challenging task in robotics, wherein an agent is required to explore and understand a previously unseen environment sufficiently well, 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 to reason about the number of chairs around the dining table, and confidently answer the question correctly.

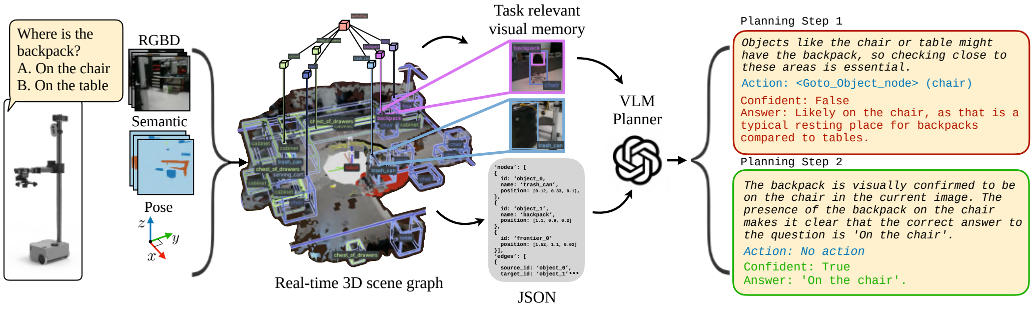
Maintaining a concise and effective memory and using it to take actions in the environment is critical for solving EQA tasks. Prior works have demonstrated the impressive commonsense reasoning

---

[*] 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.

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.



```
'nodes': [
{
  id: 'object_0',
  name: 'trash_can',
  position: [0.12, 0.33, 0.1],
},
{
  id: 'object_1',
  name: 'backpack',
  position: [1.1, 0.9, 0.2]
},
{
  id: 'frontier_0'
  position: [1.52, 1.1, 0.02]
}],
'edges': [
{
  source_id: 'object_0',
  target_id: 'object_1'...
```

Planning Step 1

Objects like the chair or table might have the backpack, so checking close to these areas is essential.

Action: <Goto_Object_node> (chair)

Confident: False

Answer: Likely on the chair, as that is a typical resting place for backpacks compared to tables.

Planning Step 2

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'.

Action: No action

Confident: True

Answer: 'On the chair'.

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.

capabilities of Vision Language Models (VLMs) as planning agents, while leveraging a semantic map for retrieval [2] or semantic exploration [3]. In such approaches, the VLMs are not grounded in the current environment, and commonsense reasoning and context-based decision-making are disconnected. Recent works [4, 5, 6, 7, 8] 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, extensive set of images [8, 9] or have to perform an expensive offline processing step to obtain a compact representation [6, 7, 10]. Thus, such semantic memory modules are either semantically rich [7, 2, 11], compact [4, 5], *or* online [5], but not all at the same time.

To address these limitations, we propose GraphEQA, an approach for embodied question answering that builds an online, compact, multimodal semantic memory combining global, semantically-sparse, and task-agnostic information from real-time 3D scene graphs [12] with local and semantically-rich information from task-relevant images [10]. 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 [12] 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, while maintaining 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 [3] and OpenEQA [13] datasets in the Habitat simulation environment [14] and also in the real world using the Hello Robot Stretch mobile manipulator in two scenes. While the focus of this work is EQA tasks, our contributions advance the broader goal of grounding vision-language foundation models in unseen 3D environments, making GraphEQA applicable to a wider range of robotics tasks. 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 [3] and OpenEQA [13] datasets, and in the real-world in two indoor environments, using the Hello Robot Stretch RE2 mobile manipulator.

2 Related Work

3D Semantic Scene Graph Representations for Planning: 3D semantic scene graphs (3DSGs) [15, 16, 17, 18, 19] have emerged as compact, semantically-rich representations of indoor environments, spurring advances in both offline [11, 2, 20, 21] and online [12, 22, 23] prediction. Offline approaches [11, 2] focus on enriching 3DSG nodes and edges with open-vocabulary semantics via Vision-Language Models (VLMs) [24, 25], supporting retrieval-based, language-guided tasks. Online methods [12, 22] enable real-time deployment of embodied agents but often rely on closed-set semantics [12] or fixed task sets [22], limiting open-world generalization. Our approach bridges these paradigms by building a *multimodal memory*: an online-constructed 3DSG with closed-set semantics guides a VLM agent toward task-relevant areas to capture semantically rich images [10], enabling open-world embodied question answering (EQA). Recent work has leveraged textual 3DSG representations for VLM-based planning [4, 26, 27, 28], primarily targeting object search or rearrangement tasks, but without addressing the deeper semantic understanding required for EQA. Other spatio-temporal representations of environments have also been effective for planning, e.g., [29], which constructs a custom 3D data structure to maintain dynamic memory.

VLMs for 3D Scene Understanding and Planning: Several previous works leverage the commonsense reasoning capabilities of foundation models for long-horizon planning [30, 31, 32]. 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 [33, 34, 31, 35]. Previous works have explored the use of VLMs for building dense queryable open-vocabulary 3D semantic representations using explicit pixel-level [36, 37, 38, 39, 40] or implicit neural [41, 42, 43] representations. However, these maps are built offline, before they are used for downstream retrieval-based tasks. Moreover, 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 [44, 8] 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 [45, 2, 11] or semantic exploration [3, 46, 47, 48, 49], however such approaches disconnect context-based decision-making and commonsense reasoning. Offline methods that build topological maps [50], keyframe selections [10], 3D semantic graphs [2, 11, 51, 20, 21] and experience summaries [6, 7, 52, 53] 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 [1, 54, 55, 56] 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. Ren et al. [3] build an explicit task-specific 2D semantic map of the environment to guide exploration, Anwar et al. [6], Xie et al. [7] build offline experience modules that the LLM can query, and Majumdar et al. [9] 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 Method

3.1 Preliminaries

Hierarchical 3D Metric-semantic Scene Graphs. 3D metric-semantic scene graphs (3DSGs) provide a structured, layered representation of environments and encode spatial, semantic, and relational information [15, 16, 17]. Recent works like Clio [22], Hydra [12], and Open Scene Graphs [57] introduce efficient real-time frameworks for incremental construction of hierarchical SG layers consisting of objects, regions, rooms, buildings, etc. See Appendix A.3 for details.

2D Occupancy Mapping and Frontier Detection. 3D voxel-based occupancy maps are an 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

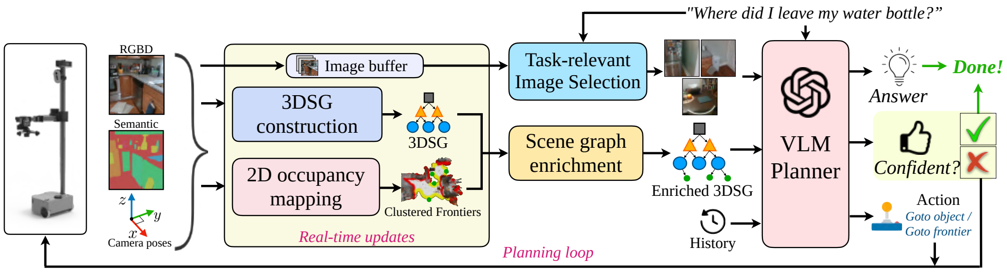


Figure 2: **Overall GraphEQA architecture.** As the agent explores the environment, it used its sensor data (RGBD images, semantic map, camera poses and intrinsics) to construct a 3D metric-semantic hierarchical scene graph (3DSG) as well as a 2D occupancy map for frontier selection in real time. The constructed 3DSG is enriched as discussed in Section 3.3. From the set of images collected during each trajectory execution, a task-relevant subset is selected, called the task-relevant visual memory (Section 3.4). A VLM-based planner (Section 3.5) 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.

are marked unexplored. Typically, the 3D occupancy map is projected into 2D, where *frontiers*—boundaries between explored and unexplored regions—are identified to guide further exploration. We employ this approach in our method for identifying frontiers, clustering them and adding them to the scene graph.

