# LEARNING MULTI-STEP 3D REASONING FOR EMBODIED QUESTION ANSWERING

**Anonymous authors**
Paper under double-blind review

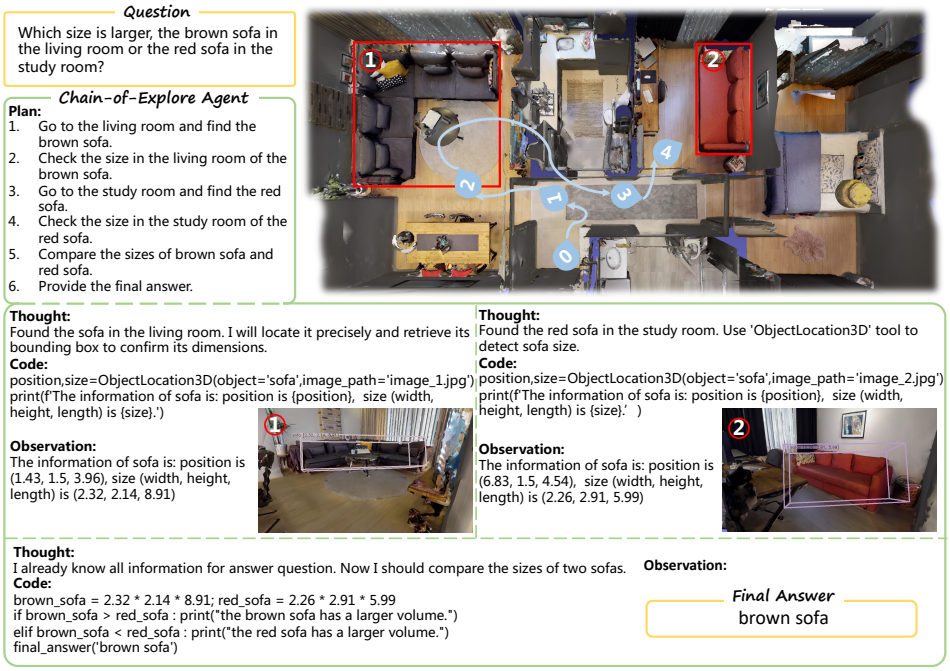



Figure 1: Overview of the proposed agent for Embodied Question Answering (EQA). Our agent decomposes questions into structured plans and reasons to select tools for exploration and answering.

## ABSTRACT

Embodied Question Answering (EQA) requires embodied agents to explore a 3D environment to find evidence and answer questions related to the environment. Some methods employ off-the-shelf vision-language models (VLMs) to solve EQA tasks, which exhibit limited 3D reasoning capabilities, making it difficult to accurately retrieve evidence and resulting in inefficient exploration and incorrect responses. In this paper, we propose a chain-of-exploration EQA agent that unifies exploration, spatial evidence acquisition, and question answering through iterative reasoning with tool use. Specifically, our agent reasons over both current observations and historical information, determines what spatial information is required for solving the task, selects appropriate tools to acquire geometric evidence, and decides where to explore next. To improve the reasoning and exploration behaviors of the agent, we further design an EQA data generation pipeline that automatically constructs EQA tasks with 3D reasoning trajectories. Based on the pipeline, we collect the EQA-RT dataset, which contains 18K tasks and is divided into a training set, EQA-RT-Train, and two test sets, EQA-RT-Seen (scenes overlapping with the training set) and EQA-RT-Unseen (unseen scenes). We perform supervised fine-tuning on EQA-RT-Train to initialize the model's tool-use capability, followed by evidence-grounded reinforcement learning to enhance task-relevant spatial evidence acquisition. Across public benchmarks, our method improves LLM-Match by 0.69 points and grounding-aware success by 2.93 points, while



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

reducing exploration distance by up to 61.4%. On our EQA-RT, it improves target-object recall by 0.06–0.12 and LLM-Match by 4.19–5.63 points.

## 1 INTRODUCTION

Embodied Question Answering (EQA) requires agents to explore a 3D environment, gather evidence, and answer questions about the environment Das et al. (2018); Gordon et al. (2018). Existing methods leverage vision-language models (VLMs) to reason from RGB observations (Ren et al., 2024a; Anwar et al., 2024; Jiang et al., 2025), or construct scene graphs and semantic maps to support exploration and answering (Saxena et al., 2025; Sakamoto et al., 2024; Li et al., 2026). Despite these advances, questions involving object dimensions, distances, and spatial relations require agents to determine what spatial information is needed and how to acquire it during exploration. As illustrated in Figure 1, insufficient geometric information can lead to incorrect answers about objects' 3D properties, while questions involving multiple objects across different regions require agents to acquire and integrate spatial evidence over successive exploration steps. The difficulty lies in deciding, from current observations and previously acquired evidence, what information is still missing, which actions can obtain it, and when sufficient evidence has been collected to answer. These tasks therefore call for multi-step 3D reasoning that connects spatial evidence acquisition with exploration decisions and answer prediction.

In this paper, we propose a chain-of-exploration agent that performs multi-step 3D reasoning through iterative tool use. The agent reasons over current observations and accumulated spatial memory to identify missing spatial information, selects measurement tools or navigation actions to acquire it, and uses newly collected evidence to guide subsequent exploration and answering. To learn these behaviors, the controller needs demonstrations of how to connect intermediate reasoning with tool use and exploration decisions. We therefore construct EQA tasks with 3D reasoning trajectories and use them for supervised fine-tuning to initialize the controller's tool-use capability. We further introduce reinforcement learning after supervised initialization to improve task-relevant spatial evidence acquisition while accounting for answer quality and exploration cost.

In doing so, we address two challenges. (1) Constructing reliable EQA tasks and coherent 3D reasoning trajectories is challenging. Visually and semantically similar objects complicate unambiguous question generation, while accurate answers require geometric measurements or scene-level understanding. The trajectories must identify missing information and select appropriate tools or exploration actions at each step, while remaining consistent with available observations and avoiding information unavailable to the agent. (2) Designing an effective reward is challenging because it requires identifying task-relevant evidence from executed tool interactions and converting it into a learning signal that balances evidence acquisition, answer quality, and exploration cost.

To address the first challenge, we introduce a novel EQA data generation pipeline that automatically generates EQA tasks with reasoning trajectories via three steps: EQA task generation, 3D reasoning trajectory generation, and validation. We first apply a 3D detection model to identify scene objects and extract their sizes and spatial coordinates. VLM then generates diverse question–answer pairs from this object-level information. To distinguish objects of the same category, we refine target descriptions using their spatial relations to surrounding objects. Next, we extract the objects mentioned in each question and use their positions with the A-star algorithm to construct reference navigation paths. Along these paths, VLM generates reasoning trajectories that connect intermediate observations, required spatial evidence, and tool-use decisions. The generated tasks and trajectories are then assessed by an EQA task verifier and a trajectory verifier to filter low-quality samples. Using this pipeline, we construct EQA-RT with approximately 18K question–answer pairs and corresponding reasoning trajectories. The dataset comprises EQA-RT-Train and two test sets, EQA-RT-Seen and EQA-RT-Unseen. The Seen split shares scenes with training, whereas the Unseen split contains scenes absent from training. We manually verify the tasks and trajectories in both test sets.

To address the second challenge, we propose an evidence-grounded reinforcement learning method that constructs trajectory-level rewards from task-relevant evidence acquired through tool interactions. Specifically, we measure interaction contributions based on the useful evidence obtained from tool outputs. We then compute the answer reward by scaling answer correctness with the coverage of required evidence, so that correct answers supported by more complete evidence receive higher rewards. We further introduce an evidence reward to encourage the acquisition of required evi-

dence while penalizing incomplete evidence, redundant or invalid actions, and excessive interaction cost. This provides trajectory-level learning signals that encourage task-relevant evidence acquisition while discouraging redundant exploration. On EQA-RT-Seen and EQA-RT-Unseen, our method improves LLM-Match over Pred-EQA by 5.63 and 4.19 points, increases Recall@10 by 0.12 and 0.06, and reduces exploration distance by 29.6% and 32.5%, respectively. On public benchmarks, it improves LLM-Match by 0.69 points and grounding-aware success by 2.93 points, while reducing exploration distance by up to 61.4%. Our contributions are four-fold.

- We propose a chain-of-exploration agent that performs multi-step 3D reasoning through iterative tool use, connecting spatial evidence acquisition with exploration and question answering.
- We develop an automatic EQA data generation pipeline that constructs tasks and reasoning trajectories, with task and trajectory verification to improve data reliability.
- We introduce EQA-RT, a dataset and benchmark containing 18K tasks with multi-step 3D reasoning and tool-use trajectories, supporting training and evaluation in seen and unseen scenes.
- We introduce an evidence-grounded reinforcement learning method that provides rewards for tool interactions, enabling effective and efficient evidence acquisition.

## 2 Related Works

### 2.1 Embodied Question Answering

Early EQA studies connect visual perception, navigation, and answering in interactive environments (Das et al., 2018; Gordon et al., 2018). Recent benchmarks extend evaluation to open-vocabulary questions, exploration-aware answering, decision reasoning, and abstention (Majumdar et al., 2024; Jiang et al., 2025; Gong et al., 2026; Wu et al., 2026b). To guide exploration, Explore-EQA and FAST-EQA use question relevance and confidence to prioritize informative observations and regions (Ren et al., 2024a; Zhang et al., 2026). Prune-Then-Plan calibrates frontier assessments to reduce unstable exploration (Frahm et al., 2026). Structured representations and persistent memory support evidence retrieval and spatial reasoning in GraphEQA (Saxena et al., 2025) and 3D-Mem (Yang et al., 2025). Pred-EQA further couples predictive hierarchical planning with separate structural and visual evidence memories to guide exploration (Yuan et al., 2026). These methods improve how agents explore and organize observations. Our work focuses on learning exploration and geometric measurement decisions for questions requiring spatial evidence across multiple steps.

