Explore with Long-term Memory: A Benchmark and Multimodal LLM-based Reinforcement Learning Framework for Embodied Exploration

Supplementary Material

1. Full-set Evaluation

Due to resource constraints, we used approximately 35% of the test set (58/166) for comparison in the main text. To fully illustrate the superiority of our method, we present the comparison results of existing embodied exploration methods on the full LMEE-Bench in Tab. 1. Due to the limitation of inference speed, completing all 166 tasks requires a substantial amount of time. The experimental conclusions show no significant difference between the subset and the full test set, demonstrating that the subset is sufficient for accurately evaluating the model.

We also evaluate the answer quality across different question types. As shown in Fig. 1, we test MemoryExplore on both the subset and the full dataset, and the distribution of performance across question types remains largely consistent, further demonstrating the generalizability of the subset results. In addition, for counting and relational questions, there is a noticeable performance gap between open-ended answers and multiple-choice answers, indicating that large models still struggle with certain challenging open-ended question types.

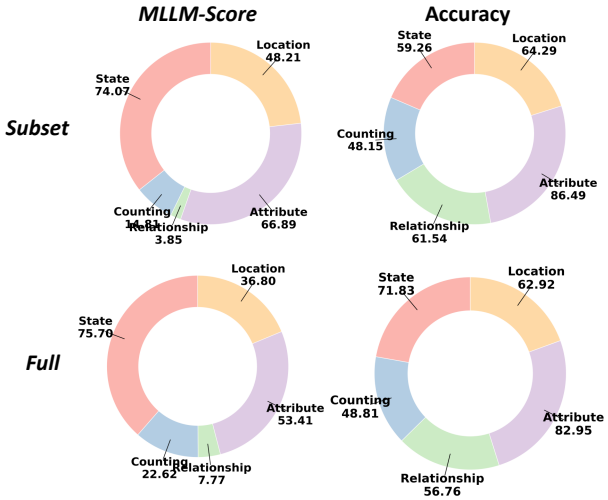


Figure 1. Answer quality across different question types.

2. Ablation Study

First, we provide additional experiments to verify the effectiveness of the training task design. As shown in Tab. 2, supervising the agent only on autonomous navigation does not lead to a clear performance improvement. Once the memory-retrieval tool is introduced, the model achieves a significant performance gain, demonstrating that learning active retrieval is crucial for improving accuracy in long-horizon navigation and memory-based question answering.

Second, we conduct ablation studies on the reward design. Our proposed multi-task reward consists of an action–frontier consistency penalty and a tool-usage penalty. As shown in Tab. 3, incorporating these penalties leads to improved model performance.

Finally, Tab. 4 presents additional ablation experiments on the training hyperparameter settings.

3. Real World Testing

To verify the sim-to-real generalization and practical applicability of our MemoryExplorer agent, we deployed it on a physical robotic platform. This section details our experimental setup and presents the qualitative results from tests conducted in real-world, unstructured office environments. Our goal was to demonstrate that the core long-term memory mechanism can be effectively transferred from simulation to reality. The experiments were performed on a ROSMASTER X3 robot as shown in Fig. 2, which uses an Orbbec Astra Pro depth camera as its primary visual sensor. Our system architecture involved the robot, a local computer, and a remote server with an NVIDIA H200 GPU.

Figure 2. ROSMASTER X3.

We evaluate the agent's ability to perform multi-goal navigation and utilize its memory in two different office environments (a meeting room and a reception room). In the

Table 1. Experiments on subset and full-set LMEE-Bench.

Method	Easy				Medium				Hard				Total				Time
	Nav		QA		Nav		QA		Nav		QA		Nav		QA		
	SR	SPL	Score	Acc	SR	SPL	Score	Acc	SR	SPL	Score	Acc	SR	SPL	Score	Acc	
Subset																	
Explore-EQA [2]	4.26	4.26	-	-	15.17	8.67	-	-	14.89	7.25	-	-	13.24	7.66	-	-	2h
3D-Mem [4]	21.28	9.68	30.71	42.86	15.73	6.09	34.57	44.68	17.02	6.97	25.00	18.75	16.91	6.86	32.59	41.38	17h
RA-Mem	25.53	15.65	34.29	54.29	**21.35**	12.14	37.77	62.77	14.89	8.86	25.00	43.75	20.96	12.18	35.52	58.62	10h
MemoryExplorer (Ours)	**31.91**	**21.11**	**35.71**	**68.57**	**21.35**	**14.03**	**48.14**	**63.83**	**23.40**	**12.51**	**34.38**	**68.75**	**23.53**	**14.99**	**43.62**	**65.52**	10h
Full																	
Explore-EQA [2]	9.40	3.44	-	-	10.02	5.78	-	-	7.59	5.12	-	-	9.50	5.33	-	-	8h
3D-Mem [4]	29.06	15.43	30.94	37.50	16.10	7.05	32.35	38.60	14.48	6.38	22.69	44.44	17.66	8.13	30.79	39.16	49h
RA-Mem	29.91	16.66	**40.62**	53.75	20.04	11.10	34.47	56.25	14.48	8.51	27.31	51.85	20.46	11.44	34.73	55.17	29h
MemoryExplorer (Ours)	**33.33**	**20.62**	37.50	**66.25**	**20.39**	**13.99**	**41.18**	**64.34**	**19.31**	**10.21**	**30.09**	**64.81**	**22.05**	**14.26**	**38.98**	**64.78**	29h