3.2 Problem Formulation

An overview of our proposed method is shown in Figure 2. Given a multiple choice question q, we wish to find the correct answer $a^* \in \mathcal{A}$ where $\mathcal{A}$ is the set of multiple-choice answers to question q available to the agent. To find a^*, the agent is equipped with a VLM-based planner $\mathcal{V}(q, \mathcal{S}_t^e, \{\mathcal{I}_k\}_{k=1}^K, \mathcal{H}_t, X_t) = (a_t, c_t, u_t)$, where $\mathcal{S}_t^e$ is the enriched scene graph constructed online at planning time t, and includes frontier nodes from 2D the occupancy map (Section 3.1). $\{\mathcal{I}_k\}_{k=1}^K$ is a small set of task-relevant images maintained in memory (Section 3.4), a_t is the current best answer to the multiple choice question q and u_t is the next action the agent should take in the environment. $\mathcal{H}_t$ represents the accrued history up to time t and X_t represents the current state of the agent. We query the VLM planner $\mathcal{V}$ (Section 3.5) at time t with the inputs described above, with the scene graph $\mathcal{S}_t^e$ continually being constructed and a set of images $\{\mathcal{I}_k\}_{k=1}^K$ chosen based on task and semantic relevance. The planner then outputs a high-level action u_t which is executed in the environment while the scene graph, visual memory, and frontiers are all updated in real time. In the following sections we provide details for each of these components.

3.3 Scene Graph Construction and Enrichment

We use Hydra [12] to construct a layered metric-semantic scene graph (see Appendix A.3), while also maintaining a 2D occupancy map of the environment depicting the explored, occupied, and unexplored navigable regions of the environment as mentioned in Section 3.1. 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 the scene graph. The output of the LLM is then used to update the room labels.

Frontier enrichment: 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-j object nodes nearest to each clustered frontier node, within a maximum distance d. We add edges to the scene graph connecting each frontier node to its top-j 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 general exploration, for example, it could be useful to choose frontier nodes near doors. We use $j = 3$ and $d = 2$ meters in our experiments, but can be varied based on the environment.

3.4 Task-relevant Visual Memory

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 [58] 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 $K = 2$; we append these K images together, along with the agent's current view, and use it as the visual input to the VLM planner at the next planning step, as shown in Figure 2.

3.5 Hierarchical Vision-Language Planner

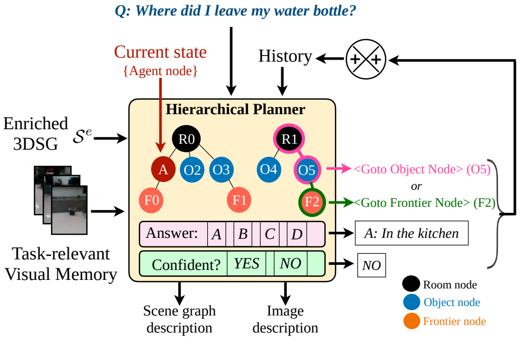


Figure 3: **VLM Planner Architecture.** The Hierarchical Vision-Language planner takes as input the question, enriched scene graph, task-relevant visual memory, current state of the robot (position and room) and a history of past states, actions, answers and confidence values. The planner chooses the 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, 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.

Inputs: At every planning step t, the VLM planner takes as input a multiple-choice question q, the set of multiple-choice answers $\mathcal{A}$, the enriched scene graph $\mathcal{S}_t^e$, and the task-relevant visual memory $\{\mathcal{I}_k\}_{k=1}^K$; 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 $\mathcal{S}_t^e$ (see Fig. 3).

Outputs: Given the above inputs, the planner must output an answer $a_t \in \mathcal{A}$ for q, 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 and its confidence in the rationale. 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. Finally, the planner is required to output a brief description of the scene graph as well as a brief description of the input images. We update the history $\mathcal{H}_t$ 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 `<Goto_Object_node>(object_id)` action types, we enforce a hierarchical planning behavior 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 behavior reflects the hierarchical structure of the 3DSG and forces the planner to reason about the hierarchical semantics of the scene to explore and answer the questions. For `<Goto_Frontier_node>(frontier_id)` action types, we require the planner to provide rationale for its choice of frontier node by referring to the object nodes connected to the selected frontier by edges in the scene graph, enforcing semantic reasoning in the frontier-selection process so that chosen frontiers are task-relevant and for information-gathering.

Termination condition: A planning episode is terminated when the planner outputs $c_t = \text{True}$ 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: We provide the planner with a system prompt detailing how to understand the scene graph structure, and explain the criteria behind choosing the actions—hierarchically for object nodes and task-relevant or information-gathering for frontier nodes. We explain that the 3DSG 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 Appendix B.

4 Experimental Setup

We identify the key research questions that 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** information from **visual memory** to actively take information gathering actions until it can confidently answer an embodied question?

4.1 Baselines and Ablations

To evaluate our method and answer the above research questions, we compare against several baselines and focus on methods that employ VLM-based planners for solving EQA or object goal navigation tasks. We compare against a strong baseline, **Explore-EQA** [3], which calibrates Prismatic-VLM [59] to answer embodied questions confidently while maintaining a 2D semantic memory and using prompted images to guide exploration. Note that ExploreEQA is always executed for a pre-specified maximum number of steps, with the highest confidence step chosen to answer the question, while GraphEQA terminates based on a confidence threshold. We implement additional variants of ExploreEQA with newer foundation models—e.g., GPT4o, Llama 4 Maverick, and Gemini 2.5 Pro—for fair comparison with respective VLM variants of GraphEQA.

We also compare GraphEQA against a modified version of **SayPlan** [4] 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. For fair comparison, we further augment SayPlanVision with some abilities of GraphEQA in order to evaluate the effectiveness of our real-time approach and to answer **Q2**; we provide it with a task-relevant visual memory and confidence-based termination.

We further evaluate our method for **Q3**, using two ablations: **GraphEQA-SG**, where the planner only has access to the real-time 3DSG and does not have access to images; and **GraphEQA-Vis**, where the planner only takes the visual memory as input and exploration is done via random frontier-based exploration. These ablations will help us evaluate the benefits of multimodality in GraphEQA.

Experimental Settings: Since we focus on multi-room environments, we evaluate GraphEQA and the baselines mentioned in Section 4.1 in simulation in Habitat-Sim [60] on scenes from HM3D-Semantics [14] on the HM-EQA and OpenEQA datasets and in the real-world in two unique home environments. GraphEQA supports open-vocabulary answers and evaluating them using an LLM, as in OpenEQA. However, to ensure consistent evaluation across both benchmarks, we augment the OpenEQA dataset with multiple choice answers using an LLM (see A.2). We perform numerous experiments with different foundation models as the VLM planner, including GPT4o, Gemini 2.5 Pro, and Llama 4 Maverick. For the real-world setup, we deploy and evaluate our approach on the *Hello Robot Stretch RE-2* mobile manipulation platform with the Stretch AI codebase [61]. All experiments are conducted on a desktop machine with two (2) NVIDIA TITAN RTX GPUs, 64GB of RAM, and an Intel i9-10900K CPU.

Resource Allocation: The average token count is broken down as 475 per image (3 images), 1133 for the prompt, and on average 5,425 scene graph tokens, for a total 7983 tokens per VLM step.

Metrics: We use the following three metrics to compare against the baselines and ablations in Section 4.1: 1) *Success Rate* (%): an episode is considered a success if the agent answers the embodied question correctly with high confidence; 2) *Average # Planning Steps*: For successful episodes, we calculate the average number of VLM planning steps. Note that while Explore-EQA runs for a preset maximum number of steps and post-calculates the maximum confidence step, we report the number of steps taken *until* the max confidence step; 3) *Average Trajectory Length* (meters): for successful episodes, we calculate the average length of the path traveled by the robot. For SayPlanVision, this includes the path traveled to generate the full scene graph.

4.2 Experimental Results

Table 1: Comparison to simulation baselines for HM-EQA and OpenEQA datasets: Success rate (%), average # of planning steps over successful trials, and L_τ average trajectory length over successful trials. Methods with a † indicate our implementations of that particular baseline.

Method	**HM-EQA**			**OpenEQA**		
	Success Rate (%) ↑	#Planning Steps ↓	L_τ (m) ↓	Success Rate (%) ↑	#Planning Steps ↓	L_τ (m) ↓
Explore-EQA [3]	51.7	18.7	38.1	55.3	20.8	39.7
Explore-EQA-GPT4o†	46.4	3.4	6.3	46.4	4.88	8.30
Explore-EQA-Llama4-Mav †	43.8	5.58	10.4	48.1	4.25	7.60
Explore-EQA-Gemini-2.5Pro †	54.3	6.40	12.3	53.0	6.45	10.9
SayPlanVision†	54.8	2.6	5.3	-	-	-
GraphEQA-GPT4o	63.5	5.1	12.6	**69.1**	3.97	8.29
GraphEQA-Llama4-Mav	57.7	**2.36**	**3.59**	53.3	2.37	3.45
GraphEQA-Gemini-2.5Pro	**67.0**	2.94	7.41	62.0	**2.16**	**4.03**

Comparison to Baselines. Table 1 shows simulation results comparing **GraphEQA** to the baselines discussed in Section 4.1 on the HM-EQA and OpenEQA datasets. Overall, GraphEQA outperforms all other baselines. Compared to **Explore-EQA**, our method completes tasks in significantly fewer planning steps and with lower trajectory length, indicating more efficient navigation. We also observe that the GPT and Llama variants of Explore-EQA have lower success rates than **Explore-EQA**, with qualitative results indicating overconfidence in VLMs' predictions (see Appendix C.4), leading to terminating episodes before exploring sufficiently. We note that Explore-EQA's Gemini variant performs better than **Explore-EQA**, likely due to the inherent spatial reasoning capabilities of Gemini 2.5 Pro. GraphEQA outperforms **SayPlanVision** even though SayPlanVision has access to the complete scene graph. For additional OpenEQA results please refer to Appendix C.5. We discuss these results in more detail below.

