

7 Appendix

7.1 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 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.

7.2 HM-EQA and OpenEQA Datasets

The Habitat-Matterport Embodied Question Answering (HM-EQA) dataset introduced by Yadav et al. [14] is based in the Habitat-Matterport 3D Research Dataset (HM3D) of photo-realistic, diverse indoor 3D scans [62]. 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. The OpenEQA dataset [13] contains data spanning the categories spatial understanding, object state recognition, functional reasoning, attribute recognition, world knowledge, object localization, and object recognition. We use both HM-EQA and OpenEQA to benchmark our results against baselines in simulation.

7.3 Hydra 3D Scene Graphs

Hydra 3DSGs are comprised of the following layers: *Layer 1 (bottom)*: a metric-semantic 3D mesh, *Layer 2*: objects 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 from the robot’s camera, camera pose and camera intrinsics. Using off-the-shelf image segmentation models [61], the object nodes are assigned semantic object labels.

7.4 Prompting

7.4.1 GPT Prompt

The full prompt provided to GPT4o for GraphEQA is given in Section 7.12.2. 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.

7.4.2 Hierarchical Nature of 3DSGs and Planning

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

7.5 Baseline Performance on Task Categories

Table 3 shows the performance of baselines and GraphEQA across the different task categories in the HM-EQA dataset. GraphEQA outperforms all other methods across all task categories, but is

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

602 particularly more performant in comparison when considering Counting and State tasks. It is worth
603 noting that the Counting and State categories are among the most challenging.

604 7.6 Additional Ablations:

605 We perform some additional ablations to evaluate the utility of different components of our method:
606 **GraphEQA-NoEnrich**, which does not use frontier enrichment (Section 3.3), and **GraphEQA-**
607 **CurrView**, which uses only the current view as input to the VLM and does not choose additional
608 task-relevant keyframes (Section 3.4). All ablations of GraphEQA use GPT-4o. Here we analyze
609 these two additional ablations, **GraphEQA-NoEnrich** and **GraphEQA-CurrView**. We observe
610 that GraphEQA-NoEnrich performs slightly worse than GraphEQA which demonstrates that en-
611 riching the scene graph with additional semantic information in the form of edges between frontiers
612 and nearest objects, as discussed in Section 3.3, lends itself to semantically informed exploration.
613 We observe that the performance drop is worse in the case of GraphEQA-CurrView, where we do
614 not use task-relevant visual memory, but only the current view of the agent. This demonstrates that
615 task-relevant visual memory is very useful in long-horizon tasks where the current view of the robot
616 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

617 7.7 Additional Quantitative results: Zero-shot reasoning for commonsense questions

618 We perform this evaluation to answer the following question: how many questions in the HM-EQA
619 dataset can be answered *correctly* purely based on commonsense reasoning or guessing, without
620 exploring the environment? We aim to perform this analysis to roughly estimate the percentage of
621 false positives that can occur in the HM-EQA dataset using different VLMs. To answer this question
622 we define the following three additional baselines: **EQAZeroShotGPT4o**, **EQAZeroShotPris-**
623 **matic** and **EQAZeroShotGPT4oQuestionOnly**. **EQAZeroShotGPT4o** evaluates the zero-shot
624 performance of GPT-4o when answering an EQA question using the initial image and the question.
625 **EQAZeroShotPrismatic** evaluates the zero-shot performance of the calibrated Prismatic model
626 from Explore-EQA [3] when answering an EQA question using the initial image and the question.
627 **EQAZeroShotQuestionOnly** evaluates the zero-shot performance of GPT-4o when answering an
628 EQA question using only the question. In all the above baselines, no exploration steps are taken.
629 Prompts for the above baselines are identical to ones used by Explore-EQA [3]. An episode is
630 considered a success if the question is answered correctly and with high confidence (> 0.5).

631 Table 5 shows the simulation results for the baselines mentioned above compared to the baselines
632 discussed in Section 4.1. We observe that, given only the question, **EQAZeroShotQuestionOnly**
633 answers 6.6% of the questions correctly with high confidence. This can be attributed to the VLM
634 answering questions based on commonsense reasoning or even just random guessing, and getting
635 them correct. **EQAZeroShotPrismatic** answers only 1.8% of the questions correctly with a con-
636 fidence > 0.5 . This can be attributed to the fact that the Prismatic VLM is calibrated to avoid

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

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.