Table 2. Ablation study on training task setting.

	LMEE-Bench			
	SR	SPL	Score	Acc
baseline	20.96	12.18	35.52	58.62
only nav	20.59	12.34	39.14	58.62
w/ memory tool	**23.53**	**14.99**	**43.62**	**65.52**

Table 3. Ablation study on reward design.

	LMEE-Bench			
	SR	SPL	Score	Acc
MemoryExplorer	**23.53**	**14.99**	**43.62**	**65.52**
w/o consistency penalty c	22.43	13.76	41.38	**65.52**
w/o tool-usage penalty α	21.32	12.79	41.03	64.14

Table 4. Ablation study on hyperparameters.

w_{act}	w_{front}	w_{ans}	w_{fmt}	LMEE-Bench			
				SR	SPL	Score	Acc
0.2	0.2	0.4	0.2	**23.53**	**14.99**	**43.62**	65.52
0.3	0.3	0.3	0.1	22.43	14.65	42.76	**66.90**
0.4	0.4	0.1	0.1	22.43	12.43	42.24	64.83

first case, conducted in a meeting room, the task instruction is: "*Start in the meeting room to find a bottle of water, then look for a rubbish bin.*" After completing the navigation task, we construct the memory bank from the observation images and then perform memory-based question answering. When asked "*Where is the bottle of water?*", MemoryExplorer retrieves the relevant memory and responds: "*The bottle of water is on the chair.*"

In the second example, the task instruction is: "*Start in the reception room to find a rubbish bin, a potted cactus, and an umbrella.*" After finishing navigation, we intentionally ask a question unrelated to the navigation targets: "Where is the tripod? Please describe its location in detail." MemoryExplorer generates an appropriate retrieval query and accurately answers: "*The tripod is in the corner of the room, near a wall with a blue triangle on it.*"

In summary, our real-world experiments demonstrate the robustness and generalization ability of MemoryExplorer, highlighting the strong coupling between cognition and decision-making that lies at the core of our approach.

4. Illustration of the Full Task Process

In Fig. 4, we present the content of a complete task, which consists of five navigation goals and two memory-based QA tasks. The agent successfully locates the refrigerator, coffee machine, and nightstand, but fails to find the dresser and picture due to incorrect memory retrieval. When searching for the picture, the agent retrieves the picture that appears in its memory, locates its position, and approaches it, but the target it is looking for is not the picture in its memory. For the memory-based QA tasks, the first question about the coffee machine is answered incorrectly because the retrieved memory is inaccurate. In contrast, the second question selects the correct memory entry, enabling the agent to produce the correct answer.

5. Failure Case Analysis

In addition to common navigation failures (such as selecting the wrong object in the memory or exceeding the maximum exploration steps), we further analyze the causes of memory-based question answering failures. First, ambiguities that inevitably arise during data generation, as in Fig. 5, may cause the model to retrieve the correct memory but still produce an incorrect answer. Second, due to the spatial understanding limitations of MLLMs, the agent may retrieve the wrong memory, as illustrated in Fig. 6, or generate an incorrect description even when the correct memory is retrieved, as shown in Fig. 7.