Baseline And Ablation Study. Regarding **Q1**, we observe from Table 1 that **GraphEQA** has higher success rate, compared to all **Explore-EQA** variants, across both HM-EQA and OpenEQA datasets, 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, as this would constrain 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.

Additional error analysis in Appendix C.4 reveals that the GPT variant of Explore-EQA has a significantly high percentage of *false positives*, i.e., questions that are answered successfully using commonsense reasoning/guessing, but without grounding the answer in the current scene. 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 Appendix C.6.

For **Q2**, we observe from Table 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 context is too large, providing for a much harder inference problem for the VLM. As a result, 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—that *incrementally* building memory by exploring task-relevant regions and maintaining a more concise representation benefits EQA tasks.

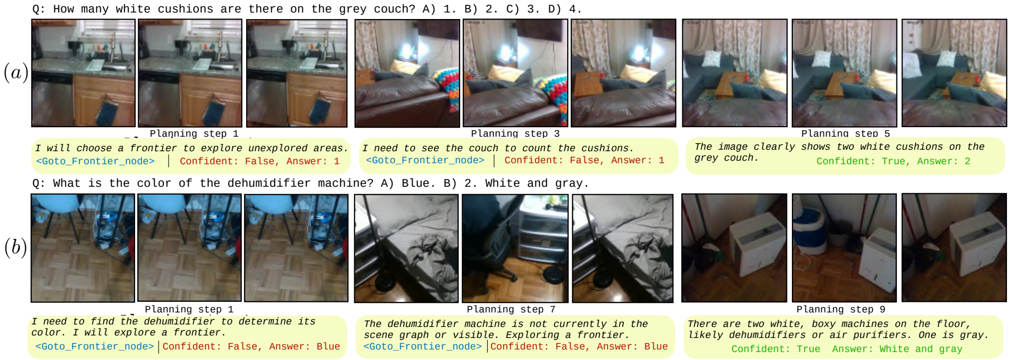

Q: How many white cushions are there on the grey couch? A) 1. B) 2. C) 3. D) 4.
(a)
Planning step 1
I will choose a frontier to explore unexplored areas.
<Goto_Frontier_node> | Confident: False, Answer: 1
Planning step 3
I need to see the couch to count the cushions.
<Goto_Frontier_node> | Confident: False, Answer: 1
Planning step 5
The image clearly shows two white cushions on the grey couch.
Confident: True, Answer: 2
Q: What is the color of the dehumidifier machine? A) Blue. B) 2. White and gray.
(b)
Planning step 1
I need to find the dehumidifier to determine its color. I will explore a frontier.
<Goto_Frontier_node> | Confident: False, Answer: Blue
Planning step 7
The dehumidifier machine is not currently in the scene graph or visible. Exploring a frontier.
<Goto_Frontier_node> | Confident: False, Answer: Blue
Planning step 9
There are two white, boxy machines on the floor, likely dehumidifiers or air purifiers. One is gray.
Confident: True Answer: White and gray

Figure 4: **Images from real-world experiments**, deploying GraphEQA on the Hello Robot Stretch RE2 platform in two unique home environments (a, b). Each set of images is from the head camera on the Stretch robot, representing the top-K task-relevant images at each planning step as it constructs the scene graph and attempts to answer the question with high confidence. Provided under the images are planning step, answers, confidence, and explanations output from the VLM planner.

For **Q3**, the ablation results in Table 2 for **GraphEQA-SG** show lower success rate, higher average planning steps, and higher average trajectory length, compared to GraphEQA, as it only uses the 3DSG as textual input to the VLM planner, demonstrating that a semantic scene graph constructed using closed-set semantics and without any task-specific semantic enrichment will provide an incomplete and insufficient environment understanding; GraphEQA's task-relevant visual memory 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 scene structure to guide exploration. However, qualitatively we observe that without access to a scene graph to ground the agent in the current environment, GraphEQA-Vis exhibits overconfidence (taking very few planning steps) and tries to answer the question solely based on the current image. **GraphEQA** outperforms all ablations, providing clear evidence on the utility of a multimodal approach that combines global spatial and semantic information from 3D scene graphs with local but rich semantic information from images, for solving challenging EQA tasks. We also observe from Table 1 that GraphEQA exhibits lower average planning steps and lower average trajectory length, while retaining a higher success rate, highlighting the capability of GraphEQA to reason about multimodal data in the form of semantically sparse and task-agnostic scene graphs and more semantically rich information in the form of task-relevant visual memory. Additional ablations are available in Appendix C.2.

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	**63.5**	5.1	12.6

Real-world Experiments. We deploy GraphEQA on the **Hello Robot Stretch RE2** platform across two home environments. We conduct ten trials in Home (a) and five trials in Home (b). Figure 4 illustrates representative runs from each setting. For each environment, we design a custom set of Embodied Question Answering (EQA) tasks (see Appendix D and website), aligned with the task categories described in Appendix A.2. To construct the underlying 3D metric-semantic scene graph, GraphEQA employs Detic [62] for semantic segmentation and integrates RGB-D images along with camera intrinsics and extrinsics as input to the Hydra scene graph generator [12].

5 Conclusion

We present GraphEQA, an approach for solving embodied question answering tasks in unseen environments by grounding a 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. We show that GraphEQA achieves improved performance over existing approaches on EQA tasks in both the HM-EQA and OpenEQA benchmarks, and validate performance across both *closed* and *open* multimodal foundation models. Furthermore, we validate its practical applicability through real-world deployments in two indoor environments.

6 Limitations

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. Additionally, GraphEQA is currently limited to single-floor scenes, and is unable to traverse multiple floors or buildings. Accommodating multi-floor scenes is within the scope of future work.

References

[1] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–10, 2018. 1, 3

[2] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, C. Gan, C. M. de Melo, J. B. Tenenbaum, A. Torralba, F. Shkurti, and L. Paull. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning, 2023. URL `https://arxiv.org/abs/2309.16650`. 2, 3, 11

[3] A. Z. Ren, J. Clark, A. Dixit, M. Itkina, A. Majumdar, and D. Sadigh. Explore until confident: Efficient exploration for embodied question answering, 2024. URL `https://arxiv.org/abs/2403.15941`. 2, 3, 6, 7, 1, 5

[4] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf. Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning, 2023. URL `https://arxiv.org/abs/2307.06135`. 2, 3, 6

[5] R. Shah, A. Yu, Y. Zhu, Y. Zhu, and R. Martn-Martn. Bumble: Unifying reasoning and acting with vision-language models for building-wide mobile manipulation, 2024. URL `https://arxiv.org/abs/2410.06237`. 2

[6] A. Anwar, J. Welsh, J. Biswas, S. Pouya, and Y. Chang. Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation, 2024. URL `https://arxiv.org/abs/2409.13682`. 2, 3

[7] Q. Xie, S. Y. Min, T. Zhang, K. Xu, A. Bajaj, R. Salakhutdinov, M. Johnson-Roberson, and Y. Bisk. Embodied-rag: General non-parametric embodied memory for retrieval and generation, 2024. URL `https://arxiv.org/abs/2409.18313`. 2, 3

[8] Z. Xu, H.-T. L. Chiang, Z. Fu, M. G. Jacob, T. Zhang, T.-W. E. Lee, W. Yu, C. Schenck, D. Rendleman, D. Shah, et al. Mobility vla: Multimodal instruction navigation with long-context vlms and topological graphs. In *8th Annual Conference on Robot Learning*, 2024. 2, 3

[9] A. Majumdar, A. Ajay, X. Zhang, P. Putta, S. Yenamandra, M. Henaff, S. Silwal, P. Mcvay, O. Maksymets, S. 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. 2, 3