### 2.2 Multi-step Reasoning and Embodied Tool Use

Reasoning and action can be interleaved to gather information during problem solving (Yao et al., 2023), while executable programs connect language reasoning with perception and control modules (Gupta & Kembhavi, 2023; Liang et al., 2023). Visually grounded reasoning traces connect intermediate inference with object references, plans, and robot states (Li et al., 2025; Zawalski et al., 2025). For spatial reasoning, dynamic APIs and geometric constraints support tool-based grounding and computation (Marsili et al., 2025; Chen et al., 2026b). In embodied environments, TANGO composes perception and navigation primitives (Ziliotto et al., 2025), MTU3D learns object grounding and frontier selection from exploration trajectories (Zhu et al., 2025), and OmniEVA incorporates adaptive 3D grounding and physical constraints into planning (Liu et al., 2026b). Recent work also uses reinforcement learning to learn multi-step reasoning and tool use, including SimpleTIR, VTool-R1, SpaceTools, and MemoryExplorer (Xue et al., 2026; Wu et al., 2026a; Chen et al., 2026a; Wang et al., 2026). Our work focuses on learning coupled navigation and geometric measurement decisions for EQA in 3D scenes, where the agent explores in 3D scenes and uses tools to gather evidence for answering.

## 3 Method

We propose a chain-of-exploration agent that integrates exploration, spatial evidence acquisition, and question answering through iterative reasoning with tool use. To train the agent, we develop an EQA data generation pipeline that constructs tasks and reasoning trajectories connecting observations, exploration decisions, and tool calls, yielding the EQA-RT dataset. We first perform

supervised fine-tuning on these trajectories to initialize the controller's tool-use capability, followed by evidence-grounded reinforcement learning to improve task-relevant spatial evidence acquisition and efficient question answering.

### 3.1 Chain-of-Exploration Agent

The agent performs multi-step 3D reasoning over current observations and accumulated evidence to guide exploration and measurement. As shown in Figure 2, it comprises (1) a planner that generates the task plan $p$; (2) a controller that generates thoughts $t$ and tool-invoking code $c$ from observations, interaction history, and spatial memory; (3) a spatial memory module $m$ that stores spatial evidence and exploration history; and (4) an executor that runs the code and returns updated observations.

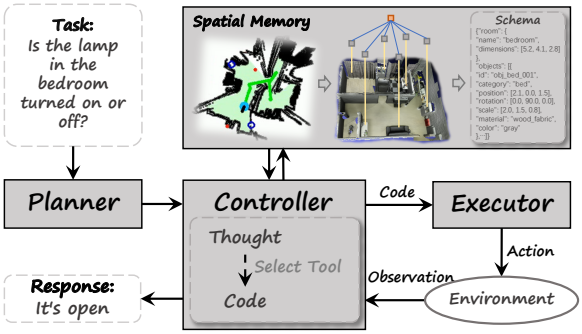



Figure 2: Overview of the chain-of-exploration agent workflow.

Given a query $Q$, the agent acquires information about the environment through successive tool interactions. At the $i$-th step, the controller generates thought $t_i$ and code $c_i$ according to

$$(t_i^*, c_i^*) = \arg\max_{t_i, c_i} P(t_i, c_i \mid Q, o_i, h_i, m_i, p), \tag{1}$$

where $P(\cdot)$ denotes the conditional probability modeled by the controller, $o_i$ is the current observation, and $m_i$ denotes the spatial memory state. The interaction history is defined as $h_i = \{(t_j, c_j, r_j)\}_{j=1}^{i-1}$, where $r_j$ denotes the execution feedback from step $j$. Based on current observations and accumulated evidence, the controller reasons about what information is still missing and selects tools to explore, locate relevant objects, or acquire geometric measurements.

The executor runs the generated code, and the resulting feedback is incorporated into the interaction history and spatial memory for subsequent reasoning. This allows the agent to integrate spatial evidence acquired at different steps. The interaction continues until the controller submits an answer through `FinalAnswer` or the interaction budget is exhausted. Details are provided in Appendix A.

### 3.2 EQA Task and Reasoning Trajectory Generation

As illustrated in Figure 3, the pipeline consists of EQA task generation, reasoning trajectory generation, and data verification. We use a pretrained VLM for data generation and model-based verification. We use HM3D Ramakrishnan et al. (2021) as the scene source.

**EQA Task Generation.** We first apply a 3D detection model to the entire scene to identify objects and extract their bounding boxes, positions, and categories. We sample objects from the detected set and provide their attributes and corresponding images to the data-generation VLM, together with example question answer pairs designed to reflect natural home conversations. The generated questions cover six types, including `relationship`, `status`, `distance`, `location`, `counting`, and `attribute`. To distinguish objects of the same category, we refine object descriptions using their relations to surrounding objects and scene context. For questions requiring geometric measurements or scene-level aggregation, we apply rule-based answer correction using the relevant objects and their properties. The complete scene annotations and reference answers are used for data construction and are not directly exposed to the controller during exploration.

**Reasoning Trajectory Generation.** We extract the objects mentioned in the question and use their locations and the initial agent position to construct reference navigation paths, obtaining intermediate waypoints and navigation directions. Along these paths, the data-generation VLM generates thoughts and tool-invoking code based on the observations available at each step. We categorize

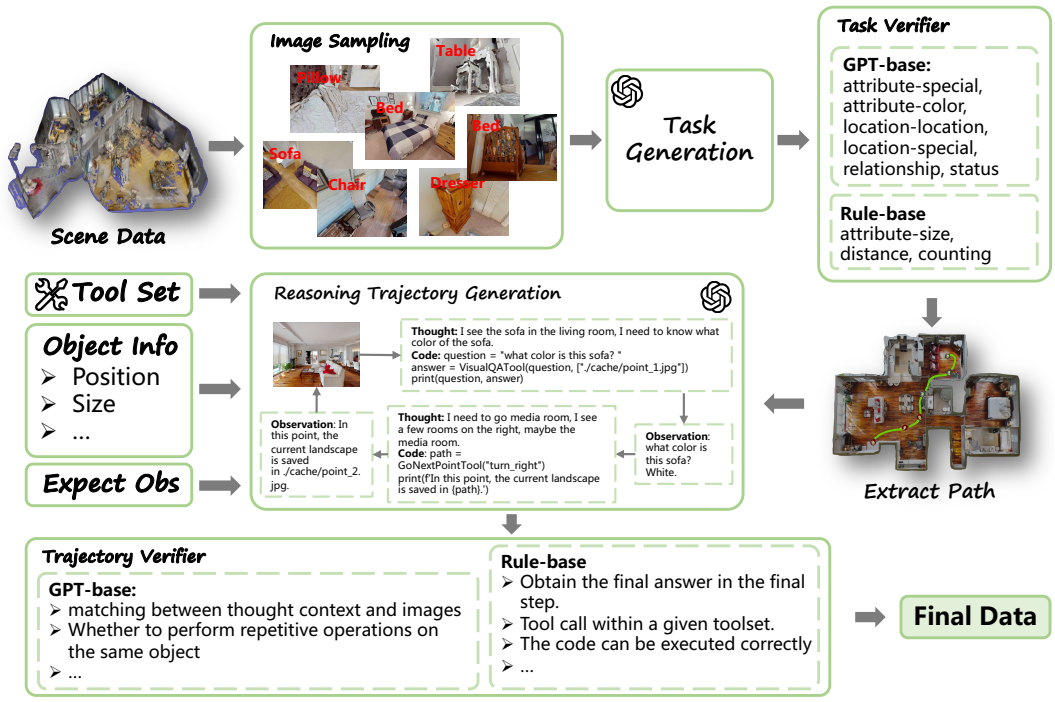



Figure 3: EQA Data Generation Pipeline.

steps as key or non-key according to whether a target object is found. For non-key steps, the VLM receives the current images and exploration directions to generate reasoning that guides further exploration. For key steps, question-type-specific prompts specify relevant considerations, reasoning strategies, tool-usage guidelines, and examples, guiding the VLM to select tools for acquiring the required object properties or geometric measurements.

**Data Verification.** We filter low-quality samples using task and trajectory verifiers. For task verification, we extract object descriptions from questions and options, retrieve the corresponding images, and assess image–description matching with Grounded-SAM Ren et al. (2024b) and question–image consistency with VLM scoring. Samples below either threshold are removed. For trajectory verification, rule-based checks validate required tool use at appropriate steps, tool names, and parameters. The VLM checks reasoning–observation consistency, whether the reasoning and observations support the answer, and semantic agreement between the predicted and reference answers. Samples failing these checks are discarded. A blinded study with 40 participants rating 771 samples before and 771 after filtering found that the mean quality score increased from 8.36 to 9.29 out of 10 (Appendix D.5).

**EQA-RT.** We represent each retained sample as $(S, pos, Q, p, \tau, A)$, where $S$ is the scene, $pos$ is the initial agent position, $Q$ is the question, $p$ is the task plan, and $A$ is the reference answer. The trajectory $\tau$ records the thoughts, code, and observations across exploration steps. Using this pipeline, we construct EQA-RT with approximately 18K tasks. The dataset contains EQA-RT-Train with 16,348 samples, EQA-RT-Seen with 845 samples, and EQA-RT-Unseen with 1,069 samples. EQA-RT-Seen shares scenes with the training set, and EQA-RT-Unseen contains scenes absent from training. We manually verify both test sets. Dataset statistics are provided in Appendix B.1.

### 3.3 Learning to Explore and Acquire Spatial Evidence

#### 3.3.1 Supervised Fine-Tuning

We first train the controller for two epochs using the reasoning trajectories from the full EQA-RT-Train split. At each step, the controller learns to generate thoughts and tool-invoking code based

on current observations, historical interactions, spatial memory, and the task plan. We optimize the controller using the cross-entropy loss

$$\min_{\theta} \mathbb{E}_{\tau\sim\mathbb{D}}\left[-\sum_{i=1}^{n}\log P_\theta(t_i, c_i \mid Q, o_i, h_i, m_i, p)\right], \tag{2}$$

where $\mathbb{D}$ denotes EQA-RT-Train and $n$ is the number of steps in the demonstrated trajectory. The loss is computed over the thought and code tokens, with observations and tool outputs serving as context. We train only the controller and leave the planner unchanged.