6. Data Construction Details

The HM3DSem includes labels for objects and their corresponding regions. We utilize the room names correspond-

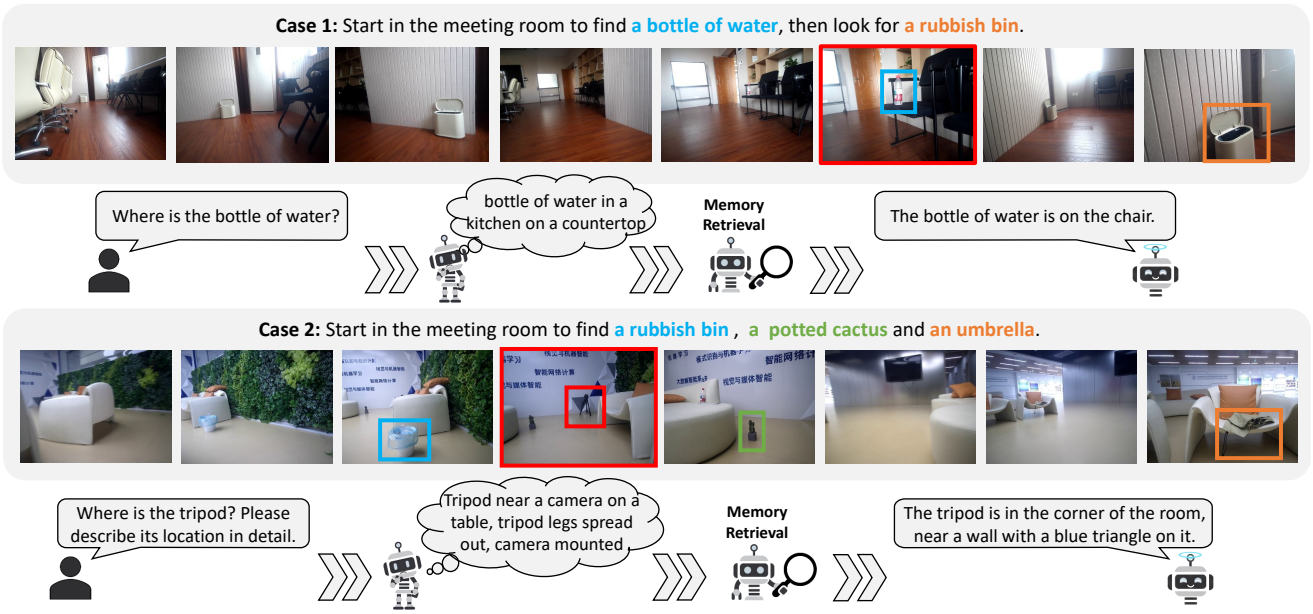


Figure 3. Real-world testing.

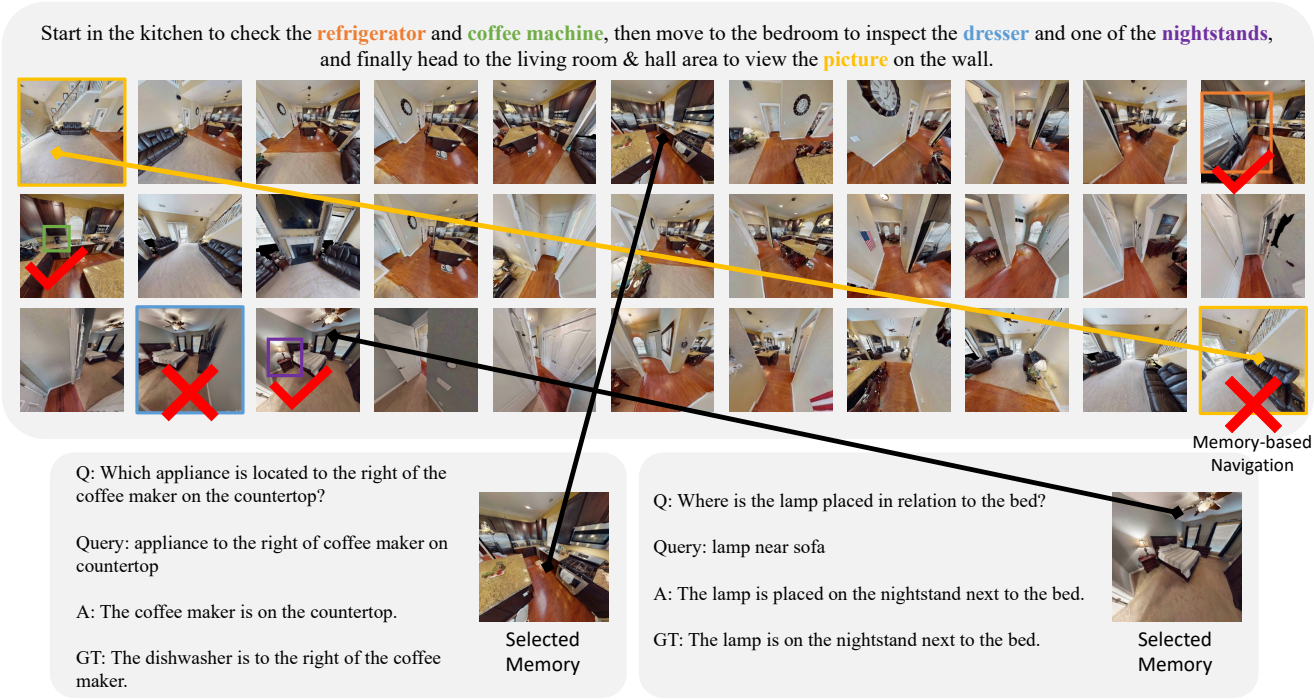
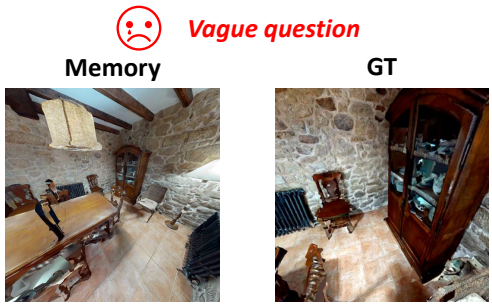