[10] R. Xu, Z. Huang, T. Wang, Y. Chen, J. Pang, and D. Lin. Vlm-grounder: A vlm agent for zero-shot 3d visual grounding. *arXiv preprint arXiv:2410.13860*, 2024. 2, 3

[11] A. Werby, C. Huang, M. Bchner, A. Valada, and W. Burgard. Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation. In *Robotics: Science and Systems XX*, RSS2024. Robotics: Science and Systems Foundation, July 2024. doi:10.15607/rss.2024.xx.077. URL `http://dx.doi.org/10.15607/RSS.2024.XX.077`. 2, 3

[12] N. Hughes, Y. Chang, and L. Carlone. Hydra: A real-time spatial perception system for 3d scene graph construction and optimization, 2022. URL `https://arxiv.org/abs/2201.13360`. 2, 3, 4, 8, 1

[13] A. Majumdar, A. Ajay, X. Zhang, P. Putta, S. Yenamandra, M. Henaff, S. Silwal, P. Mcvay, O. Maksymets, S. Arnaud, K. Yadav, Q. Li, B. Newman, M. Sharma, V. Berges, S. Zhang, P. Agrawal, Y. Bisk, D. Batra, M. Kalakrishnan, F. Meier, C. Paxton, S. Sax, and A. Rajeswaran. Openeqa: Embodied question answering in the era of foundation models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2, 1

[14] K. Yadav, R. Ramrakhya, S. K. Ramakrishnan, T. Gervet, J. Turner, A. Gokaslan, N. Maestre, A. X. Chang, D. Batra, M. Savva, et al. Habitat-matterport 3d semantics dataset. *arXiv preprint arXiv:2210.05633*, 2022. URL `https://arxiv.org/abs/2210.05633`. 2, 6

[15] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera, 2019. URL `https://arxiv.org/abs/1910.02527`. 3

[16] U.-H. Kim, J.-M. Park, T.-j. Song, and J.-H. Kim. 3-d scene graph: A sparse and semantic representation of physical environments for intelligent agents. *IEEE Transactions on Cybernetics*, 50(12):49214933, Dec. 2020. ISSN 2168-2275. doi:10.1109/tcyb.2019.2931042. URL `http://dx.doi.org/10.1109/TCYB.2019.2931042`. 3

[17] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone. Kimera: from slam to spatial perception with 3d dynamic scene graphs, 2021. URL `https://arxiv.org/abs/2101.06894`. 3

[18] J. Wald, H. Dhamo, N. Navab, and F. 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. 3

[19] E. Greve, M. Büchner, N. Vödisch, W. Burgard, and A. Valada. Collaborative dynamic 3d scene graphs for automated driving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11118–11124. IEEE, 2024. 3

[20] W. Li, Z. Yu, Q. She, Z. Yu, Y. Lan, C. Zhu, R. Hu, and K. Xu. Llm-enhanced scene graph learning for household rearrangement, 2024. URL `https://arxiv.org/abs/2408.12093`. 3

[21] H. Chang, K. Boyalakuntla, S. Lu, S. Cai, E. Jing, S. Keskar, S. Geng, A. Abbas, L. Zhou, K. Bekris, and A. Boularias. Context-aware entity grounding with open-vocabulary 3d scene graphs, 2023. URL `https://arxiv.org/abs/2309.15940`. 3

[22] D. Maggio, Y. Chang, N. Hughes, M. Trang, D. Griffith, C. Dougherty, E. Cristofalo, L. Schmid, and L. Carlone. Clio: Real-time task-driven open-set 3d scene graphs. *IEEE Robotics and Automation Letters*, 9(10):8921–8928, 2024. doi:10.1109/LRA.2024.3451395. 3

[23] S.-C. Wu, J. Wald, K. Tateno, N. Navab, and F. 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. 3

[24] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 3

[25] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 3

[26] D. Honerkamp, M. Bchner, F. Despinoy, T. Welschehold, and A. Valada. Language-grounded dynamic scene graphs for interactive object search with mobile manipulation. *IEEE Robotics and Automation Letters*, 9(10):82988305, Oct. 2024. ISSN 2377-3774. doi:10.1109/lra.2024.3441495. URL `http://dx.doi.org/10.1109/LRA.2024.3441495`. 3

[27] A. Rajvanshi, K. Sikka, X. Lin, B. Lee, H.-P. Chiu, and A. Velasquez. Saynav: Grounding large language models for dynamic planning to navigation in new environments, 2024. URL `https://arxiv.org/abs/2309.04077`. 3

[28] C. Agia, K. M. Jatavallabhula, M. Khodeir, O. Miksik, V. Vineet, M. Mukadam, L. Paull, and F. Shkurti. Taskography: Evaluating robot task planning over large 3d scene graphs. In *Conference on Robot Learning*, pages 46–58. PMLR, 2022. 3

[29] P. Liu, Z. Guo, M. Warke, S. Chintala, C. Paxton, N. M. M. Shafiullah, and L. Pinto. Dynamem: Online dynamic spatio-semantic memory for open world mobile manipulation. *arXiv preprint arXiv:2411.04999*, 2024. doi:10.48550/arXiv.2411.04999. URL `https://doi.org/10.48550/arXiv.2411.04999`. 3

[30] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng. Do as i can, not as i say: Grounding language in robotic affordances, 2022. URL `https://arxiv.org/abs/2204.01691`. 3

[31] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents, 2022. URL `https://arxiv.org/abs/2201.07207`. 3

[32] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022. 3

[33] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. 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

[34] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. 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

[35] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. 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

[36] R. Ding, J. Yang, C. Xue, W. Zhang, S. Bai, and X. 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

[37] S. Peng, K. Genova, C. Jiang, A. Tagliasacchi, M. Pollefeys, T. 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

[38] C. Huang, O. Mees, A. Zeng, and W. Burgard. Visual language maps for robot navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10608–10615. IEEE, 2023. 3

[39] J. Zhang, R. Dong, and K. 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

[40] K. M. Jatavallabhula, A. Kuwajerwala, Q. Gu, M. Omama, T. Chen, A. Maalouf, S. Li, G. Iyer, S. Saryazdi, N. Keetha, et al. Conceptfusion: Open-set multimodal 3d mapping. *arXiv preprint arXiv:2302.07241*, 2023. 3

[41] N. M. M. Shafiullah, C. Paxton, L. Pinto, S. Chintala, and A. Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory. *arXiv preprint arXiv:2210.05663*, 2022. 3

[42] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. Lerf: Language embedded radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19729–19739, 2023. 3

[43] V. Tschernezki, I. Laina, D. Larlus, and A. 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

[44] J. Zhang, K. Wang, R. Xu, G. Zhou, Y. Hong, X. Fang, Q. Wu, Z. Zhang, and H. Wang. Navid: Video-based vlm plans the next step for vision-and-language navigation, 2024. URL `https://arxiv.org/abs/2402.15852`. 3

[45] M. Chang, T. Gervet, M. Khanna, S. Yenamandra, D. Shah, S. Y. Min, K. Shah, C. Paxton, S. Gupta, D. Batra, R. Mottaghi, J. Malik, and D. S. Chaplot. Goat: Go to any thing, 2023. URL `https://arxiv.org/abs/2311.06430`. 3

[46] K. Zhou, K. Zheng, C. Pryor, Y. Shen, H. Jin, L. Getoor, and X. E. Wang. Esc: Exploration with soft commonsense constraints for zero-shot object navigation. In *International Conference on Machine Learning*, pages 42829–42842. PMLR, 2023. 3

[47] N. Yokoyama, S. Ha, D. Batra, J. Wang, and B. 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

[48] D. Shah, M. R. Equi, B. Osiński, F. Xia, B. Ichter, and S. Levine. Navigation with large language models: Semantic guesswork as a heuristic for planning. In *Conference on Robot Learning*, pages 2683–2699. PMLR, 2023. 3

[49] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess, A. Wahid, Z. Xu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint arXiv:2402.07872*, 2024. 3

[50] D. Shah, B. Osiński, S. 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

[51] D. Qiu, W. Ma, Z. Pan, H. Xiong, and J. Liang. Open-vocabulary mobile manipulation in unseen dynamic environments with 3d semantic maps, 2024. URL `https://arxiv.org/abs/2406.18115`. 3

[52] S. Bustamante Gomez, M. W. Knauer, T. Jeremias, S. Schneyer, B. Weber, and F. 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

[53] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser. Tidybot: Personalized robot assistance with large language models. *Autonomous Robots*, 47(8):1087–1102, 2023. 3