#### 3.3.2 Evidence-Grounded Reinforcement Learning

Answer correctness alone does not indicate whether an agent has acquired the spatial evidence required by an EQA task. We jointly optimize evidence acquisition and answer quality while accounting for interaction cost. For each rollout, we first compute evidence coverage from the required evidence acquired through tool interactions. We then construct an answer reward for answer correctness and an evidence reward for evidence acquisition and interaction efficiency, and combine them with GDPO (Liu et al., 2026a) for policy optimization.

**Evidence Coverage.** Given a question $Q$, let $\mathcal{F}_Q$ denote the required evidence facts specified by its question type and target-object annotations. For example, answering a distance question requires obtaining 3D positions of target objects. For a rollout $\tau$, we construct the acquired evidence facts $\mathcal{E}(\tau)$ from outputs of executed tool calls. We then compute the evidence coverage $C(\tau)$, which measures the proportion of required evidence acquired during the rollout, as

$$C(\tau) = \frac{|\mathcal{E}(\tau)\cap\mathcal{F}_Q|}{|\mathcal{F}_Q|}. \tag{3}$$

The required evidence is credited once, so repeated acquisition of the same evidence does not increase $C(\tau)$. Evidence credited to individual tool outputs is accumulated into this terminal coverage, which is used to compute the trajectory rewards below.

**Answer Reward.** Let $s(\tau)$ measure the correctness of the submitted answer against the reference answer, following the scoring rules in Appendix C. We condition this score on the acquired evidence and assign a separate penalty when no answer is submitted, which is computed as

$$R_{\text{ans}}(\tau) = \begin{cases} C(\tau)s(\tau), & \text{if an answer is submitted,} \\ -2, & \text{otherwise.} \end{cases} \tag{4a}$$

For a correct answer with a positive score, higher coverage yields a larger answer reward. A submitted answer with zero coverage receives zero answer reward regardless of its correctness.

**Evidence Reward.** The evidence reward encourages evidence acquisition while penalizing incomplete evidence at answer submission and inefficient interactions. We define it as

$$R_{\text{ev}}(\tau) = 2C(\tau) - \mathcal{P}_{\text{stop}}(\tau) - \mathcal{P}(\tau). \tag{4b}$$

$2C(\tau)$ rewards the agent for acquiring sufficient task-relevant evidence.

The stopping penalty $\mathcal{P}_{\text{stop}}(\tau)$ discourages the agent from submitting an answer when required evidence is still missing. Mathematically, for a submitted answer, $\mathcal{P}_{\text{stop}}(\tau)$ is defined as

$$\mathcal{P}_{\text{stop}}(\tau) = 0.5(1 - C(\tau)) + 0.3\mathbb{I}[C(\tau) = 0],$$

The term $0.5(1 - C(\tau))$ penalizes the fraction of required evidence that remains missing, while $0.3\mathbb{I}[C(\tau) = 0]$ adds an extra penalty for answering without acquiring any required evidence. The stopping penalty vanishes at full coverage. For trajectories without a submitted answer, $\mathcal{P}_{\text{stop}} = 0$, and the missing answer is penalized separately through $R_{\text{ans}} = -2$.

The interaction cost $\mathcal{P}(\tau)$ discourages the agent from invalid, duplicate, and no-progress tool calls, excessive tool usage, and overly long navigation paths. $\mathcal{P}(\tau)$ is computed as

$$\mathcal{P}(\tau) = 0.2N_{\text{invalid}} + 0.1N_{\text{duplicate}} + 0.05N_{\text{no-progress}} + 0.01N_{\text{tool}} + 0.2\min\left\{\max\left(\frac{L}{L^\star} - 1, 0\right), 2\right\}. \tag{4c}$$

Here, $N_{\text{invalid}}$ counts execution errors, invalid code, and outputs failing tool-specific validity checks; $N_{\text{duplicate}}$ counts repeated action–argument–input-image signatures or explicitly rejected duplicate calls; $N_{\text{no-progress}}$ counts valid 2D/3D localization and visual question-answering calls that do not increase evidence coverage, excluding navigation and cropping; and $N_{\text{tool}}$ counts navigation, localization, cropping, and visual question-answering calls, excluding computation, final-answer submission, invalid code, and unknown actions. $L$ and $L^\star$ denote the executed and reference path lengths, respectively, and the path term is omitted when either is unavailable or $L^\star \leq 0$.

**Advantage Estimation.** For each task $Q$, we sample a group of $G$ rollout trajectories $\mathcal{G}_Q = \{\tau_j\}_{j=1}^{G}$. Following GDPO, we normalize each reward dimension separately within this group as

$$A_k(\tau_j) = \frac{R_k(\tau_j) - \mu_k(Q)}{\sigma_k(Q) + \epsilon_{\text{norm}}}, \qquad k \in \{\text{ev}, \text{ans}\}, \tag{5}$$

where $\mu_k(Q)$ and $\sigma_k(Q)$ are the group mean and standard deviation, and $\epsilon_{\text{norm}} > 0$ ensures numerical stability. The combined advantage is computed as

$$A(\tau_j) = A_{\text{ev}}(\tau_j) + A_{\text{ans}}(\tau_j). \tag{6}$$

# 4 Experiments

## 4.1 Settings

**Benchmarks and Baselines.** We evaluate our method on EQA-RT-Seen and EQA-RT-Unseen, together with three public benchmarks, HM-EQA (Ren et al., 2024a), OpenEQA (Majumdar et al., 2024), and EXPRESS-Bench (Jiang et al., 2025). We compare with Explore-EQA (Ren et al., 2024a), FineEQA (Jiang et al., 2025), 3D-Mem (Yang et al., 2025), GraphEQA (Saxena et al., 2025), FAST-EQA (Zhang et al., 2026), and Pred-EQA (Yuan et al., 2026), with the evaluated baselines varying by benchmark.

**Implementation Details.** We use Qwen3-VL-8B-Instruct as the controller and train it on EQA-RT-Train through SFT followed by evidence-grounded RL. We evaluate on public benchmarks without benchmark-specific fine-tuning. Our ablation variants share the same backbone, tool implementations, initial states, and test-time interaction budgets. At inference, the planner uses the available question and observations without access to reference plans or target annotations. Detailed configurations are provided in Appendix D.1.

**Evaluation Metrics.** We report answer accuracy for closed-vocabulary tasks and LLM-Match for open-vocabulary tasks. On EQA-RT, we additionally report target-object recall ($recall@D$), trajectory length, and path efficiency ($e_{path}@D$). For public benchmarks, we follow their respective evaluation protocols. On EXPRESS-Bench, these include grounding-aware success, answer correctness, path efficiency, and the geodesic distance from the final position to the target. Metric definitions are provided in Appendix C.

## 4.2 Main Results

Table 1 compares our method with existing approaches on EQA-RT-Seen and EQA-RT-Unseen. Our method achieves LLM-Match scores of 50.32 and 48.16, exceeding Pred-EQA, the strongest reported baseline for this metric, by 5.63 and 4.19 points, respectively. It also achieves Recall@10 of 0.41 and 0.37, improving over Pred-EQA, the strongest reported baseline for this metric, by 0.12 and 0.06. Exploration trajectory lengths are 22.15 m on Seen and 23.19 m on Unseen, decreases of 29.6% and 32.5% relative to Pred-EQA, respectively. These joint improvements in answering, target-object retrieval, and exploration efficiency support the effectiveness of connecting spatial evidence acquisition with exploration decisions. The results on Unseen further show that the learned behavior transfers to scenes absent from training.

Table 2 reports results on HM-EQA, OpenEQA, and EXPRESS-Bench. On HM-EQA, our method achieves 65.77% accuracy with an average exploration distance of 20.93 m. On OpenEQA, it obtains an LLM-Match score of 53.99 and an LLM-Match SPL of 47.21. On EXPRESS-Bench, it

| Dataset | Model | $recall \uparrow$ | | | $e_{path} \uparrow$ | | | traj. | LLM- |
|---|---|---|---|---|---|---|---|---|---|
| | | @5 | @10 | @15 | @5 | @10 | @15 | length (m) ↓ | Match ↑ |
| EQA-RT Seen | Explore-EQA (Ren et al., 2024a) | 0.04 | 0.10 | 0.13 | 0.04 | 0.06 | 0.09 | 136.7 | 33.11 |
| | FineEQA (Jiang et al., 2025) | 0.04 | 0.09 | 0.12 | 0.04 | 0.06 | 0.10 | 55.86 | 34.18 |
| | 3D-Mem (Yang et al., 2025) | 0.05 | 0.10 | 0.13 | 0.04 | 0.09 | 0.11 | 46.19 | 32.31 |
| | GraphEQA (Saxena et al., 2025) | 0.06 | 0.11 | 0.18 | 0.08 | 0.19 | 0.23 | 34.41 | 31.57 |
| | FAST-EQA (Zhang et al., 2026) | 0.11 | 0.23 | 0.32 | 0.07 | 0.20 | 0.25 | 49.56 | 39.62 |
| | Pred-EQA (Yuan et al., 2026) | **0.14** | 0.29 | 0.38 | 0.07 | 0.21 | 0.27 | 31.48 | 44.69 |
| | **Ours** | 0.13 | **0.41** | **0.50** | **0.08** | **0.21** | **0.28** | **22.15** | **50.32** |
| EQA-RT Unseen | Explore-EQA (Ren et al., 2024a) | 0.05 | 0.09 | 0.15 | 0.05 | 0.10 | 0.13 | 159.4 | 32.72 |
| | FineEQA (Jiang et al., 2025) | 0.05 | 0.09 | 0.17 | 0.05 | 0.11 | 0.17 | 58.99 | 33.75 |
| | 3D-Mem (Yang et al., 2025) | 0.05 | 0.09 | 0.15 | 0.06 | 0.15 | 0.18 | 42.23 | 35.98 |
| | GraphEQA (Saxena et al., 2025) | 0.06 | 0.13 | 0.17 | 0.07 | 0.20 | 0.23 | 35.76 | 31.13 |
| | FAST-EQA (Zhang et al., 2026) | 0.10 | 0.19 | 0.41 | 0.08 | 0.17 | 0.25 | 50.92 | 40.41 |
| | Pred-EQA (Yuan et al., 2026) | 0.14 | 0.31 | 0.40 | 0.08 | 0.20 | 0.25 | 34.35 | 43.97 |
| | **Ours** | **0.15** | **0.37** | **0.49** | **0.08** | **0.21** | **0.27** | **23.19** | **48.16** |