Figure 4. Complete task. Task is executed in a left-to-right, top-to-bottom order.

ing to each region provided in [3], such as bedroom, bathroom, etc. Finally, we input the object and its corresponding room information into LLM (Qwen3-235B-A22B-Instruct) and generate a multi-object navigation task based on the prompts in Fig. 8. Since the illusion problem in LLM can generate non-existent or incorrect objects, we automatically filter out these erroneous task instructions when generating trajectories using Habitat-sim. Then, we input the observation images of successfully navigated targets in the trajectory into VLM (Qwen3-VL-235B-A22B-Instruct) to gener-



Question: Where is the wooden chair positioned in relation to the stone wall?

Ground Truth: The wooden chair is placed directly against the stone wall.

Prediction: The wooden chair is to the right of the stone wall.

Figure 5. Failure cases.

Question: Where is the decorative wall piece located relative to the vase with flowers?

Ground Truth: The decorative wall piece is mounted above the vase with flowers.

Prediction: The decorative wall piece is to the right of the vase with flowers.

Figure 6. Failure cases.

ate question-answer pairs, using prompts as shown in Fig. 9. Tab. 5 shows the specific statistical information of the trajectories and targets in our constructed dataset.

Training Sample Construction. Since the trajectory data contains detailed information for each step, we designed a training sample construction Algorithm 1 based on multimodal information. A task includes multiple trajectories corresponding to navigation trajectories for multiple target objects. The memory bank includes images, text, and location information. Since text generated based on an image tagging model is inaccurate, and location information is limited, similarity calculation mainly relies on image information. We set ω_o, ω_f, and ω_p to be 0.5, 0.3, and 0.2, respectively. However, memory updates based on similarity filtering found that they could not accurately collect goal-related observation images, *i.e.*, correct memories. Therefore, we forcibly insert goal-related memories into each trajectory to ensure correct memory retrieval.

Question: How many sinks are visible in the kitchen?

Ground Truth: There is one sink under the window in the kitchen.

Prediction: There are two sinks visible in the kitchen.

Figure 7. Failure cases.

We use a 20-step action sampling interval to avoid high sample repetition, and a 10-step memory sampling interval to reduce the computational cost of memory retrieval during training. We calculate the mean and standard deviation of the similarity of the 10 most recent samples and dynamically filter context memories in each step based on an adaptive similarity threshold. The continuous action window is 6 steps to ensure action coherence. Finally, 11,684 samples are obtained as training data.

7. Training Details

We use EasyR1, a simplified version of the Verl framework. The specific training hyperparameters are shown in Tab. 6. The training format prompt is shown in Fig. 10. We use 8 NVIDIA H200 GPUs for training, which takes approximately 60 hours.

8. Experimental Details

3D-Mem is an embodied exploration method based on a multimodal large language model. It constructs a 3D memory bank by collecting multi-view observations. When performing embodied tasks, the agent builds front-end snapshots based on observations and depth images for explo-

ration, and saves memory snapshots to find objects. Once the target object is confirmed in the memory snapshot, the agent stops running. The memory bank includes observed images and their corresponding object categories and masks. Due to the limitations of the MLLM context window, not all memory information can be input into MLLM simultaneously. It mitigates this problem by using object category-based relevance to filter memories.

RA-Mem is our embodied exploration method for actively retrieving memories, developed based on 3D-Mem [4]. The query prompt in Fig. 11 is input into MLLM to generate query text, and then retrieve the most relevant memories using feature similarity matching to help the model navigate and perform embodied question answering. This effectively improved model performance and reduced task completion time as shown in Tab. 1.