[54] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. 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

[55] E. Wijmans, S. Datta, O. Maksymets, A. Das, G. Gkioxari, S. Lee, I. Essa, D. Parikh, and D. 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

[56] J. Francis, N. Kitamura, F. Labelle, X. Lu, I. Navarro, and J. Oh. Core challenges in embodied vision-language planning. *Journal of Artificial Intelligence Research*, 74:459–515, 2022. 3

[57] J. Loo, Z. Wu, and D. Hsu. Open scene graphs for open world object-goal navigation, 2024. URL `https://arxiv.org/abs/2407.02473`. 3

[58] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training, 2023. URL `https://arxiv.org/abs/2303.15343`. 5

[59] S. Karamcheti, S. Nair, A. Balakrishna, P. Liang, T. Kollar, and D. Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. *arXiv preprint arXiv:2402.07865*, 2024. 6

[60] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. Chaplot, O. Maksymets, A. Gokaslan, V. Vondrus, S. Dharur, F. Meier, W. Galuba, A. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 6

[61] Hello-Robot. Hello-robot, stretch ai. URL `https://github.com/hello-robot/stretch_ai`. 6

[62] X. Zhou, R. Girdhar, A. Joulin, P. Krhenbhl, and I. Misra. Detecting twenty-thousand classes using image-level supervision, 2022. URL `https://arxiv.org/abs/2201.02605`. 8, 2

[63] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. M. Turner, E. Undersander, W. Galuba, A. Westbury, A. X. Chang, M. Savva, Y. Zhao, and D. 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. URL `https://openreview.net/forum?id=-v4OuqNs5P`. 1

A Experiment setup

A.1 Habitat environment setup

The Habitat-Sim setup for our experiments is identical to the setup used in [3]. 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.

A.2 HM-EQA and OpenEQA Benchmarks

HM-EQA: The Habitat-Matterport Embodied Question Answering (HM-EQA) dataset introduced by Ren et al. [3] is based in the Habitat-Matterport 3D Research Dataset (HM3D) of photo-realistic, diverse indoor 3D scans [63]. 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. A sample from the *identification* category is as follows: `Q: Which rug did I put next to the kitchen sink? A. There is no rug. B. White one. C. Gray one. D. Green one.`

OpenEQA: The OpenEQA benchmark [13] is composed of data for two different settings, *EM-EQA* (episodic memory) and *A-EQA* (active exploration), with questions spanning seven categories, spatial understanding, object state recognition, functional reasoning, attribute recognition, world knowledge, object localization, and object recognition, in two different environments, HM3D and ScanNet. The active exploration dataset (A-EQA) consists of 557 questions and corresponding open vocabulary answers, for example, `Q: Do I have Cayenne pepper left at home? A: I found a bottle of Cayenne pepper in the pantry.` In this setting, the agent is supposed to explore the environment autonomously to answer the question. The episodic memory dataset (EM-EQA) consists of a sequence of historical sensory (RGB) observations in addition to each question-answer pair. In this setting, the agent is supposed to analyze the episodic memory to answer the question. To evaluate for success, an LLM is used to score the agent's answer based on similarity to the ground truth answer in the dataset. Evidently, the A-EQA dataset is relevant to our GraphEQA setting, where the agent needs to explore an *unseen* environment to solve an EQA task. We consider A-EQA questions only in the HM3D environment, resulting in a total of 113 questions. We exclude ScanNet questions, as ScanNet is predominantly composed of single-room environments and therefore does not present a substantial exploration challenge for GraphEQA. Furthermore, although GraphEQA supports querying natural language answers from the VLM planner (instead of multiple choice answers) and evaluating them using an LLM, as in OpenEQA; to ensure consistent evaluation to HM-EQA, we augmented the A-EQA dataset with multiple choice answers using an LLM. The correct answer is provided as one of the four choices.

A.3 3D Scene Graph Construction

We use Hydra [12] to construct a layered metric-semantic scene graph. Hydra 3DSGs are comprised of the following layers: *Layer 1 (bottom)*: a metric-semantic 3D mesh, *Layer 2*: objects with corresponding semantic labels and the 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, semantic segmentation masks, camera extrinsics, and intrinsics. The HM3D-Semantics dataset provides ground-truth semantic segmentation masks for simulation experiments. In real-world experiments, we use Detic [62] to obtain segmentation masks.

B Prompting

B.1 GPT Prompt

The full prompt provided to GPT4o for GraphEQA is given below. 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.

B.2 Exploiting Hierarchical Nature of 3DSGs for 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.

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.’’

The prompt used in the implementation of GraphEQA.