7.8 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 7 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 uniformly sample a set of 114 questions from the HM-EQA dataset and manually label the results across the four categories: True Positives, True Negatives, False Positives and False Negatives. The results are shown in Table 6 where each number is a percentage of the total number of questions considered (114).

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 Section 7.5, categorized by question type.

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



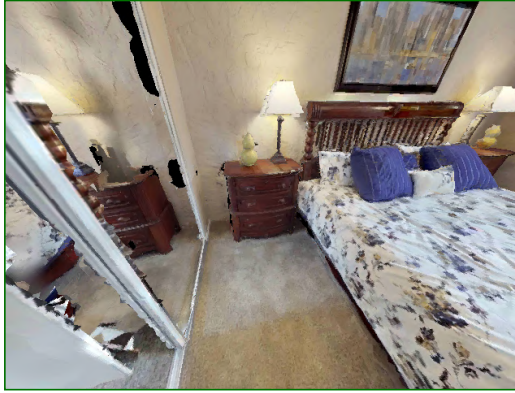
True Positive



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.

667 7.9 Structured Output

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

```

1 def create_planner_response(frontier_node_list, room_node_list,
2   region_node_list, object_node_list, Answer_options):
3
4   class Goto_frontier_node_step(BaseModel):
5       explanation_frontier: str
6       frontier_id: frontier_node_list
7
8   class Goto_object_node_step(BaseModel):
9       explanation_room: str
10      explanation_obj: str
11      room_id: room_node_list
12      object_id: object_node_list
13
14   class Answer(BaseModel):
15       explanation_ans: str
16       answer: Answer_options
17       explanation_conf: str
18       confidence_level: float
19       is_confident: bool
20
21   class PlannerResponse(BaseModel):
22       steps: List[Union[Goto_object_node_step,
23                        Goto_frontier_node_step]]
24       answer: Answer
25       image_description: str
26       scene_graph_description: str
27
28   return PlannerResponse

```

Code Listing 1: The create_planner_response function used to structure output from GPT4o.

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

677 7.10 Additional Real-world Experiments in Home Environments

678 Additional experiments for Home Environment (a) and the questions asked of GraphEQA are shown
 679 below. We provide a sequence of ten images from the head camera on the robot to illustrate explo-
 680 ration and validation of GraphEQA’s answer to the question. Each experiment for Home Environ-
 681 ment (a) was executed twice successfully. We show a single trial from this set for each experiment.

682 7.10.1 Home Environment (a)

Question 1: *Is there a blue pan on the stove?*

A. Yes

B. No

Answer: A



Figure 7: **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. To see the full experiment in video, please see grapheqa.github.io.

Question 2: *How many white cushions are there on the grey couch?*

A. 1

B. 2

C. 3

D. 4

Answer: B



Figure 8: **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.**” To see the full experiment in video, please see grapheqa.github.io.

Question 3: *Where is my handbag?*

A. On the coat rack

B. On the floor

C. On the couch

Answer: A

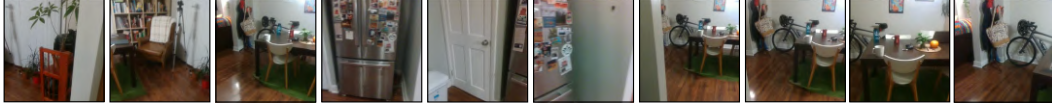


Figure 9: **Home Environment (a) Question 3: Where is my handbag?** The agent takes a total of three VLM steps to determine the location of the handbag, beginning with exploration guided by the scene graph; “The scene graph shows several frontiers connected to objects or areas, but none seem directly 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 grapheqa.github.io.

Question 4: *Where is the trashcan?*

- A. Next to the sink
- B. Next to the refrigerator

Answer: A

686

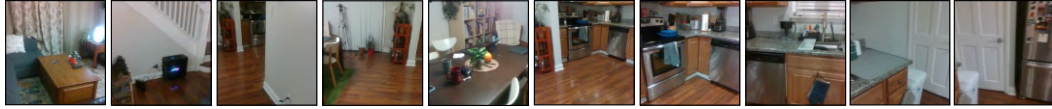


Figure 10: **Home Environment (a) Question 4: Where is the trashcan?** The agent takes a total of six VLM steps to determine the location of the trashcan, beginning with exploration guided by the scene graph; “The image and scene graph don’t provide information about a kitchen or a trashcan location. 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. To see the full experiment in video, please see grapheqa.github.io.