Table 1: Evaluation on EQA-RT-Seen and EQA-RT-Unseen under the open-vocabulary setting. Trajectory length is measured in meters.

| Model | HM-EQA | | OpenEQA (A-EQA) | | EXPRESS-Bench | | | |
|---|---|---|---|---|---|---|---|---|
| | Acc.(%) ↑ | $L$ (m) ↓ | LLM-Match ↑ | LLM-SPL ↑ | Succ. ↑ | Succ.$^*$ ↑ | $E_{path}$ ↑ | $d_T$ ↓ |
| Explore-EQA (Ren et al., 2024a) | 50.4 | 93.687 | 46.9 | 23.4 | – | – | – | – |
| FineEQA (Jiang et al., 2025) | 56.0 | 54.267 | 43.3 | 29.2 | 40.55 | 63.95 | 16.22 | 6.43 |
| 3D-Mem (Yang et al., 2025) | – | – | 52.6 | 42.0 | – | – | – | – |
| FAST-EQA (Zhang et al., 2026) | **69.2** | – | 49.0 | 27.70 | – | 68.7 | – | – |
| Pred-EQA (Yuan et al., 2026) | – | – | <u>53.3</u> | **48.5** | <u>52.58</u> | <u>70.54</u> | <u>47.66</u> | **5.64** |
| Ours | <u>65.77</u> | **20.93** | **53.99** | <u>47.21</u> | **55.51** | **72.61** | **48.37** | <u>5.89</u> |

Table 2: Results on public benchmarks. Baseline results are taken from published papers or official repositories.

achieves a grounding-aware success score of 55.51 and a path-efficiency score of 48.37. These evaluations extend the assessment to multiple-choice and open-vocabulary questions across different benchmarks.

### 4.3 Ablation Study

| Training | EQA-RT-Seen | | | | EQA-RT-Unseen | | | |
|---|---|---|---|---|---|---|---|---|
| | LLM-Match (%) ↑ | Recall @10 ↑ | Evidence Coverage (%) ↑ | $L$ (m) ↓ | LLM-Match (%) ↑ | Recall @10 ↑ | Evidence Coverage (%) ↑ | $L$ (m) ↓ |
| None | 38.32 | 0.12 | 17.87 | 43.65 | 35.17 | 0.10 | 15.62 | 55.37 |
| SFT-only | 41.25 | 0.13 | 17.72 | 25.76 | 39.83 | 0.10 | 15.50 | 33.49 |
| RL-only | 47.53 | 0.32 | 33.77 | 38.48 | 45.09 | 0.35 | 30.69 | 44.51 |
| SFT+RL | **50.32** | **0.41** | **41.09** | **22.15** | **48.16** | **0.37** | **38.50** | **23.19** |

Table 3: Comparison of training regimes with the same controller backbone, agent architecture, and evaluation protocol. None denotes no task-specific training.

**Training Regimes.** Table 3 compares four settings: None uses the controller without task-specific training, SFT-only applies only supervised fine-tuning, RL-only applies only reinforcement learning, and SFT+RL is our full method. Compared with None, SFT-only increases LLM-Match from 38.32% to 41.25% on Seen and from 35.17% to 39.83% on Unseen, while reducing exploration length by 17.89 m and 21.88 m, respectively, with little change in Recall@10 or evidence coverage. Compared with SFT-only, RL-only improves LLM-Match from 41.25% to 47.53% on Seen and from 39.83% to 45.09% on Unseen, and achieves higher Recall@10 and evidence coverage on both

| Variant | EQA-RT-Seen | | | | EQA-RT-Unseen | | | |
|---|---|---|---|---|---|---|---|---|
| | LLM-Match (%) ↑ | Recall @10 ↑ | Evidence Coverage (%) ↑ | $L$ (m) ↓ | LLM-Match (%) ↑ | Recall @10 ↑ | Evidence Coverage (%) ↑ | $L$ (m) ↓ |
| SFT-only | 41.25 | 0.13 | 17.72 | 25.76 | 39.83 | 0.10 | 15.50 | 33.49 |
| w/o $\mathcal{P}_{\text{stop}}(\tau)$ | 47.38 | 0.38 | 34.51 | 33.66 | 47.26 | 0.31 | 30.78 | 30.92 |
| w/o $\mathcal{P}(\tau)$ | 49.63 | 0.41 | 40.08 | 30.57 | 42.73 | 0.35 | 37.99 | 32.31 |
| w/o $C(\tau)$ in $R_{\text{ev}}$ | 46.58 | 0.36 | 35.50 | 25.86 | 43.84 | 0.33 | 33.94 | 27.56 |
| w/o $C(\tau)$ in $R_{\text{ans}}$ | 51.08 | 0.38 | 36.27 | 23.74 | 48.62 | 0.35 | 35.82 | 23.53 |
| w/o $C(\tau)$ in $R_{\text{ans}}$ and $R_{\text{ev}}$ | 43.46 | 0.35 | 31.34 | 25.80 | 42.97 | 0.31 | 30.40 | 37.75 |
| Full reward | 50.32 | 0.41 | 41.09 | 22.15 | 48.16 | 0.37 | 38.50 | 23.19 |

Table 4: Reward-component ablations on EQA-RT-Seen and EQA-RT-Unseen.

splits, but requires longer exploration paths. Starting from the SFT controller, evidence-grounded RL further improves LLM-Match from 41.25% to 50.32% on Seen and from 39.83% to 48.16% on Unseen, and increases Recall@10 from 0.13 to 0.41 and from 0.10 to 0.37, respectively. It also increases evidence coverage from 17.72% to 41.09% on Seen and from 15.50% to 38.50% on Unseen, while reducing exploration length by 3.61 m and 10.30 m, respectively. Our full SFT+RL method achieves the best results across all four metrics among the four settings, supporting the effectiveness of the two-stage training scheme in improving answer quality, target retrieval, evidence acquisition, and exploration efficiency. A paired trajectory analysis suggests that RL improves evidence acquisition through more frequent target detection and more consistent use of detector outputs in subsequent visual tool calls (Appendix D.3).

**Reward Components.** Table 4 examines the components of Equation 4, with SFT-only as a reference. All RL variants use the same SFT initialization, RL data, and training budget. The joint ablation without $C(\tau)$ in $R_{\text{ans}}$ and $R_{\text{ev}}$ removes the direct evidence reward, stopping penalty, all coverage factors, and coverage-based no-progress penalty, retaining only answer rewards and coverage-independent interaction costs. Compared with this control, the full reward increases LLM-Match from 43.46% to 50.32% on Seen and from 42.97% to 48.16% on Unseen. Evidence coverage rises from 31.34% to 41.09% and from 30.40% to 38.50%, respectively, alongside higher Recall@10 and shorter exploration paths. For the variant without $C(\tau)$ in $R_{\text{ev}}$, we replace $C(\tau)$ in the direct evidence reward $2C(\tau)$ with 1, leaving $\mathcal{P}_{\text{stop}}$, $\mathcal{P}$, and $R_{\text{ans}}$ unchanged. This variant reduces LLM-Match, Recall@10, and evidence coverage while increasing trajectory length on both splits. The remaining individual ablations remove only the indicated term or factor, leaving all other reward components unchanged. Removing $\mathcal{P}_{\text{stop}}(\tau)$ lowers evidence coverage despite longer exploration. Removing $\mathcal{P}(\tau)$ lengthens paths by 8.42 m on Seen and 9.12 m on Unseen without improving LLM-Match or coverage, supporting its role in controlling exploration cost. Removing $C(\tau)$ from $R_{\text{ans}}$ raises LLM-Match to 51.08% on Seen and 48.62% on Unseen but reduces evidence coverage and Recall@10. This trade-off shows that higher answer scores do not necessarily reflect more complete evidence acquisition.

## 5 Conclusion

In this work, we propose a chain-of-exploration agent that enables multi-step 3D reasoning through iterative tool use for solving EQA tasks. The proposed agent jointly performs exploration, spatial evidence acquisition, and question answering by reasoning over current observations and accumulated evidence to determine what information is still needed and how to acquire it. The constructed data generation pipeline can automatically construct EQA tasks and reasoning trajectories, with task and trajectory verification to improve data reliability. Using this pipeline, we construct EQA-RT, a dataset containing 18K EQA tasks with reasoning trajectories. We first perform supervised fine-tuning to initialize the controller's tool-use capability. The designed evidence-grounded reinforcement learning method can further improve tool-use capability by rewarding task-relevant evidence acquisition, scaling answer rewards by evidence coverage, and penalizing inefficient interactions. Experiments on EQA-RT and multiple public benchmarks evaluate answer quality, target-object recall, and exploration efficiency, examining the effectiveness of learning to acquire spatial evidence for embodied question answering.

## AI USE STATEMENT

We used GPT-4o (`gpt-4o-2024-08-06`) to generate and verify synthetic EQA tasks and reasoning trajectories, including thoughts and tool-invoking code, for EQA-RT. The Qwen models used as the controller, visual question-answering tool, and automated answer evaluator are documented in the method and appendix. We also used OpenAI Codex to review the manuscript and assist with English wording.

## REPRODUCIBILITY STATEMENT

We describe the agent architecture, data generation pipeline, and training objectives in the main text. The supplementary material provides dataset statistics and split information, pretrained model identifiers, SFT and RL configurations, evidence validation rules, and policy optimization details. RL hyperparameters are summarized in Table 6. Evaluation metrics, including evidence coverage, are defined in Appendix C, and tool implementations and inference configurations are specified in Appendix E. We also provide the prompts used for planning and agent execution in the supplementary material to clarify the inputs and instructions used by the system. We will publicly release our code, datasets, and model weights to facilitate reproducibility.

## ETHICS STATEMENT

This work studies embodied question answering in simulated indoor environments using existing scene datasets and synthetically generated tasks. Our human evaluation involved 40 participants, all of whom provided informed consent and received compensation, as described in Appendix D.5.

## REFERENCES

Abrar Anwar, John Welsh, Joydeep Biswas, Soha Pouya, and Yan Chang. Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation. *arXiv preprint arXiv:2409.13682*, 2024.