MemoryExplorer builds upon the RA-Mem method, which utilizes only MLLM for inference, by introducing reinforcement learning fine-tuning. This allows for end-to-end training of an MLLM, enhancing its active memory retrieval and exploration capabilities.

Embodied Exploration. Similar to Goat-Bench [1], an LMEE task consists of multiple subtasks. We define each subtask type as instance-level text description, image, and question-and-answer: "The complete task is: {task_instruct} Now you need to perform the subtask of finding the {goal_name}, which is exactly described as the {lang_desc}", "The complete task is: {task_instruct} Now you need to perform the subtask of finding the exact {goal_name}, which is captured at the center of the following image? You need to pay attention to the environment and find the exact object.", and "{question}". The agent first executes the multi-goal navigation task with navigation prompt as shown in Fig. 12, where each subtask includes the overall task instructions and task type description to drive the task progress. Then, the agent executes memory-based question-and-answer, using memory retrieval to answer the given questions with a question answering prompt as shown in Fig. 13.

Most of our exploration settings follow 3D-Mem [4], the frontier-based exploration framework is built upon the Explore-EQA [2], we maintain a 3D voxel occupancy map (0.1 m resolution) and update free space using depth observations and camera poses. The navigable region is defined as the free-voxel slice at 0.4 m height. Areas within 1.7 m of the agent's trajectory are treated as explored; the rest remain unexplored. Frontiers are formed by clustering pixels in the unexplored region using Density-Based Spatial Clustering of Applications with Noise (DBSCAN). A frontier $F = (r, p, I_{obs})$ includes the pixel cluster r, a navigable boundary point p, and an observation I_{obs}. We filter out small clusters (< 20 pixels), update frontiers when the IoU with the previous version drops below 0.95, and split wide frontiers ($> 150^\circ$ FOV) via K-means to improve navigation flexibility. Note that this voxel representation does not support multi-floor scenes. For VLM prompting, only image observations are included. If the VLM chooses a frontier F, its associated location p becomes the navigation target. At each time step t, we collect 3 egocentric views with a 60° angular interval. The original views are captured at 1280 × 1280 resolution to improve object detection quality and then resized to 360 × 360 as input candidates for the VLM. Frontier snapshots are directly captured at 360 × 360. We use YOLOv8x-World, implemented by Ultralytics, with a 200-class detection set from ScanNet. Each task is limited to a maximum of 50 steps. And a task completion condition where the agent's location is within 1m of the target object. We set the number of topk retrieved memories to 3, consistent with the training settings, while maintaining the MLLM's inference speed.

9. Limitations

The primary limitation of MLLM-based embodied exploration lies in its slow inference speed, which prevents real-time execution of embodied tasks. Developing more lightweight models will be an important direction for future research. In addition, the results on LMEE-Bench indicate that current methods are not yet able to effectively handle challenging embodied tasks that require long-term memory. Improving the accuracy and efficiency of long-term memory storage and retrieval will be crucial for advancing practical deployment.

References

[1] Mukul Khanna, Ram Ramrakhya, Gunjan Chhablani, Sriram Yenamandra, Theophile Gervet, Matthew Chang, Zsolt Kira, Devendra Singh Chaplot, Dhruv Batra, and Roozbeh Mottaghi. Goat-bench: A benchmark for multi-modal lifelong navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16373–16383, 2024. 5

[2] Allen Z Ren, Jaden Clark, Anushri Dixit, Masha Itkina, Anirudha Majumdar, and Dorsa Sadigh. Explore until confident: Efficient exploration for embodied question answering. *arXiv preprint arXiv:2403.15941*, 2024. 2, 5

[3] Xinshuai Song, Weixing Chen, Yang Liu, Weikai Chen, Guanbin Li, and Liang Lin. Towards long-horizon vision-language navigation: Platform, benchmark and method. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 12078–12088, 2025. 3

[4] Yuncong Yang, Han Yang, Jiachen Zhou, Peihao Chen, Hongxin Zhang, Yilun Du, and Chuang Gan. 3d-mem: 3d scene memory for embodied exploration and reasoning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 17294–17303, 2025. 2, 5

Table 5. Overall Statistics for trajectory and goal count across difficulty levels.