Question 5: *Is the front door next to the staircase open?*

- A. Yes
- B. No

Answer: B

687

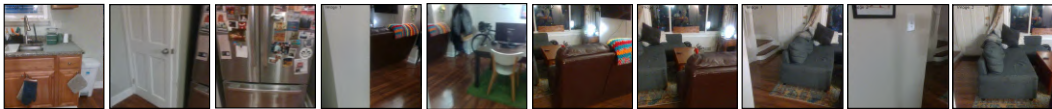


Figure 11: **Home Environment (a) Question 5: Is the front door next to the staircase open?** The agent takes a total of four VLM steps to determine the state of the door, beginning its exploration by identifying where it is located and choosing an action appropriately; “The scene graph shows the agent’s 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 potentially 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.” Finally, 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. Just beyond the couch is a door that is closed.”**. To see the full experiment in video, please see grapheqa.github.io.

688 **7.10.2 Home Environment (b)**

Question 1: *What is the color of the dehumidifier?*

A. Blue

B. White and Gray

Answer: **B**

689



Figure 12: **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.” To see the full experiment in video, please see grapheqa.github.io.

Question 2: *Is there any object on the black plastic drawer?*

A. Yes

B. No

Answer: **B**

690

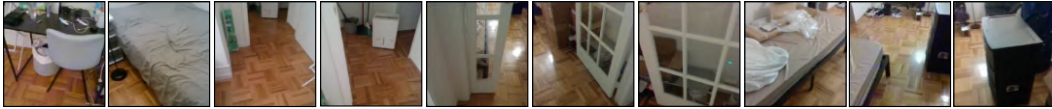


Figure 13: **Home Environment (b) Question 2: Is there any object on the black plastic drawer?** The agent takes a total of nine VLM steps. The first step taken is a frontier node step chosen due to its connection to a candidate object; “I need to explore the environment to find the 'black plastic drawer' 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. To see the full experiment in video, please see grapheqa.github.io.

Question 3: *What is next to the white shopping bag?*

A. Stool

B. Broom

C. Dehumidifier

Answer: **A**

691

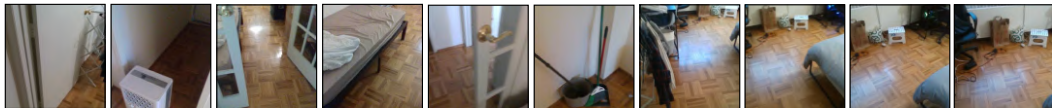


Figure 14: **Home Environment (b) Question 3: What is next to the white shopping bag?** The agent takes a total of four VLM steps in this experiment, all of which are exploratory GotoFrontierNodeStep actions until the stool is identified as being next to the shopping bag.

692 7.11 Additional Real-world Experiments in Office and Workshop Environments

693 Additional experiments in the office environment are shown in Figure 16 and Figure 17. They include both success and failure cases. For videos of additional experiments in the home environment, 694 please refer to the [website](#). 695

696 7.11.1 Qualitative Analysis

697 Workshop environment

698 GraphEQA was provided with the following multiple choice question:

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

699

700 The agent first takes a `<Goto_Object_node>(chair)` step in the environment after an initial rotate-
701 in-place mapping operation to populate the scene graph, choosing to investigate a chair found in
702 the environment. An explanation for this choice is provided by the VLM: “Objects like the chair
703 or table might have the backpack, so checking close to these areas is essential.” The robot then
704 begins navigating to the chair to determine if the backpack is located there. During execution of
705 the trajectory toward the chair, GraphEQA leverages its task-relevant visual memory to score three
706 images encountered on its way to the chair. These images are shown in Figure 15.



Figure 15: Task-relevant images selected by GraphEQA for the task “Where is the backpack?”.

707 After finishing the execution of this trajectory, GraphEQA answers the question with ‘On the chair’
708 and provides the following explanation of its answer: “The backpack is visually confirmed to be on
709 the chair in the current image. The presence of the backpack on the chair makes it clear that the
710 correct answer to the question is ‘On the chair’.”