Siyi Chen, Mikaela Angelina Uy, Chan Hee Song, Faisal Ladhak, Adithyavairavan Murali, Qing Qu, Stan Birchfield, Valts Blukis, and Jonathan Tremblay. Spacetools: Tool-augmented spatial reasoning via double interactive rl. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026a.

Zeren Chen, Xiaoya Lu, Zhijie Zheng, Pengrui Li, Lehan He, Yijin Zhou, Jing Shao, Bohan Zhuang, and Lu Sheng. Geometrically-constrained agent for spatial reasoning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026b.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Noah Frahm, Prakrut Patel, Yue Zhang, Shoubin Yu, Mohit Bansal, and Roni Sengupta. Prune-then-plan: Step-level calibration for stable frontier exploration in embodied question answering. *Findings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Xicheng Gong, Qiwei Li, Peiran Xu, and Yadong Mu. Extending embodied question answering from perception to decision. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. Iqa: Visual question answering in interactive environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Kaixuan Jiang, Yang Liu, Weixing Chen, Jingzhou Luo, Ziliang Chen, Ling Pan, Guanbin Li, and Liang Lin. Beyond the destination: A novel benchmark for exploration-aware embodied question answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Ji Li, Bo Wang, Jing Xia, Mingyi Li, and Shiyan Hu. Himm: Human-inspired long-term memory modeling for embodied exploration and question answering. *arXiv preprint arXiv:2602.15513*, 2026.

Zejun Li, Ruipu Luo, Jiwen Zhang, Minghui Qiu, Xuanjing Huang, and Zhongyu Wei. Vocot: Unleashing visually grounded multi-step reasoning in large multi-modal models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2025.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2023.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. GDPO: Group reward-decoupled normalization policy optimization for multi-reward RL optimization. *arXiv preprint arXiv:2601.05242*, 2026a.

Yuecheng Liu, Dafeng Chi, Shiguang Wu, Zhanguang Zhang, Yuzheng Zhuang, Bowen Yang, He Zhu, Lingfeng Zhang, Pengwei Xie, David Gamaliel Arcos Bravo, Yingxue Zhang, Jianye Hao, and Xingyue Quan. Omnieva: Embodied versatile planner via task-adaptive 3d-grounded and embodiment-aware reasoning. *The Fourteenth International Conference on Learning Representations*, 2026b.

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

Damiano Marsili, Rohun Agrawal, Yisong Yue, and Georgia Gkioxari. Visual agentic ai for spatial reasoning with a dynamic api. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Santhosh K Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alex Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, et al. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv preprint arXiv:2109.08238*, 2021.

Allen Z Ren, Jaden Clark, Anushri Dixit, Masha Itkina, Anirudha Majumdar, and Dorsa Sadigh. Explore until confident: Efficient exploration for embodied question answering. In *Robotics: Science and Systems (RSS)*, 2024a.

Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024b.

Koya Sakamoto, Daichi Azuma, Taiki Miyanishi, Shuhei Kurita, and Motoaki Kawanabe. Map-based modular approach for zero-shot embodied question answering. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

Saumya Saxena, Blake Buchanan, Chris Paxton, Peiqi Liu, Bingqing Chen, Narunas Vaskevicius, Luigi Palmieri, Jonathan Francis, and Oliver Kroemer. Grapheqa: Using 3d semantic scene graphs for real-time embodied question answering. In *Proceedings of the 9th Conference on Robot Learning (CoRL)*, 2025.

Sen Wang, Bangwei Liu, Zhenkun Gao, Lizhuang Ma, Xuhong Wang, Yuan Xie, and Xin Tan. Explore with long-term memory: A benchmark and multimodal llm-based reinforcement learning framework for embodied exploration. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Mingyuan Wu, Jingcheng Yang, Jize Jiang, Meitang Li, Kaizhuo Yan, Hanchao Yu, Minjia Zhang, ChengXiang Zhai, and Klara Nahrstedt. Vtool-r1: Vlms learn to think with images via reinforcement learning on multimodal tool use. *The Fourteenth International Conference on Learning Representations*, 2026a.

Tao Wu, Chuhao Zhou, Guangyu Zhao, Haozhi Cao, Yewen Pu, and Jianfei Yang. When robots should say "i don't know": Benchmarking abstention in embodied question answering. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026b.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *The Fourteenth International Conference on Learning Representations*, 2026.

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 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Bowen Yuan, Sisi You, and Bing-Kun Bao. Predict before you explore: Predictive planning with specialized memory for embodied question answering. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *Proceedings of the 8th Conference on Robot Learning (CoRL)*, 2025.

Haochen Zhang, Nirav Savaliya, Faizan Siddiqui, and Enna Sachdeva. FAST-EQA: Efficient embodied question answering with global and local region relevancy. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2026.

Ziyu Zhu, Xilin Wang, Yixuan Li, Zhuofan Zhang, Xiaojian Ma, Yixin Chen, Baoxiong Jia, Wei Liang, Qian Yu, Zhidong Deng, et al. Move to understand a 3d scene: Bridging visual grounding and exploration for efficient and versatile embodied navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Filippo Ziliotto, Tommaso Campari, Luciano Serafini, and Lamberto Ballan. Tango: training-free embodied ai agents for open-world tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

# A CHAIN-OF-EXPLORATION AGENT

To enable the agent to reason and act in complex environments, we propose a chain-of-exploration agent that integrates tool-usage strategies for the reasoning process. Our agent conducts step-by-step reasoning on past observations and, at each step, generates corresponding thoughts and code to execute tools. Code offers greater flexibility than formats such as JSON for handling diverse inputs and outputs. As shown in Figure 2, our agent comprises four components: a planner for generating overall task plan $p$, a controller for generating thought $t$ and code $c$, a spatial memory module $m$ for maintaining accumulated spatial evidence and historical exploration information, and an executor for executing code in the environment. Given a query $Q$, the controller generates thought and code at step $i$ according to

$$(t_i^*, c_i^*) = \arg\max_{t_i, c_i} P(t_i, c_i \mid Q, o_i, h_i, m_i, p), \tag{7}$$

where $P(\cdot)$ denotes the conditional probability modeled by the controller, $o_i$ is the current observation, and $m_i$ denotes the spatial memory state. The interaction history is defined as $h_i = \{(t_j, c_j, r_j)\}_{j=1}^{i-1}$, where $r_j$ denotes the execution feedback from step $j$.

**Planner.** Given an EQA task, the planner, modeled as an LLM, takes the question and available observations as input, interprets the task objectives, and outputs an overall plan that decomposes the task into sub-goals. At inference, it has no access to reference plans or target annotations. The structured sub-goals are provided to the controller to prevent blind exploration, enhancing the efficiency and accuracy of task execution.

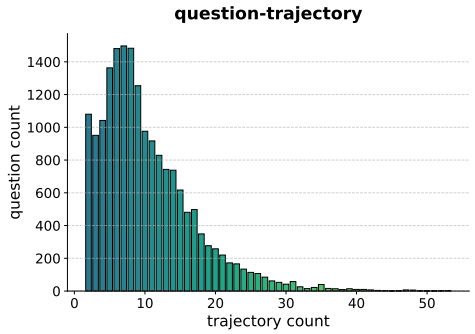
**Executor.** We deploy real-executable tools for the agent. Our tools include `GoNextPoint`, `ObjectLocation2D`, `ObjectLocation3D`, `ObjectCrop`, `VisualQA`, `FinalAnswer`, the details of tools see Appendix E. With the generated code, the executor calls executable tools in the environment to obtain new observations for further exploration, thus solving the EQA task.

**Controller.** The controller performs dynamic reasoning to decide which tool to use, based on the question, previous observations, and the guidance of plan, and invokes the executor to gather new observations for further reasoning until the answer is derived. The reasoning process can be divided into three situations.

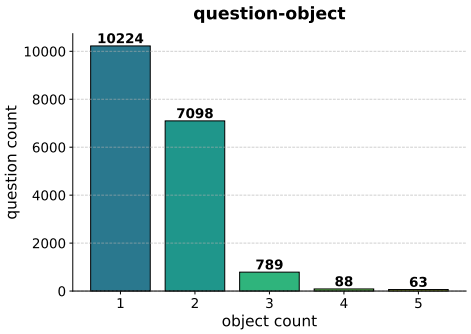
- **The collected information is insufficient and the current scene lacks required objects.** Our method infers missing objects from previous observations and the query, and estimates their likely positions. Then, our agent combines these estimates with its current location to decide a walking direction, and uses the '`GoNextPoint`' (for example, '`GoNextPoint`("turn left")') to gather the needed information.
- **The collected information is insufficient and the current scene contains involved objects.** Our method reasons over the question and invokes suitable tools to obtain relevant information. For example, as shown in Figure 1, our method utilizes '`ObjectLocation3D`' to extract the size of objects, and then continues exploration following the above step.
- **The collected information is sufficient.** Our method needs to reason over the question, use appropriate tools on the current image, and integrate information for the final answer. For example, it processes the detected object, then writes 'Python' codes for comparing sizes, and employs '`FinalAnswer`' to produce the ultimate output.

**Spatial Memory.** $m_i$ denotes the spatial memory state at step $i$, which maintains structured object-centric spatial evidence distilled from tool outputs obtained during exploration. As shown in Figure 2, the spatial memory module receives the outputs from the executor after each tool call and updates its internal representation in a deterministic, rule-based manner without requiring learnable parameters. Specifically, when a tool returns spatial information (*e.g.*, `ObjectLocation3D` outputs object position and size, `GoNextPoint` outputs a new exploration position), the module parses the output into a structured object-wise buffer:

- Explored positions are accumulated as a list of 3D coordinates from `GoNextPoint` calls.
- Detected objects are stored as key-value entries keyed by object category, where each entry records the 3D position, size (width, height, length), and the step at which the object was detected.

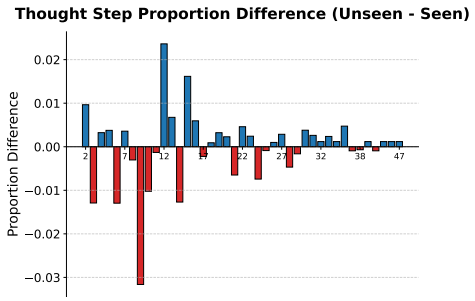



(a) Statistics on the number of questions with different step counts.