	Trajectory Statistics			Goal Statistics						
Difficulty	**Tasks**	**Total Steps**	**Avg**	**Train Total**	**Train Max**	**Train Min**	**Test Total**	**Test Max**	**Test Min**	**Distance**
All	1982	377311	190.37	8880	8	2	828	9	2	1-30m
Easy	764	62961	82.41	2743	7	2	118	5	2	1-5m
Medium	1058	256598	242.53	5315	8	3	563	8	3	5-10m
Hard	160	57752	360.95	822	8	5	147	9	5	10-30m

Algorithm 1: Memory-Augmented Training Data Construction

Input: Task set $\mathcal{D}$, CLIP encoder $\mathcal{E}$, sample interval S, memory interval U, continuous action window W
Output: Training dataset $\mathcal{T}$
foreach *task* $d \in \mathcal{D}$ **do**
  Load instruction, text, trials, and QA pairs;
  Initialize memory bank $\mathcal{M} \leftarrow \varnothing$;
  foreach *trial* k *in order* **do**
    Load position p, get all images and text features o and f using $\mathcal{E}$;
    Select a QA pair from past trials;
    for $i = 1$ *to* T **do**
      `// Dynamic memory update`
      if $i - i_{last\_mem} \geq U$ **then**
        Compute similarity between current memory (p_c, f_c, o_c) and recent memory entries;
        if *Novelty condition satisfied* **then**
          Append new memory entry to memory bank $\mathcal{M}$;
          Update $i_{\text{last\_mem}} \leftarrow i$;
      `// Sample continuous action`
      if *Continuous action window* W *around* i *contains too many distinct actions* **then**
        continue;
      `// Sample training data`
      if $i - i_{sample} < S$ **then**
        continue;
      Build prompt with triplet images, instruction, memory hint, and QA;
      Get next-action label y_i;
      Append sample $(\text{prompt}, \text{images}, \mathcal{M}, y_i, \text{answer})$ to $\mathcal{T}$;
      Update $i_{\text{sample}} \leftarrow i$;
    `// Goal-related memory`
    Add final-step memory entry to $\mathcal{B}$;
return $\mathcal{T}$;

Table 6. Hyperparameters Used in Training

Hyperparameter	**Value**
learning_rate	1e-6
global_batch_size	128
rollout_batch_size	256
temperature	1.0
num_generations	5
num_train_steps	160
max_prompt_length	16384
max_response_length	4096
batch_size_per_device_for_update	4
batch_size_per_device_for_experience	8
KL coefficient	0.1
Top-k	3
consistency coefficient	0.5
Scaling factor	1.2 for success / 0.5, 0.6 for fail
$w_{\text{act}}, w_{\text{front}}, w_{\text{ans}}, w_{\text{fmt}}$	0.2, 0.2, 0.4, 0.2

Task Generation Prompt

You are an embodied AI task generator. Your goal is to create realistic exploration tasks for an intelligent agent based on a given indoor scene. Each task must consist of multiple subtasks involving navigation and observation of objects across one or more rooms.
The task should reflect a coherent exploration process that could realistically occur in an indoor environment.

Task Format

The task you generate should consist of the following subtasks:
"""
Move_to("object_id"): Navigate to a specific object to observe it.
IMPORTANT: "object_id" MUST strictly match one of the object identifiers from the input scene (e.g., 'bed_2', 'cabinet_10').
The object's region must also be consistent with the scene input.
Do not invent new objects or regions.
"""
There are rules you MUST follow:
"""
1. The task instruction should describe a realistic exploration behavior — e.g., searching for, locating, or checking multiple objects across rooms.
2. ONLY use objects that appear in the input scene. Do NOT create new objects or rename them.
3. Generate exactly 3 to 6 subtasks.
4. Each Move_to() must contain a valid object ID that exists in the scene.
5. Subtask order must make logical sense (e.g., visiting nearby or related rooms sequentially).
6. The task instruction should sound conversational and natural, e.g.,
"Check the towel in the bathroom, then go see the bed in the bedroom, and finally inspect the television in the living room."
7. The task may vary in complexity:
- EASY: Subtasks in 1-2 rooms.
- MEDIUM: Subtasks across 3-4 rooms.
- HARD: Subtasks across 4 or more rooms.
8. DO NOT explicitly mention the difficulty level in the output; the complexity will be inferred later.
9. Mention the corresponding room names in the task instruction to maintain realism.
"""

Task Instruction Style

The instruction should:
"""
Describe a goal-driven exploration (e.g., inspecting, searching, verifying, checking).
Mention room names naturally to guide the flow.
Use natural transitions:
- "First, go to the bedroom to check the bed,"
- "then move to the bathroom to look at the mirror,"
- "and finally, visit the kitchen to inspect the stove."
Avoid robotic enumeration, aim for smooth, human-like narrative flow.
"""

Output Format

Your output must be a Python dictionary with the following keys:
"""
```
{
 "Task instruction": "A conversational description of the navigation or observation mission.",
 "Subtask list": [
    "Move_to("object_id")",
    ...
 ]
}
```
"""

<**Examples**>

Figure 8. Task instruction generation prompt.