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

711

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

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

Question: *Where is the blue water bottle?*

- A. On the table
- B. On the cabinet
- C. On the floor

Answer: **B**

721

722 After exploring the environment with one `<Goto_Object_node>(cabinet)` step, see Figure 4, the
723 agent successfully finds the water bottle and confirms its location, providing the following justifica-
724 tion for its answer: “The image shows a cabinet with some objects on top, including a blue water
725 bottle. There is also a computer monitor and various tools visible on the countertop.”

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

Question: *Is my sweater on the blue couch?*

- A. Yes
- B. No Answer. **A**

727

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

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

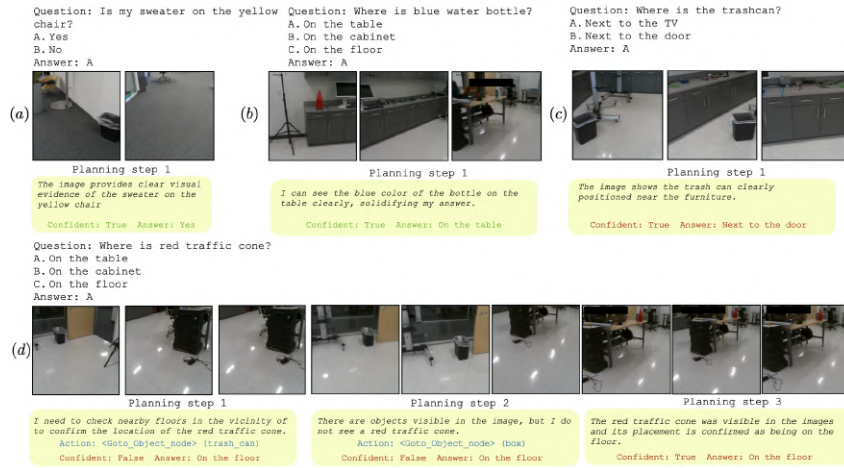


Figure 16: Additional experiments from deploying GraphEQA on the Hello Robot Stretch RE2 platform in a workshop environment (b, c, d) and in an office (a). (c) and (d) are failure cases. Each set of images is from the head camera on the Stretch robot, and represents the top-k task-relevant images at each planning step. Provided under the images are the answers, confidence levels, and explanations output from the VLM planner.

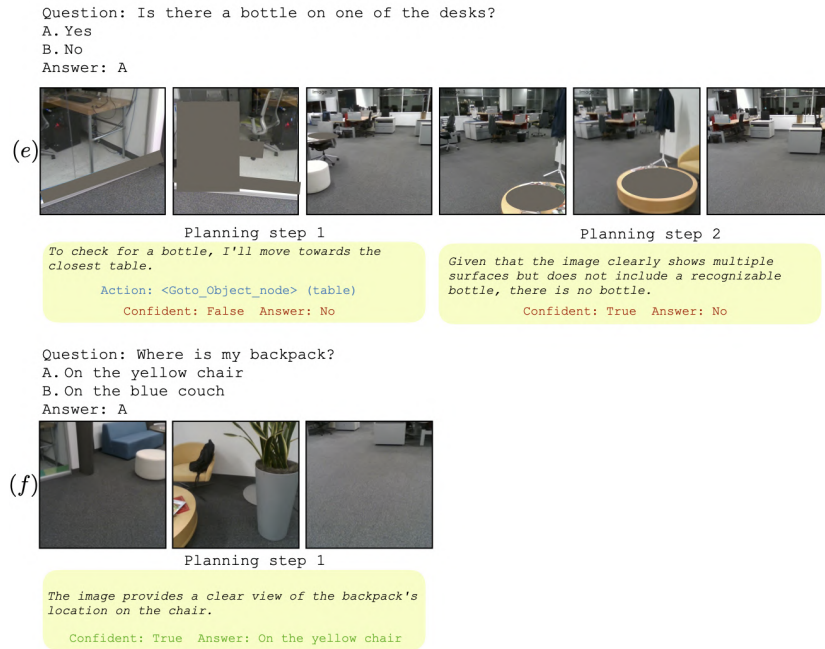


Figure 17: Additional experiments from deploying GraphEQA on the Hello Robot Stretch RE2 platform in an office environment (e, f). (e) is a failure case. Each set of images is from the head camera on the Stretch robot, and represents the top-k task-relevant images at each planning step. Provided under the images are the answers, confidence levels, and explanations output from the VLM planner.