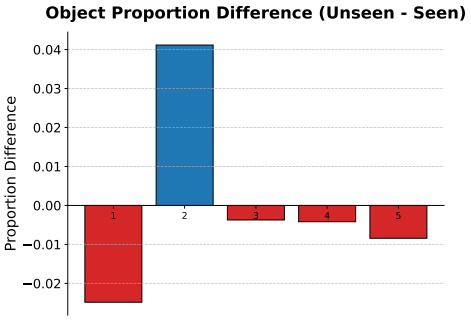



(b) Statistics on the number of questions with different object counts.

Figure 4: All data statistics on Thought Steps and Objects.




(a) The difference in the ratio of the average thought steps between the EQA-RT-Seen and the EQA-RT-Unseen.




(b) The difference in the ratio of the average number of objects between the EQA-RT-Seen and the EQA-RT-Unseen.

Figure 5: Comparison of EQA-RT-Seen and EQA-RT-Unseen on Thought Step and Object.

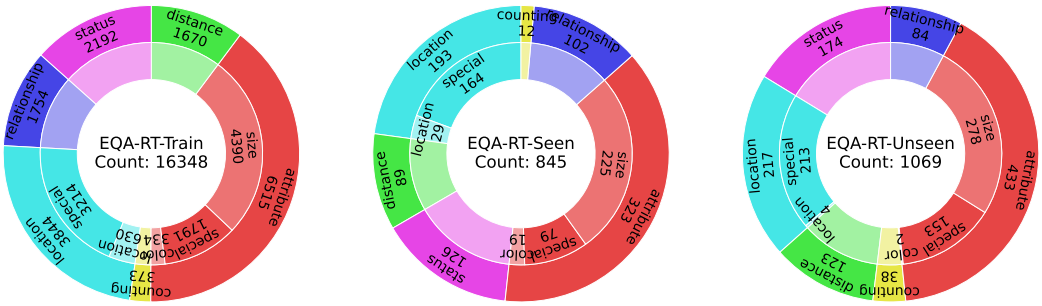
- Object images cropped by `ObjectCrop` are saved at their corresponding entry for visual reference.

At each reasoning step, the spatial memory is serialized into a textual representation and appended to the controller's input prompt, providing a concise summary of accumulated spatial evidence. This design differs from the raw interaction history $h_i$ in that $h_i$ contains verbose, noisy tool outputs and execution logs, whereas $m_i$ offers a compressed, structured view of object-centric spatial information. During training, $m_i$ can be deterministically constructed from the trajectory data by applying the same rule-based parsing over the first $i-1$ steps' tool observations, requiring no additional annotation. The structured representation enables the controller to efficiently retrieve cross-view spatial information accumulated over time, facilitating reliable comparison and reasoning across multiple objects.

# B EQA-RT

## B.1 Data Statistic

As shown in Figure 4, the dataset exhibits a pronounced long-tail distribution in both the average exploration steps and the number of related objects per question. Most questions require around ten exploration steps; among them, 10,224 involve a single target, 7,098 involve two targets, and 940 involve three or more objects. And Figure 5(a) compares EQA-RT-Seen and EQA-RT-Unseen in terms of the average thought steps per question, revealing that EQA-RT-Unseen involves more thought steps. Figure 5(b) further compares EQA-RT-Seen and EQA-RT-Unseen in terms of the average number of objects per question, revealing that EQA-RT-Seen involves more objects.




(a) EQA-RT-Train (b) EQA-RT-Seen (c) EQA-RT-Unseen

Figure 6: Data statistics of the training set (EQA-RT-Train) and two test sets (EQA-RT-Seen and EQA-RT-Unseen). The scenes in EQA-RT-Seen overlap with EQA-RT-Train, while the scenes in EQA-RT-Unseen are not present in the training set.

| | Step | Tool | Length (m) |
|---|---|---|---|
| EQA-RT-Train | 12.74 | 12.39 | 13.13 |
| EQA-RT-Seen | 12.13 | 11.76 | 12.38 |
| EQA-RT-Unseen | 12.56 | 12.20 | 12.71 |
| EQA-RT | 12.70 | 12.35 | 13.07 |

Table 5: Statistic about average exploration steps, the average number of tools used per task, and the average exploration length.

In addition, as shown in Table 5, we report the exact values of the average exploration steps, the average number of tools used per task, and the average exploration length across different sets. By comparing the statistics of EQA-RT-Seen and EQA-RT-Unseen, it can also be inferred that the tasks in EQA-RT-Unseen are more challenging.

## C Metric Details

We evaluate object recall, path efficiency, setting-specific answer quality, and evidence acquisition using $recall@D$, $e_{path}@D$, closed-vocabulary accuracy, open-vocabulary LLM-Match, and evidence coverage. We distinguish binary closed-vocabulary accuracy from graded open-vocabulary answer quality.

**Closed-Vocabulary Accuracy.** For closed-vocabulary evaluation, $q_i = \mathbb{I}[\hat{A}_i = A_i]$, where $\hat{A}_i$ must specify a unique option and invalid or missing answers receive zero. The evaluation protocol requires a single option letter rather than matching a substring of an option's text. Closed-vocabulary accuracy is $100\,N^{-1}\sum_i q_i$ with binary $q_i$.

**Open-Vocabulary LLM-Match.** For open-vocabulary evaluation on EQA-RT, we use Qwen3-VL-8B as a fixed text evaluator. It receives the question, reference answer, and generated answer and produces a semantic-correctness score $\sigma_i \in \{0, 1, 2, 3, 4, 5\}$; missing answers receive zero and $q_i = \sigma_i/5$. The evaluator judges meaning, including negation, numerical values, omissions, and contradictory alternatives, rather than rewarding response length. The mean normalized open-vocabulary score is

$$\textit{LLM-Match Score} = \frac{1}{N}\sum_{i=1}^{N}\frac{\sigma_i}{5} \times 100\%. \tag{8}$$

This EQA-RT normalization is fixed across comparisons; public benchmarks retain their own prescribed scoring conventions. For training, a submitted open-vocabulary answer with semantic score $\sigma(\tau) \in \{0, 1, 2, 3, 4, 5\}$ receives $s(\tau) = 2\sigma(\tau)/5 - 1$. For submitted closed-vocabulary answers, we

assign $s(\tau) = 1$ to correct answers and $s(\tau) = -1$ to incorrect or invalid answers. These submitted-answer scores are multiplied by evidence coverage as defined in Equation 4a. Trajectories without a submitted answer receive the separate penalty $R_{\text{ans}} = -2$, without evidence scaling.

**Object Recall.** The $recall@D$ metric evaluates target-object retrieval during exploration. Let $n$ denote the number of target objects and $T$ the number of camera steps. At step $t$, the camera position is $\mathbf{p}_t \in \mathbb{R}^3$ and its yaw angle around the $y$-axis is $\theta_t$. The camera's forward unit vector is $\mathbf{f}_t = (\sin\theta_t, 0, -\cos\theta_t)$. For object $j$ at position $\mathbf{o}_j \in \mathbb{R}^3$, its distance from the camera is $d_{j,t} = \|\mathbf{o}_j - \mathbf{p}_t\|$. We define

$$recall@D = \frac{1}{n}\sum_{j=1}^{n} \max_{1\le t\le T} \left\{ \left(1 - \frac{d_{j,t}}{D}\right) \times \mathbb{I}\left[d_{j,t} \le D,\ \frac{\mathbf{f}_t \cdot (\mathbf{o}_j - \mathbf{p}_t)}{\|\mathbf{o}_j - \mathbf{p}_t\|} \ge \cos\left(\frac{\text{FOV}}{2}\right)\right] \right\}, \tag{9}$$

where $\mathbb{I}[\cdot]$ is the indicator function, $D$ is the distance threshold, and FOV is the field of view.

**Path Efficiency.** The $e_{path}@D$ metric combines answer quality, target-object recall, and exploration path length:

$$e_{path}@D = \frac{1}{N}\sum_{i=1}^{N} q_i\, recall_i@D\, \exp\left(\frac{l_i}{\max(p_i, l_i)}\right), \tag{10}$$

where $q_i$ is binary correctness in the closed-vocabulary setting and normalized semantic correctness in the open-vocabulary setting, $recall_i@D$ is the recall for task $i$, $l_i$ is its reference path length, and $p_i$ is the executed path length.

**Evidence Coverage.** We use the same evidence coverage definition as in the reinforcement learning reward (Section 3.3.2). For each evaluation question $Q_i$, let $\tau_i$ denote the executed trajectory up to episode termination, $\mathcal{F}_{Q_i}$ the required evidence facts, and $\mathcal{E}(\tau_i)$ the facts satisfied by the executed tool calls and their outputs. We construct these sets using the same question-type-specific requirements and deterministic validation rules described in Appendix D.1. The required set includes the target-specific facts and one task-level completion fact, which is satisfied only when all required target facts have been acquired. Facts are counted once by target identity and fact type, so repeated acquisition does not increase coverage. The per-trajectory coverage and the reported Evidence Coverage (EC) percentage are

$$C(\tau_i) = \frac{|\mathcal{E}(\tau_i) \cap \mathcal{F}_{Q_i}|}{|\mathcal{F}_{Q_i}|}, \qquad \text{EC}\ (\%) = \frac{100}{N}\sum_{i=1}^{N} C(\tau_i), \tag{11}$$

where $N$ is the number of evaluated tasks in the corresponding split. We average over all tasks with equal weight, including trajectories that produce an incorrect answer or terminate without submitting an answer; any valid evidence acquired by these trajectories is retained in the calculation. A trajectory satisfying no required facts receives zero coverage. Reference annotations are used only for evidence validation and are not provided to the controller.