Question Answering Generation Prompt

You are an assistant that helps evaluate the memory quality of an embodied navigation agent.
The robot has just found a target object in an indoor environment.
You will be shown several images (left, front, right) captured at that moment.
Before generating the question, here are the definitions and **examples for each category**:
1. Location: Questions about the position or placement of objects.
  Example:
    Question: What is the position of the coffee table relative to the sofa?
    Choices: A) In front B) Behind C) Next to D) Far away
    Answer: C
2. Relationship: Questions about the spatial or contextual relationships between objects.
  Example:
    Question: Which object is next to the refrigerator in the kitchen?
    Choices: A) Microwave B) Cabinet C) Oven D) Sink
    Answer: B
3. State: Questions about the current state, condition, or status of an object.
  Example:
    Question: Is the door open or closed?
    Choices: A) Open B) Closed
    Answer: B
4. Attribute: Questions about an object's properties, such as color, shape, or material.
  Example:
    Question: What is the primary color of the cabinet where the television is placed?
    Choices: A) Blue B) White C) Brown D) Black
    Answer: C
5. Counting: Questions about the number of objects or quantity.
  Example:
    Question: How many chairs are around the dining table?
    Choices: A) 2 B) 4 C) 6 D) 8
    Answer: B

Your task:

1. Observe the scene carefully and reason about what is visible. You can describe what you see within <think>...</think>.
2. Then generate ONE multiple-choice question about the scene. The choices section can have 2–4 options depending on the question.
3. The question should test whether the robot remembers the found object and its surroundings.
4. Avoid mentioning "camera" or "view" in the question.
5. The question must have one clear correct answer.
6. The question MUST be about the following category: {category}.
7. After providing the correct choice, also give a **very short open-ended answer** (1 simple sentence) describing the fact, not reasoning.

Output your response in this strict JSON format:

```
{{
 "category": "{category}",
 "question": "...",
 "choices": {{
  "A": "...",
  "B": "...",
  "C": "...",  // optional
  "D": "..."   // optional
 }},
 "answer": "B",
 "open_answer": "..."  // concise factual description, e.g., "The radiator is below the window."
}}
```

Figure 9. Question answering generation prompt.

Training Format Prompt

You are an embodied navigation and reasoning assistant designed to operate in a memory-augmented environment.
Your task is to analyze the current task instruction, the three frontier observations (left, front, right), and optionally retrieve relevant memories to complete the current question or navigation step.
You have access to the following tool from memory_tool.py:

```python
def retrieve_memories(memory, query_text):
    """
    Retrieve relevant past memories.
    Args:
        memory (List[List[Dict]]): The memory list from training data, each memory contains:
            - "img": PIL.Image.Image object
            - "object_text": text description
        query_text (str): text query for retrieval

    Returns:
        List[Dict]: each dictionary contains:
            - "img": PIL.Image.Image object
            - "objects": corresponding text description
    """
```

You can call this tool at any time using:

```python
retrieved_mem = retrieve_memories(memory, query_text="...")
display(retrieved_mem)
```

display loads the retrieved memories into the program so that the agent can analyze them.
The function retrieves relevant memories based on the query and displays them for analysis.

RULES

1. Always reason step by step inside THOUGHT.
2. The TASK INSTRUCTION describes a multi-step task.
3. You are given three frontier observation images in the order: left, front, right.
4. Use retrieve_memories with a query_text.
5. Always write the retrieval as a CODE block with Python syntax, including both retrieve_memories and display(mems).
6. After retrieving memories, based on the task instruction, the observations, and the retrieved memories, provide accurate answer to question in task instruction as ANSWER.
7. Analyze the current task progress, the retrieved memories, and the frontier observations, decide which frontier to go next (0 = left, 1 = front, 2 = right). This is called FRONTIER.
8. Finally, output the ACTION using the valid actions: turn_left, move_forward, or turn_right.

Output format:

THOUGHT 0: Analyze task instruction, question and observations carefully. Identify what the current question is asking.
CODE 0:

```python
retrieved_mem = retrieve_memories(memory, query_text="Write a short query text summarizing what information is needed from the memory")
display(retrieved_mem)
```

OBSERVATION: Execution success with retrieved results.
THOUGHT 1: Analyze the task instructin and the retrieved memories, depending on the question type, provide the appropriate answer. Then analyze the current task progress, the retrieved memories, and the frontier observations (left, front, right). Decide which frontier direction is most relevant for the next action toward completing the task.
ANSWER: For multiple-choice questions, output the letter of the correct option, e.g., “B”
FRONTIER: The frontier index (0 = left, 1 = front, 2 = right).
ACTION: One of [turn_left, move_forward, turn_right].
<Examples>

Figure 10. Training format prompt.