7.12 Qualitative results

7.12.1 Exploration Efficiency and Trajectory Generation

We illustrate the differences in exploration between GraphEQA and Explore-EQA through the following example. Figure 18(a) shows the trajectory taken by an agent employing Explore-EQA in a scene in the HM3D dataset taking 30 VLM steps, while Figure 18(b) shows the trajectory taken by an agent employing GraphEQA in that 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.

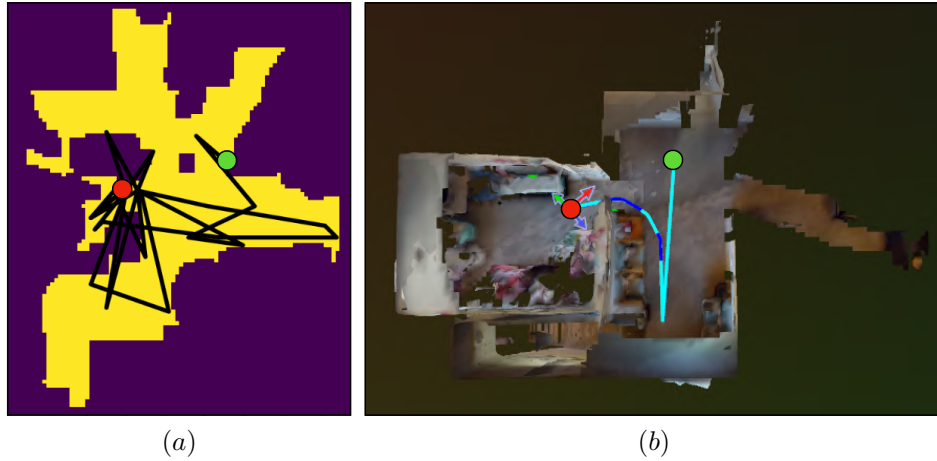


Figure 18: (a) A top down view of the map constructed by Explore-EQA illustrating explored areas for a scene in the HM3D dataset. The agent’s initial position is depicted in green and its final position in red. The trajectory of the agent is shown in black. (b) A top down view of scene reconstruction output by Hydra 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.

744

7.12.2 Choosing object-nodes and frontiers semantically

In 7.12.2 we provide a qualitative examples of GraphEQA planner choosing to go near a ‘couch’ to better answer a question about living room lights. This shows that the planner chooses task-relevant objects to guide exploration. In 7.12.2 we provide a qualitative examples of GraphEQA planner choosing task-relevant frontier nodes and object nodes to guide exploration.

Qualitative results - choosing task-relevant object nodes

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

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

Qualitative results - choosing task-relevant frontier nodes

Question:: Is the kitchen sink currently free of dishes?
A. (Do not choose this option)
B. (Do not choose this option)
C. Yes, it's empty
D. No, there are dishes.
Answer: C

At t=0::
Agent state: The agent is currently at node agent 1 at position [0.9115259647369385, -0.7087370157241821, -1.5000251531600952] at room node: room 0 with name bedroom
Action: explanation frontier="The kitchen typically contains objects like chairs, tables, etc. Frontier 4 is near objects that could indicate a transition to a dining or kitchen area, as it is near the couch and chair, which might lead to a kitchen. It's worth exploring this frontier to find the kitchen." frontier id= frontier 4
Answer: explanation ans="Currently, the kitchen is not visible in the scene. The scene graph and current image provide information about a bedroom, but not a kitchen. Therefore, I can't answer whether the kitchen sink is free of dishes." answer= A: '(Do not choose this option)' explanation conf="I haven't yet gathered any information on the kitchen or the sink from my current location in the bedroom, and no other regions of the building have been explored yet." confidence level=0.0 is confident=False

At t=1::
Agent state: The agent is currently at node agent 6 at position [2.5569498538970947, 2.228832721710205, -1.5000251531600952] at room node: room 0 with name living room
Action: explanation frontier="The frontier 2 is connected to objects like the table, cabinet, fridge, and plates, which might lead to a kitchen area or provide a clue to the sink's status." frontier id=frontier 2
Answer: explanation ans='Currently, there is no visual confirmation of a sink in the image or scene graph, and insufficient detail to determine if the kitchen sink is free of dishes.' answer=A: '(Do not choose this option)' explanation conf="The scene graph and current image give no direct view of a kitchen sink or its status, so it's uncertain whether it's free of dishes." confidence level=0.2 is confident=False