# D More Experiments

## D.1 Implementation Details.

For `GPT-4o` data generation and verification, we use the `gpt-4o-2024-08-06` checkpoint with temperature 0.0 and top-$p$ 0.9, and a maximum of 2048 output tokens. For `Grounded-SAM`, we use the `Swin-B` checkpoint with a box threshold of 0.3 and a text threshold of 0.25. For SFT, we train Qwen3-VL-8B-Instruct on the full EQA-RT-Train split for two epochs, freezing the vision encoder

| Parameter | Configuration |
|---|---|
| Base controller | Qwen3-VL-8B-Instruct |
| Update / rollout backend | Full-parameter FSDP / vLLM |
| Advantage estimator | GDPO; no value network |
| Learning rate / KL coefficient | $10^{-6}$ / $10^{-3}$ |
| Prompts per update / rollouts per prompt | 1 / 6 |
| Rollout temperature / top-$p$ | 0.8 / 1.0 |
| RL schedule | Single-stage joint optimization |
| RL training iterations | 900 |
| Evidence / answer objective weights | 1 / 1 |
| Training task source | EQA-RT-Train |
| Task sampling | Category-based |
| Context / persistent prompt token cap | 16,384 / 12,288 |
| Episode response / normal-turn token cap | 3,072 / 768 |
| Maximum normal assistant turns | 100 |
| Position-matching tolerance | 2.0 m |

Table 6: Hyperparameters for single-stage evidence-grounded reinforcement learning. The forced final-answer turn has a separate generation allowance.

and updating the vision-language connector and language model. We use a learning rate of $2 \times 10^{-7}$ with a cosine schedule, a warmup ratio of 0.03, and no weight decay. Training uses four GPUs with a per-device batch size of one and four gradient accumulation steps, yielding an effective global batch size of 16 with sequence packing enabled. We use BF16 precision, DeepSpeed ZeRO-3, gradient checkpointing, and a maximum sequence length of 16,384 tokens.

**RL Configuration** Starting from the SFT checkpoint, we jointly optimize evidence acquisition and answer quality using GDPO with KL regularization. The reward also accounts for interaction costs to encourage efficient exploration. At each iteration, we sample one task from EQA-RT-Train using category-based sampling and generate six rollout trajectories for that task. We train for 900 iterations, corresponding to 900 task samples in total. Table 6 summarizes the training hyperparameters.

**Evidence validation.** We use a deterministic evidence tracker to replay executed tool calls and accumulate satisfied facts for the annotated targets. The required facts depend on the question type: size questions require `grounded`, `position`, and `size` for each target; distance and counting questions require `grounded` and `position`; location questions additionally require `visual`; and color, other attribute, relationship, and status questions require `grounded` and `visual`. Each of these task types also includes one task-level completion fact, which becomes satisfied when all required target facts are present.

**Grounding.** Object queries are matched to annotated target names using normalized names, token matching, and predefined aliases. A `ObjectLocation2D` output establishes grounding when it contains valid bounding boxes; if confidence scores are provided, their maximum must be at least 0.37, and if class labels are provided, at least one must match the query. Matching targets not already grounded are credited up to the number of returned boxes. A valid `ObjectLocation3D` output containing finite centers and dimensions, with at least one positive size vector, can also establish grounding for matching targets, up to the number of returned centers. Grounding alone does not require agreement with an annotated 3D position.

**Position and size.** For each name-compatible target, the tracker selects the nearest unassigned detected 3D center and credits a position fact if its Euclidean distance to the annotated target position is at most 2.0 m. Each detected center is assigned at most once within a tool call. A matched detection additionally supplies a size fact when its corresponding dimensions are finite and strictly positive. This check validates the availability of a size measurement rather than its numerical accuracy against a reference size.

**Visual evidence.** A `VisualQA` output supplies a visual fact when it passes the error checks, the target is already grounded, and the input image is associated with that target. The query must match the target name; if no target matches, the tracker instead accepts queries whose token-set overlap

covers at least half of the original question's token set, while retaining the grounding and image-association requirements. `ObjectCrop` propagates image associations from source images to their crops but does not independently satisfy a visual fact. These checks establish output provenance and query relevance, without independently verifying the semantic correctness of the visual response.

**Coverage.** Facts are stored by target identity and fact type, so repeated acquisition of the same fact does not increase coverage. For a distance question with two targets, the specification contains four target facts (grounding and position for each target) and one task-level completion fact. Acquiring both facts for one target gives $C = 2/5$; acquiring both targets' required facts also satisfies the completion fact, giving $C = 1$. The completion fact indicates that the required inputs have been acquired; it does not separately verify a reasoning operation or the final answer, whose quality is evaluated by the answer reward. Navigation calls support subsequent evidence acquisition but do not directly add facts.

**Reward details.** For a submitted answer, $\mathcal{P}_{\text{stop}} = 0.5(1 - C) + 0.3\mathbb{I}[C = 0]$. For a missing answer, $\mathcal{P}_{\text{stop}} = 0$ and $R_{\text{ans}} = -2$. The interaction cost accumulates over the trajectory as

$$\begin{aligned}\mathcal{P}(\tau) = {} & 0.2N_{\text{invalid}} + 0.1N_{\text{duplicate}} \\ & + 0.05N_{\text{no-progress}} + 0.01N_{\text{tool}} \\ & + 0.2\min\left\{\max\left(\frac{L}{L^\star} - 1, 0\right), 2\right\}.\end{aligned} \tag{12}$$

Here, $N_{\text{invalid}}$ counts execution errors, invalid code, and outputs failing the tool-specific validity checks; $N_{\text{duplicate}}$ counts calls with repeated action–argument–input-image signatures or calls explicitly rejected as duplicates. $N_{\text{no-progress}}$ counts valid 2D localization, 3D localization, and visual question-answering calls that do not increase evidence coverage; navigation and cropping are excluded from this count. $N_{\text{tool}}$ counts navigation, localization, cropping, and visual question-answering calls, excluding computation, final-answer submission, invalid code, and unknown actions. These counts can overlap. $L$ is the executed path length and $L^\star$ is the reference path length; the path term is omitted when either length is unavailable or $L^\star \leq 0$. The per-call costs accumulate rather than being averaged over trajectory length, and their total contribution to the evidence reward is $-\mathcal{P}(\tau)$. GDPO standardizes each reward dimension within the task's rollout group, combines the weighted advantages, and further normalizes them over valid response tokens.

**Advantage estimation and credit assignment.** We compute both reward dimensions once at trajectory termination from the accumulated evidence, answer quality, and interaction costs. Following Equations 5 and 6, GDPO standardizes each reward dimension within the task's rollout group and combines the two advantages with equal weights. This implements trajectory-level credit assignment: generated thought and code tokens within the same rollout share one normalized advantage, while tool observations remain conditioning context and do not contribute policy-gradient loss terms. Individual tool outputs affect this shared learning signal through the terminal evidence coverage and accumulated costs.

### D.2 Multi-choice Results

We provide options for all tasks, and there is only one correct answer among the options. Under this setting, we conducted experiments on all baselines as well as our method, and the experimental results are shown in the Table 7. For EQA-RT, we train separate controllers under the closed- and open-vocabulary settings using the same training procedure and hyperparameters: two SFT epochs followed by 900 RL iterations. We evaluate each setting using its corresponding final checkpoint.

### D.3 Tool-Use Analysis of RL

We analyze paired trajectories to examine how RL changes the controller's use of the fixed tool suite. Table 8 shows that RL increases detection and cropping calls while reducing navigation and local computation calls. The number of 2D detection calls increases by 21.2% on Seen and 27.1% on Unseen, although the fraction adding new evidence decreases slightly, suggesting that this change primarily provides more opportunities to ground relevant targets. Meanwhile, crop calls more frequently reuse previously detected boxes, increasing from 75.3% to 81.7% on Seen and from 81.8%

| Dataset | Model | $recall \uparrow$ @5 | @10 | @15 | $e_{path} \uparrow$ @5 | @10 | @15 | traj. length (m) ↓ | Acc.↑ |
|---|---|---|---|---|---|---|---|---|---|
| EQA-RT Seen | Explore-EQA (Ren et al., 2024a) | 0.06 | 0.13 | 0.15 | 0.05 | 0.12 | 0.13 | 111.9 | 35.86 |
| | FineEQA (Jiang et al., 2025) | 0.08 | 0.12 | 0.18 | 0.06 | 0.09 | 0.14 | 48.76 | 39.41 |
| | 3D-Mem (Yang et al., 2025) | 0.08 | 0.15 | 0.24 | 0.05 | 0.13 | 0.19 | 59.44 | 42.37 |
| | GraphEQA (Saxena et al., 2025) | 0.06 | 0.17 | 0.25 | 0.08 | 0.15 | 0.21 | 39.85 | 38.34 |
| | FAST-EQA (Zhang et al., 2026) | 0.12 | 0.21 | 0.29 | 0.10 | 0.19 | 0.24 | **21.73** | 45.56 |
| | Pred-EQA (Yuan et al., 2026) | 0.15 | 0.31 | 0.39 | 0.13 | 0.19 | 0.25 | 25.46 | 48.28 |
| | **Ours** | **0.15** | **0.39** | **0.47** | **0.14** | **0.20** | **0.27** | 24.94 | **54.91** |
| EQA-RT Unseen | Explore-EQA (Ren et al., 2024a) | 0.05 | 0.11 | 0.14 | 0.05 | 0.09 | 0.11 | 148.54 | 37.61 |
| | FineEQA (Jiang et al., 2025) | 0.07 | 0.12 | 0.17 | 0.05 | 0.10 | 0.14 | 64.81 | 39.29 |
| | 3D-Mem (Yang et al., 2025) | 0.05 | 0.10 | 0.15 | 0.06 | 0.14 | 0.18 | 51.25 | 42.56 |
| | GraphEQA (Saxena et al., 2025) | 0.05 | 0.10 | 0.17 | 0.06 | 0.17 | 0.27 | 37.65 | 51.64 |
| | FAST-EQA (Zhang et al., 2026) | 0.10 | 0.19 | 0.31 | 0.08 | 0.13 | 0.25 | 55.79 | 44.81 |
| | Pred-EQA (Yuan et al., 2026) | 0.13 | 0.26 | 0.39 | 0.09 | 0.21 | 0.26 | 40.35 | 48.64 |
| | **Ours** | **0.16** | **0.35** | **0.48** | **0.10** | **0.24** | **0.41** | **29.53** | **52.95** |

Table 7: Multi-choice evaluation on EQA-RT-Seen and EQA-RT-Unseen.

| Statistic | EQA-RT-Seen Before RL | EQA-RT-Seen After RL | EQA-RT-Unseen Before RL | EQA-RT-Unseen After RL |
|---|---|---|---|---|
| Navigation calls | 9,295 | 8,972 | 11,303 | 10,212 |
| `ObjectLocation2D` calls | 626 | 759 | 780 | 991 |
| `ObjectLocation3D` calls | 605 | 700 | 845 | 978 |
| `ObjectCrop` calls | 97 | 104 | 132 | 166 |
| `VisualQA` calls | 247 | 240 | 315 | 342 |
| Local computation calls | 284 | 212 | 337 | 281 |
| 2D calls adding evidence (%) | 42.8 | 41.8 | 55.6 | 52.4 |
| Crop calls reusing detected boxes (%) | 75.3 | 81.7 | 81.8 | 83.7 |
| VQA calls adding visual evidence (%) | 29.1 | 39.6 | 44.8 | 49.7 |
| 3D calls adding position evidence | 91 | 91 | 155 | 160 |

Table 8: Tool-use statistics from a paired evaluation on EQA-RT-Seen (845 tasks) and EQA-RT-Unseen (1,069 tasks). Counts are aggregated over all tasks in each split. Evidence-adding percentages use all calls to the corresponding tool as the denominator and count calls that add previously unsatisfied facts under the validation rules in Appendix D.1. Crop box reuse requires the supplied box to match a box returned by an earlier detection on the same image.