B.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.

```python
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 GPT4o.

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.

C Additional Simulation Experiments

C.1 Performance across Task Categories

Table 3: 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**

Table 3 shows the performance of baselines and GraphEQA across the different task categories (identification, counting, existence, state, location) 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.

C.2 Additional Ablations

We perform some additional ablations to evaluate the utility of different components of our method: **GraphEQA-NoEnrich**, which does not use frontier enrichment (Section 3.3), and **GraphEQA-CurrView**, which uses only the current view as input to the VLM and does not choose additional task-relevant keyframes (Section 3.4). All ablations of GraphEQA use GPT-4o. Here we analyze these 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 Section 3.3, 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.

Table 4: Ablations (Simulation): Success rate (%), number of planning steps and L_τ the trajectory length.

Ablation	Succ. Rate (%)	#Planning steps	L_τ (m)
GraphEQA-NoEnrich	59.5	5.1	11.1
GraphEQA-CurrView	53.1	5.7	12.2
GraphEQA	**63.5**	5.1	12.6

C.3 Additional Quantitative Analysis: 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 [3] 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 by Explore-EQA [3]. An episode is considered a success if the question is answered correctly and with high confidence (> 0.5).

Table 5 shows the simulation results for the baselines mentioned above compared to the baselines discussed in Section 4.1. 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.

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

Method	Succ. Rate (%)
Explore-EQA [3]	51.7
Explore-EQA-GPT4o	46.4
SayPlanVision	54.8
GraphEQA	**63.5**
EQAZeroShotGPT4o	17.2
EQAZeroShotPrismatic	1.8
EQAZeroShotGPT4oQuestionOnly	6.6

C.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 Section 6 Figure 5. Furthermore, we also notice ***false negatives***, where the answer was marked incorrect given 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 Figure 6. To get an estimate of the number of false positives and false negatives in our baselines, we consider 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 Table 6 where each number is a percentage of the total number of questions considered (114). We limit this error analysis to only the GPT variant, as it is performed manually by a human.

Table 6: 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

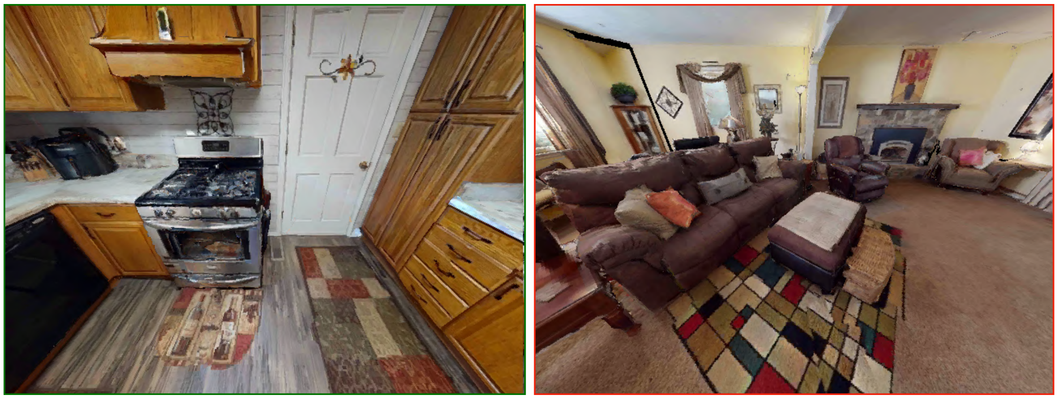
From Table 6, we observe that GraphEQA has the least number of false positives and false negatives, yielding more reliable success rates. We note that **Explore-EQA-GPT4o** has a considerable percentage of false positives, i.e., questions are answered correctly based on guessing without grounding the answer in the current environment. This sheds light on why **Explore-EQA-GPT4o** has comparable success rates to **Explore-EQA-GPT4o**, even with considerably fewer planning steps (Table 1).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. See additional results in Appendix C.1, categorized by question type.

C.5 Comparison to OpenEQA baselines on the A-EQA dataset

As discussed in A.2, to ensure consistent evaluation across both simulation benchmarks, we augmented the A-EQA dataset with multiple-choice answers using an LLM. In the original A-EQA setting (with open-vocabulary responses and LLM-based evaluation), GraphEQA-GPT4o achieves a success rate of 53.6%, which, as expected, is lower than the multiple choice setting given the increased difficulty of the open-vocabulary task.

Comparison to Multi-Frame VLM - We compare our result above, to OpenEQA's best performing method in the A-EQA setting – **Multi-Frame VLM**. Since code for A-EQA experiments is not released (only for EM-EQA), and some implementation details such as the random exploration strategy and termination condition are not specified, we refer to the results reported in the paper, where Multi-Frame VLM achieves a success rate of 41.8%. We attribute the performance gap between

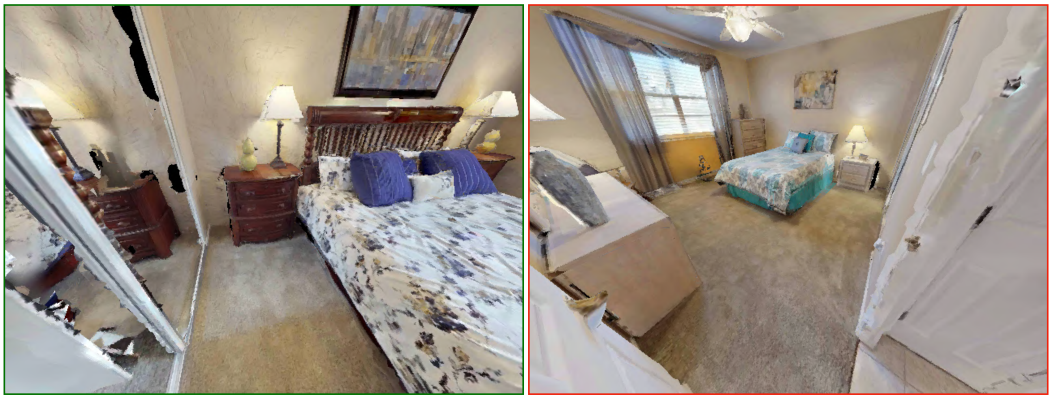
Question: Did I leave any pot on the stove?
A. Two B. None C. Three D. One
Answer: B



True Positive False Positive

Figure 5: 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.

Question: Which pillows are there on the bed right now?
A. Green ones B. Black ones C. Red ones D. Purple ones
Answer: D



True Positive False Negative

Figure 6: 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.

GraphEQA and **Multi-Frame VLM** to: (1) Multi-Frame VLM uses random exploration, whereas GraphEQA explores via semantic guidance; (2) Multi-Frame VLM’s visual memory consists of 50 *uniformly sampled* frames, which risks missing critical task-relevant images in large HM3D scenes (hundreds of images); and (3) as evidenced by comparisons to SayPlanVision, compact task-relevant memory, as used in GraphEQA, outperforms large-context inputs (large number of images in Multi-Frame VLM), since overly large contexts can degrade VLM performance.

C.6 Qualitative Analysis

C.6.1 Exploration Efficiency and Trajectory Generation

We illustrate the differences in exploration between GraphEQA and Explore-EQA through the following example. Figure 7(a) shows the trajectory taken by an agent employing Explore-EQA in a scene in the HM3D dataset taking 30 VLM steps, while Figure 7(b) shows the trajectory taken by an agent employing GraphEQA in the same scene taking a total of 5 VLM steps. We highlight here how Explore-EQA not only takes more steps, but that steps are often guided only by the semantic map constructed by images, leading to more exploration, while GraphEQA takes more structured steps in its environment, guided by the scene graph, to answer the question.

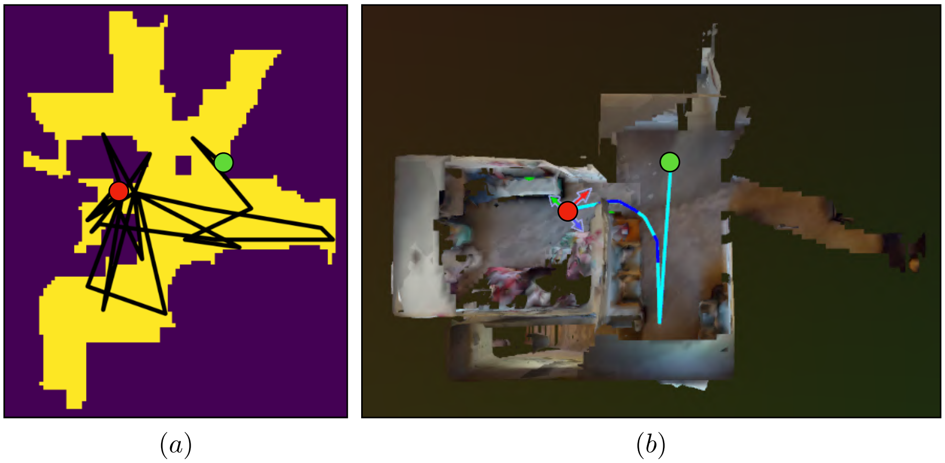

(a) (b)

Figure 7: (a) A top down view of the map constructed by Explore-EQA illustrating explored areas for a scene in the HM3D dataset. The agent's initial position is depicted in green and its final position in red. The trajectory of the agent is shown in black. (b) A top down view of scene reconstruction output by as GraphEQA explores the same HM3D scene used in the Explore-EQA map to the left. The agent's initial position is again in green and its final position in red. The agent's trajectory is shown in blue in this image. In this figure, we want to bring attention to the contrast between a typical trajectory taken by an agent employing GraphEQA vs. a typical trajectory taken by an agent employing Explore-EQA.

C.6.2 Choosing object-nodes and frontiers semantically

In C.6.3 we provide 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 C.6.4 we provide a qualitative examples of GraphEQA planner choosing task-relevant frontier nodes and object nodes to guide exploration.

C.6.3 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.

C.6.4 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.

C.7 Additional Discussion