Retrieval Query Prompt

System Prompt

You are an intelligent embodied navigation assistant with a visual memory system. Your goal is to generate a retrieval query that can be used to search a memory database of past visual observations. You will receive a task instruction and possibly one or more images (e.g., frontier images or goal images).

Guidelines for Writing Retrieval Query

1. The query must be specific, mention concrete object names, shapes, spatial relations, or room types.
2. Focus on the information most relevant for immediate navigation or memory lookup, such as which region to check or which object to find.
3. Avoid vague or generic phrases like "find it", "the target", or "search around".

The output must strictly follow this format:

THOUGHT: concise reasoning about how you derive the query

QUERY: one short, specific retrieval phrase suitable for searching the memory

<Examples>

Now given a new task, generate ONE specific retrieval query to help the agent recall relevant past observations.

Task Instruction: {question} If a goal image is provided {image_goal}

If frontier observations are available, Frontier observations are available. Consider them implicitly in your reasoning. {frontier_imgs}

Figure 11. Query generation prompt.

Navigation Prompt

System Prompt

You are an embodied agent exploring an indoor environment. Your goal is to analyze previously visited areas (Memories) and unexplored regions (Frontiers) to determine the best next step to locate or find the target object described in the task instruction.

Rules:

1. Your whole reply MUST contain exactly two sections in this order:

A THOUGHT section that begins with "THOUGHT:" (uppercase) followed by your reasoning.

A single ANSWER line that begins with "ANSWER:" (uppercase) and provides exactly ONE decision.

2. THOUGHT content rules:

For each Memory i, include a short analysis of each Object j.

For each Frontier i, describe its spatial direction or likelihood.

3. The ANSWER line MUST be one of these exact formats:

ANSWER: Memory i, Object j

ANSWER: Frontier i

where i and j are plain integers.

IMPORTANT: When choosing a Memory, you MUST specify both the memory index and the object index.

4. Do NOT invent or hallucinate new object indices.

5. After analyzing all options, select the most promising Memory Object or the Frontier that maximizes discovery.

Definitions

Memory: A focused observation of several objects. It contains a full image of the cluster of objects, and separate image crops of each object. Choosing a memory means that the object asked in the task is within the cluster of objects that the memory represents, and you will choose that object as the result of task instruction. Therefore, if you choose a memory, you should also choose the object in the memory that you think is the result of task completion.

Frontier: An unexplored region that could provide new information or a closer view of the target object. Selecting a frontier means that you will further explore that direction.

Indices start at 0. Always reference indices. Do NOT write names instead of indices.

<Examples>

Task Instruction Template in Actual Use

Task Instruction: *{question}*

Select the Memory or Frontier that would help complete the task.

Final Required Output Format

You MUST output the **THOUGHT** section followed by ONE valid **ANSWER** line:

ANSWER: Memory *i*, Object *j*

ANSWER: Frontier *i*

Figure 12. Navigation prompt.

Question Answering Prompt

System Prompt
Task: You are an agent in an indoor scene tasked with answering questions by observing the surroundings and recalling memories.
Definitions
Memory: A focused observation of several objects. Choosing a Memory means that this memory image contains enough information for you to answer the question. If you choose a Memory, you must directly give an answer to the question. If you do not have enough information to give an answer, then state that the information is insufficient.
Question Format
Question: {question}
Select the Memory that would help you answer the question.
If an egocentric view is provided, it will appear before the list of memories.
Memories
The following are all the Memories that you can choose (together with the contained object classes):
Please note that the contained classes may not be accurate (wrong or missing classes) due to limitations of the object detection model.
You must still rely on the images themselves to make decisions.(Memory images and their associated detected object classes are listed here; indices start at 0.)
If no memory is available, the prompt states:
No Memory is available.
Answer Format
Two answer formats are supported:
1. Multiple-choice questions (answer_type = "choice")
Please provide your answer in the following format:
Memory i [Answer]
i is the index of the memory you choose.
[Answer] must be one of the options provided in the question (e.g., A, B, C, D).
2. Open-ended questions
Please provide your answer in the following format:
Memory i [Answer]
i is the index of the memory you choose.
[Answer] is a direct natural-language answer.
Example:
Memory 0
The fruit bowl is on the kitchen counter.
When answering:
Provide a direct and natural answer that others can understand.
Avoid phrases like “in this memory” or “on the left of the image”.
You may refer to all provided memories during reasoning, but you must select one most relevant memory to base your answer on.

Figure 13. Memory-based question answering prompt.