At t=2::
Agent state: The agent is currently at node agent 11 at position [2.6420648097991943, -0.10468387603759766, -1.5000251531600952] at room node: room 0 with name living room
Action: explanation room='To determine if the kitchen sink is free of dishes, I need to transition from the living room to the kitchen area, where the sink is likely located. Frontiers associated with kitchen-related objects can guide me there.' explanation obj="To check the sink, moving towards the 'plates' near object 41 might lead to the kitchen area." room id=room 0: 'living room' object id=object 41: 'plates'
Answer: explanation ans="Currently, I don't have enough information to directly answer if the kitchen sink is free of dishes, based on the image or scene graph. Further exploration is needed to confirm the state of the sink." answer=A: '(Do not choose this option)' explanation conf="Given my current location and the items visible, I can't see the sink or its contents. I need to navigate closer to where I suspect the sink might be, based on kitchen-related objects like plates." confidence level=0.3 is confident=False

At t=3::
Agent state:The agent is currently at node agent 21 at position [1.7573769092559814, -0.35986196994781494, -1.5000251531600952] at room node: room 0 with name living room
Answer: explanation ans='The image shows the kitchen, and the sink area is visible. There are no dishes in the sink, indicating it is empty.' answer=C: "Yes, it's empty" explanation conf='The image clearly shows the sink, and there are no dishes visible, providing clear evidence to confidently answer the question.' confidence level=1.0 is confident=True

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

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 depicting which room they are located in. At the lowest level 2 are object nodes and agent nodes. There is an edge from region node to each object node depicting which visited area of which room the object is located in. There are also links between frontier nodes and objects nodes, depicting the objects in the vicinity of a frontier node. Finally the agent node is where you are located in the environment. There is an edge between a region node and the agent node, depicting which visited area of which room the agent is located in.

Current state description: CURRENT STATE will give you the exact location of the agent in the scene graph by giving you the agent node id, location, room_id and room name.

General Description: Given the current state information, try to answer the question. Explain the reasoning for selecting the answer. Finally, report whether you are confident in answering the question. Explain the reasoning behind the confidence level of your answer. Rate your level of confidence. Provide a value between 0 and 1; 0 for not confident at all and 1 for absolutely certain. Do not use just commonsense knowledge to decide confidence. Choose TRUE, if you have explored enough and are certain about answering the question correctly and no further exploration will help you answer the question better. Choose 'FALSE', if you are uncertain of the answer and should explore more to ground your answer in the current environment. Clarification: This is not your confidence in choosing the next action, but your confidence in answering the question correctly. If you are unable to answer the question with high confidence, and need more information to answer the question, then you can take two kinds of steps in the environment: Goto_object_node_step or Goto_frontier_node_step. You also have to choose the next action, one which will enable you to answer the question better. Goto_object_node_step: Navigates near a certain object in the scene graph. Choose this action to get a good view of the region around this object, if you think going near this object will help you answer the question better. Important to note, the scene contains incomplete information about the environment (objects maybe missing, relationships might be unclear), so it is useful to go near relevant objects to get a better view to answer the question. Use a scene graph as an imperfect guide to lead you to relevant regions to inspect. Choose the object in a hierarchical manner by first reasoning about which room you should goto to best answer the question, and then choose the specific object. Goto_frontier_node_step: If you think that using action "Goto_object_node_step" is not useful, in other words, if you think that going near any of the object nodes in the current scene graph will not provide you with any useful information to answer the question better, then choose this action. This action will navigate you to a frontier (unexplored) region of the environment and will provide you information about new objects/rooms not yet in the scene graph. It will expand the scene graph. Choose this frontier based on the objects connected this frontier, in other words, Goto the frontier near which you see objects that are useful for answering the question or seem useful as a good exploration direction. Explain reasoning for choosing this frontier, by listing the list of objects (<id> and <name>) connected to this frontier node via a link (refer to scene graph). While choosing either of the above actions, play close attention to 'HISTORY' especially the previous 'Action's to see if you have taken the same action at previous time steps. Avoid taking the same actions you have taken before. Describe the CURRENT IMAGE. Pay special attention to features that can help answer the question or select future actions. Describe the SCENE GRAPH. Pay special attention to features that can help answer the question or select future actions.'

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