Novelty of GraphEQA without a learning component: Even without a learned component, several aspects of GraphEQA are novel - a real-time, compact, actionable, multimodal memory for grounding VLMs in unseen 3D environments and a semantically-informed efficient exploration strategy. Moreover, we argue that modular and easily customizable methodologies like GraphEQA are effective for zero-shot deployment in novel real-world environments without requiring expensive end-to-end training, as reflected in our real-world experiments. The performance of certain modules can be improved with task-specific fine-tuning, however, that can come at the cost of open-world generalizability.

How can GraphEQA handle multiple questions in the same episode? To handle multiple questions, our method can be extended by maintaining a separate visual memory per question and discarding as each question is answered. However, handling temporally extended questions that require an unknown number of images remains an important direction for future work.

Utility of using both binary and numeric confidence: Using both binary and numeric confidence for episode termination is empirically motivated. In practice, this redundancy improved the reliability of numerical scores and provided a more robust indicator of the VLM's confidence.

Broader applicability of GraphEQA beyond EQA tasks: Although we mainly focus on EQA tasks, GraphEQA is broadly applicable to other navigation tasks such as object-goal navigation, leveraging its ability to perform grounded exploration in unseen 3D environments. In particular, EQA presents a greater challenge than object-goal navigation, as it requires more complex semantic and spatial reasoning, including identification, counting, existence, state, and functional understanding (see OpenEQA [9], Fig. 5). Compared to GraphEQA, methods like ConceptGraphs [2], which rely on *enriched* scene graphs without visual context, face key limitations on EQA tasks: 1) Enriching scene graphs with relational information via LLMs may suffice for object-goal navigation but falls short for EQA, which often demands nuanced, fine-grained semantic understanding that may not be captured in the graph. While task-specific enrichment is possible, it is not scalable across diverse EQA tasks. GraphEQA addresses this challenge by maintaining semantically-rich task-relevant images in its multi-modal memory and using a VLM for in-context reasoning. 2) Open-vocabulary 3D instance segmentation, used in ConceptGraphs, is computationally expensive – even when applied incrementally – and limits the real-time applicability of these methods. GraphEQA maintains a light-weight yet semantically rich multi-modal memory which can be constructed in real-time.

D Real-world Experiments in Home Environments

Experiments for Home Environments (a, b) and the questions asked of GraphEQA are given below. We provide a sequence of ten images from the head camera on the robot to illustrate exploration and validation of GraphEQA's answer to the question. Each experiment was executed twice successfully. We show a single trial from this set for each experiment. For videos of real-world experiments, please refer to the website.

D.1 Home Environment (a)

Question 1: *Is there a blue pan on the stove?*
A. Yes
B. No
Answer: **A**

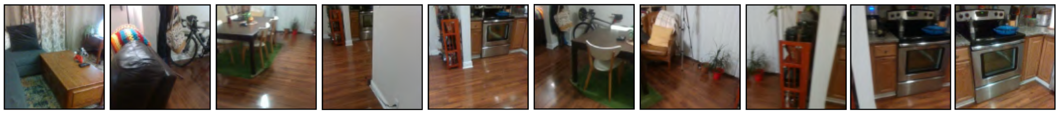


Figure 8: **Home Environment (a) Question 1: Is there a blue pan on the stove?** For this question, the agent takes four VLM steps in the environment, beginning by reasoning about the current information it has access to; "I can't answer confidently as the current view and scene graph don't reveal a kitchen or stove.". The agent then takes two GotoFrontierNodeStep steps to explore, and as shown in Frame 5 (from the left) gains access to the stove via the scene graph: "The stove is connected to region and frontier nodes, suggesting proximity.". The agent then takes a GotoObjectNodeStep(stove) step, navigates to the stove, and upon reaching it answers **"The image shows a blue pan on the stove."**, answering with high confidence.

Question 2: *How many white cushions are there on the grey couch?*
A. 1
B. 2
C. 3
D. 4
Answer: **B**

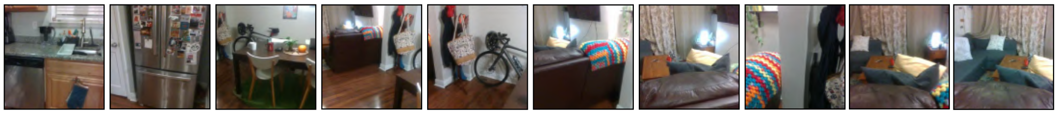


Figure 9: **Home Environment (a) Question 2: How many white cushions are there on the grey couch?** For this question, the agent takes a total of six VLM steps, initiating its exploration by reasoning about the frontiers and objects it has access to in the scene graph; "No frontiers or objects are directly related to locating the grey couch. I'll choose a frontier to explore unexplored areas.". The agent then has access to a couch (see Frame 4 from the left) and chooses GotoObjectNodeStep(couch) for two VLM steps to search around the only couch it can see. After executing these object node steps, the agent then has access to a secondary couch it has not explored, and so chooses GotoObjectNodeStep(couch) once more to explore the grey couch, stating "The current view doesn't clearly show the number of white cushions on the grey couch, so I need a closer look." In the final VLM step the agent answers the question; **"The image shows a grey couch with two white cushions. There is also a table with various items in front of the couch."**

Question 3: *Where is my handbag?*
A. On the coat rack
B. On the floor
C. On the couch
Answer: **A**

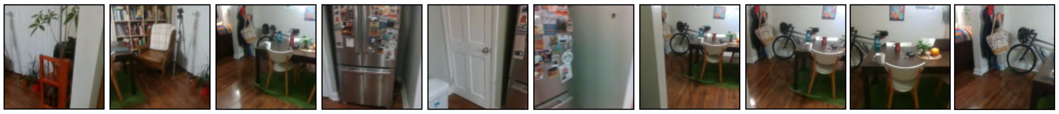


Figure 10: **Home Environment (a) Question 3: Where is my handbag?** The agent takes a total of three VLM steps to determine the location of the handbag, beginning with exploration guided by the scene graph; "The scene graph shows several frontiers connected to objects or areas, but none seem directly related to a potential couch or coat rack. Exploration of frontiers is necessary". After the first GotoFrontierNodeStep to explore, the agent identifies a handbag in the environment; "Although the scene graph indicates the handbag is in the living room, further exploration is needed to confirm its position.", but cannot yet confirm its position among the options available to it. Finally, the agent takes a GotoObjectNodeStep to gain a better view of the handbag and its position; **"I have visual confirmation from the current image showing the handbag on the coat rack.".** To see the full experiment in video, please see website.

Question 4: *Where is the trashcan?*
A. Next to the sink
B. Next to the refrigerator
Answer: **A**

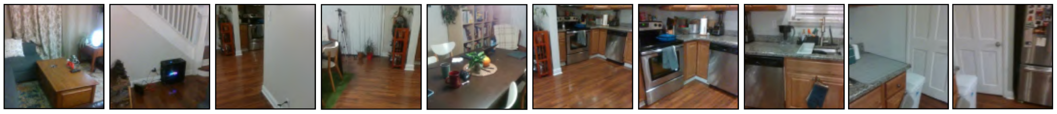


Figure 11: **Home Environment (a) Question 4: Where is the trashcan?** The agent takes a total of six VLM steps to determine the location of the trashcan, beginning with exploration guided by the scene graph; "The image and scene graph don't provide information about a kitchen or a trashcan location. Choosing a frontier that might lead to a kitchen." The following four steps are GotoFrontierNodeStep actions, as the agent prioritizes exploring over investigating specific objects in the environment; "The current scene graph shows objects like a cushion, stove, chair, table, blanket, and other kitchen-related items like a dishwasher, hand towel, cabinet, and sink in the vicinity." Finally, the agent visually confirms the location of the trashcan; **"Given the clear visibility of the trashcan next to the sink in Image 1, I am confident in answering with certainty."** Note that *Image 1* for this experiment is the 9th image in the sequence of ten above.

Question 5: *Is the front door next to the staircase open?*
A. Yes
B. No
Answer: **B**

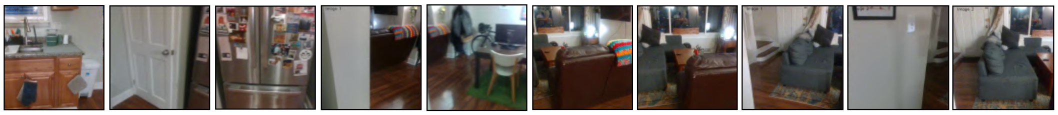


Figure 12: **Home Environment (a)** Question 5: **Is the front door next to the staircase open?** The agent takes a total of four VLM steps to determine the state of the door, beginning its exploration by identifying where it is located and choosing an action appropriately; "The scene graph shows the agent's location in the kitchen. There aren't any nodes indicating a door or staircase. I will explore a frontier as it could lead to a new area where the front door or staircase might be visible". The agent's next two actions are GotoFrontierNodeStep actions, explaining the second of these choices as follows; "To locate the staircase, I will explore the frontier connected to a couch, which could be in the living room area.". The agent then chooses another frontier since "...the current view does not show much that relates directly to the front door or staircase." Along its trajectory to that frontier node, the agent identifies relevant images to the question and answers; **"The current image shows a blue couch in the center. To the left, a staircase is visible. Beyond the couch is a door that is closed."**.

D.2 Home Environment (b)

Question 1: *What is the color of the dehumidifier?*
A. Blue
B. White and Gray
Answer: **B**

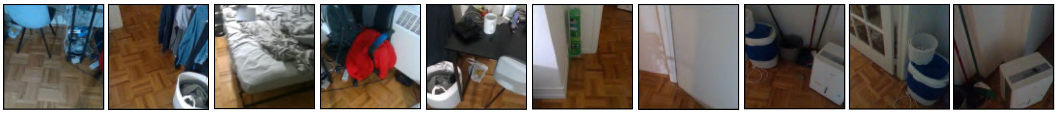


Figure 13: **Home Environment (b) Question 1: What is the color of the dehumidifier?** The agent takes a total of nine VLM steps to answer this question, and starts by exploring a frontier node; "I need to find the dehumidifier machine to determine its color. It is not currently in the scene graph or visible." The next eight steps are chosen as frontier steps, with explanations like "The dehumidifier is not in the scene graph. I need to explore to find it. The current image shows a potential candidate object (white appliance) near the chair (object3) and backpack (object17)." The final frontier step taken reveals the dehumidifier; "The object potentially representing the dehumidifier (object26, labeled 'box') is in room0. There are two white, boxy machines on the floor, likely dehumidifiers or air purifiers. One has a gray top panel."

Question 2: *Is there any object on the black plastic drawer?*
A. Yes
B. No
Answer: **B**

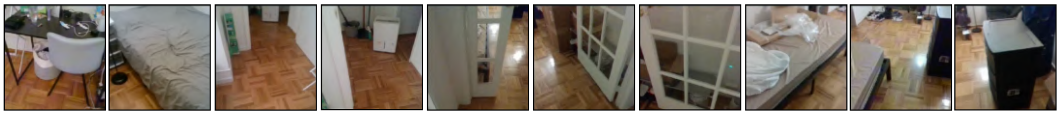


Figure 14: **Home Environment (b) Question 2: Is there any object on the black plastic drawer?** The agent takes a total of nine VLM steps. The first step taken is a frontier node step chosen due to its connection to a candidate object; "I need to explore the environment to find the 'black plastic drawer' mentioned in the question, as it is not currently in the scene graph. The image shows a potential candidate object near the 'chair' (object1). Frontier3 is linked to the 'chair' (object1) and appears to be in the direction of the area needing investigation based on the current view." The following six steps are GotoFrontierNodeStep actions with explanations similar to "This object is not currently in the scene graph or visible. Exploring a frontier might reveal new areas and objects, potentially including the drawer." Finally, the agent is able to visually confirm the presence of the black plastic drawer with high confidence.

Question 3: *What is next to the white shopping bag?*
A. Stool
B. Broom
C. Dehumidifier
Answer: **A**

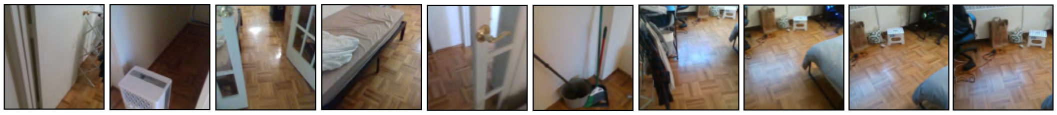


Figure 15: **Home Environment (b) Question 3: What is next to the white shopping bag?** The agent takes a total of four VLM steps in this experiment, all of which are exploratory GotoFrontierNodeStep actions until the stool is identified as being next to the shopping bag.

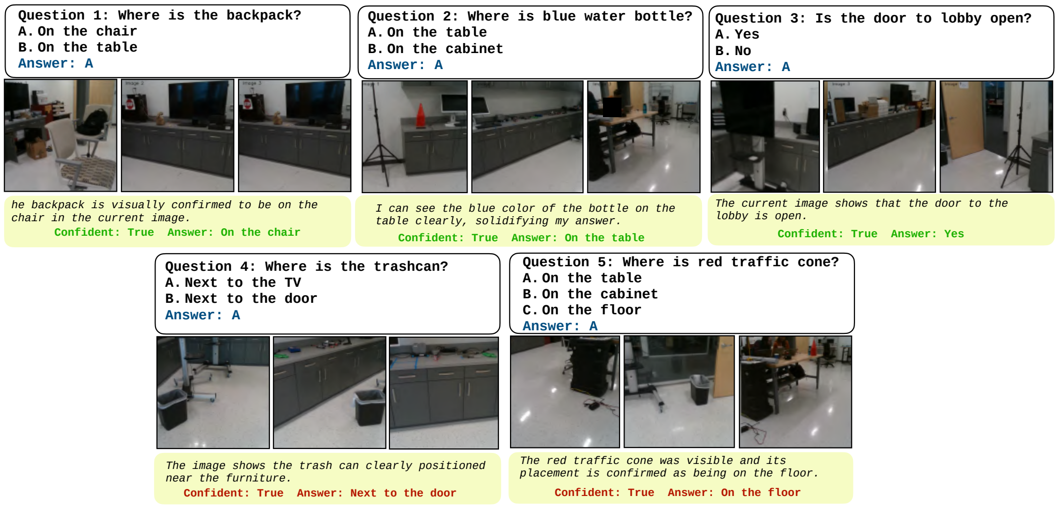
E Additional Real-world Experiments in Office and Workshop Environments

Additional experiments in the workshop and office environment are shown in Figure 16 and Figure 17 respectively. They include both success and failure cases.

E.1 Workshop environment

Question 1: *Where is the backpack?*
A. On the chair
B. On the table
Answer: **A**

The relevant images for the task are shown in Figure 16. 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. 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 1: Where is the backpack?
A. On the chair
B. On the table
Answer: A

he backpack is visually confirmed to be on the chair in the current image.
Confident: True Answer: On the chair

Question 2: Where is blue water bottle?
A. On the table
B. On the cabinet
Answer: A

I can see the blue color of the bottle on the table clearly, solidifying my answer.
Confident: True Answer: On the table

Question 3: Is the door to lobby open?
A. Yes
B. No
Answer: A

The current image shows that the door to the lobby is open.
Confident: True Answer: Yes

Question 4: Where is the trashcan?
A. Next to the TV
B. Next to the door
Answer: A

The image shows the trash can clearly positioned near the furniture.
Confident: True Answer: Next to the door

Question 5: Where is red traffic cone?
A. On the table
B. On the cabinet
C. On the floor
Answer: A

The red traffic cone was visible and its placement is confirmed as being on the floor.
Confident: True Answer: On the floor

Figure 16: **Workshop environment:** Each set of images is from the head camera on the Stretch robot, and represents the task-relevant images at the last planning step. Provided under the images are the answers, confidence levels, and explanations output from the VLM planner.

Question 2: *Where is the blue water bottle?*
A. On the table
B. On the cabinet
C. On the floor
Answer: **B**

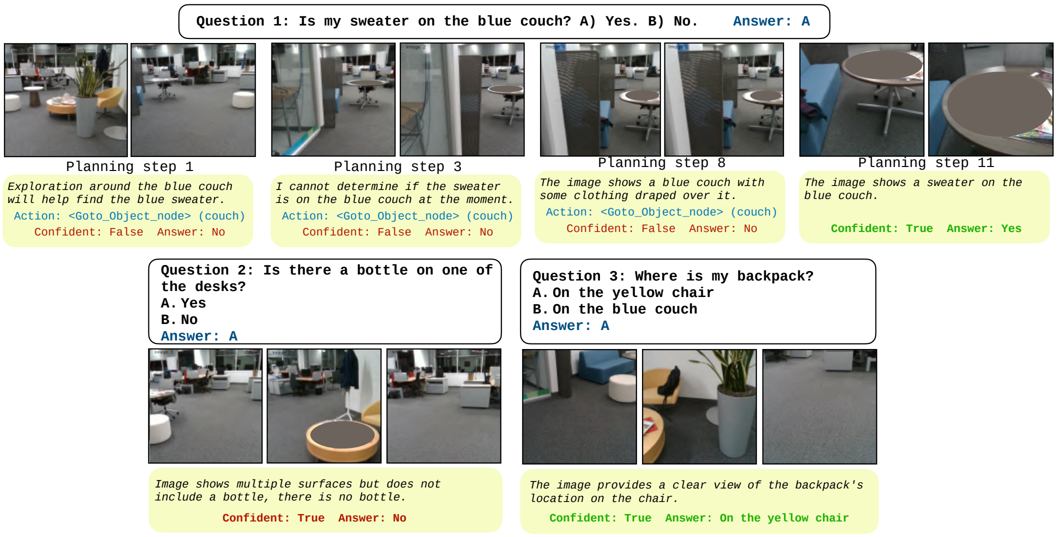
As shown in Figure 16, after exploring the environment with one `<Goto_Object_node>(table)` step, the agent successfully finds the water bottle and confirms its location, providing the following justification for its answer: "The image shows a table with some objects on top, including a blue water bottle. There is also a computer monitor and various tools visible on a countertop”. Qualitative results for additional questions are provided in Figure 16.

E.2 Office environment

Question 1: *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. Qualitative results for additional questions are provided in Figure 17.


Question 1: Is my sweater on the blue couch? A) Yes. B) No. Answer: A

Exploration around the blue couch will help find the blue sweater.
Action: <Goto_Object_node> (couch)
Confident: False Answer: No

I cannot determine if the sweater is on the blue couch at the moment.
Action: <Goto_Object_node> (couch)
Confident: False Answer: No

The image shows a blue couch with some clothing draped over it.
Action: <Goto_Object_node> (couch)
Confident: False Answer: No

The image shows a sweater on the blue couch.
Confident: True Answer: Yes

Question 2: Is there a bottle on one of the desks?
A. Yes
B. No
Answer: A

Image shows multiple surfaces but does not include a bottle, there is no bottle.
Confident: True Answer: No

Question 3: Where is my backpack?
A. On the yellow chair
B. On the blue couch
Answer: A

The image provides a clear view of the backpack's location on the chair.
Confident: True Answer: On the yellow chair

Figure 17: **Office environment:** Each set of images for Question 1 is from the head camera on the Stretch robot, and represents the top-k task-relevant images at each planning step. For questions 2 and 3, the images correspond to the relevant images at the last planning step. Provided under the images are the answers, confidence levels, and explanations output from the VLM planner.