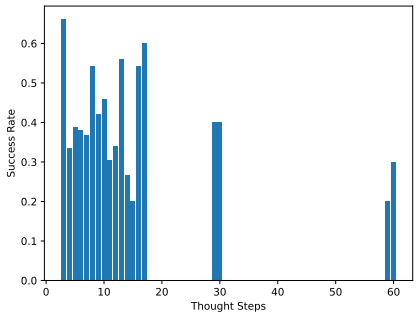
to 83.7% on Unseen. The fraction of VQA calls adding visual evidence also increases from 29.1% to 39.6% and from 44.8% to 49.7%, respectively. On Seen, this improvement occurs despite a decrease in the total number of VQA calls from 247 to 240. These patterns suggest that RL promotes both more frequent target detection and more consistent use of acquired observations in subsequent visual tool calls. In comparison, the number of 3D calls adding position evidence changes little, indicating that the clearest observed improvement concerns the coordination of detection, cropping, and visual question answering.

We conducted four analyses on the output results of our finetuned agent: (1) the impact of step count on success rate, (2) the distribution of tool calls, (3) the efficiency of our method, and (4) the ability to call untrained tools.

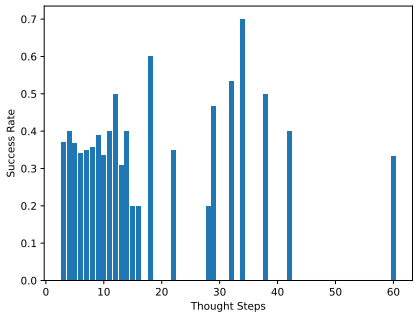
As shown in Figure 7, success rates initially increase and then decrease with the number of thought steps, indicating that more steps are not consistently associated with better performance.

Figure 8 shows different tool-use distributions across Seen and Unseen tasks, suggesting varied tool-use behavior across task settings.

We conducted an efficiency analysis of our method, and Table 9 reports the average time, LLM token, and GPU memory consumption per EQA task, broken down by module. It is worth noting

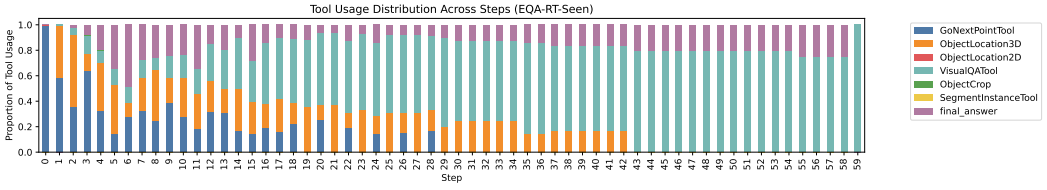



(a) Success rate with different thought steps on EQA-RT-Seen.

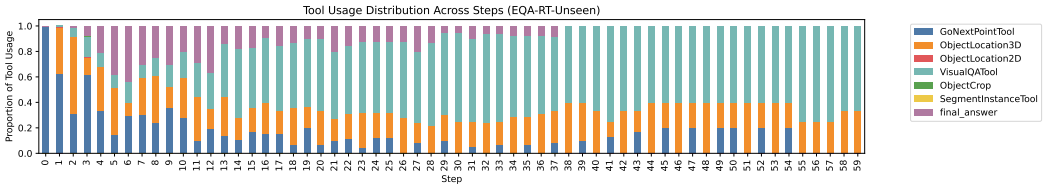



(b) Success rate with different thought steps on EQA-RT-Unseen.

Figure 7: The relationship between thought steps and success rate.




(a) Distribution of tool calls on EQA-RT-Seen.




(b) Distribution of tool calls on EQA-RT-Unseen.

Figure 8: Distribution of tool calls at different exploration steps.

that the planner is executed only once before the exploration begins, whereas the controller and executor run continuously throughout the entire exploration process.

To examine generalization to new tools, we manually construct several tasks requiring tools outside the training tool set and provide their interfaces to the controller at inference. As shown in Figure 9, the controller can invoke these additional tools in the presented examples without further supervised fine-tuning.

| | Time (s) | Token | Memory (G) |
|---|---|---|---|
| Planner | 0.52 | 118 | - |
| Controller | 25.4 | 9512.7 | 40.5 |
| Executor | 5.54 | - | - |

Table 9: Average time, token, and GPU memory consumption per EQA task, broken down by module.

### D.4 Qualitative Results

As shown in Figure 12, the untuned agent uses the same agent framework and tools, but its controller receives no task-specific supervised or reinforcement learning. Figure 12 compares our agent be-




Figure 9: Our method can call untrained tools in a given toolset.

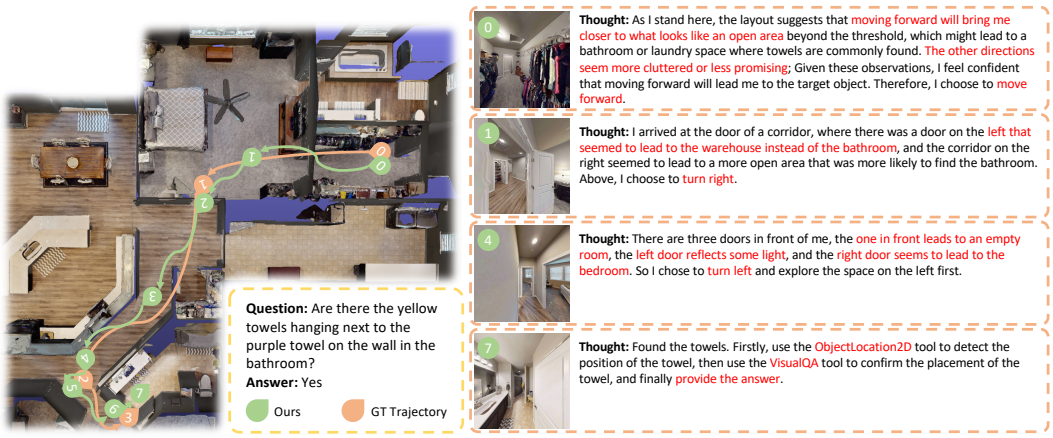



Figure 10: Illustration of how explicit reasoning guides efficient exploration, enabling our method to answer questions faster and more accurately.

fore and after SFT followed by RL on a question requiring height comparison between two objects in different parts of a scene. Before training, the agent assigns heights of 0.8 m and 2.4 m to the chair and curtain without invoking a geometric measurement tool, and terminates after three navigation steps with an incorrect answer. After training, the agent locates the curtain at step 6 and uses `ObjectLocation3D` to obtain its dimensions. At step 12, it recognizes that the area containing the other target remains unobserved and continues exploring. At step 15, it identifies the chair using the surrounding visual context and invokes the measurement tool before correctly answering that the chair is taller. This case illustrates the behavioral difference between the untrained and trained controllers, linking continued exploration and geometric evidence acquisition to answering. Although the trained agent takes a longer trajectory in this example, it gathers the information needed to avoid an unsupported answer.

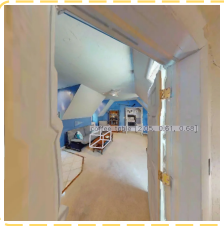
As shown in Figure 10, the decision-making process integrates spatial layout, functional space, and environmental cues to guide navigation toward the target object (towels). Each step is supported by clear reasoning, such as moving forward to approach potential bathroom space, turning right to explore a promising corridor, or turning left after excluding non-target rooms. This information-driven and reasoning-based decision paradigm ensures that the generated exploration trajectory maintains a high degree of proximity to the ground-truth trajectory, effectively validating the rationality and effectiveness of the decision-making framework in target-oriented spatial exploration tasks.

Figure 11 highlights the clear advantage of tool-driven reasoning over direct Vision-Language Models (VLMs) inference. Without tools, VLMs often fail to localize objects precisely or distinguish fine-grained attributes such as size. In contrast, by integrating these specialized tools, our method obtained critical, fine-grained information (precise object localization, clutter-free cropping, and accurate size identification) that cannot be reliably captured by direct VLMs inference on unpro-

**Question:** Which has a smaller volume, the chair at the desk under the window beside the light fixture or the coffee table in front of the couch across from the windows?



**Thought:**
Found an object that matches the description of the chair. Now, I will locate it precisely and get its bounding box and calculate the volume of chair.
**Code:**
```
bbox=ObjectLocation3D(object='chair', image_path='cache/0-3.png')
print(f'The size of chair is {bbox}')
volume = math.prod(bbox)
print(f'The volume of chair is {volume}')
```
**Observation:**
The size of chair is [0.45, 0.83, 0.42]
The volume of chair is 0.15678



**Thought:**
Found the object coffee table. Now, I will locate it precisely and get its bounding box. Calculate and save the volume of the coffee table.
**Code:**
```
bbox=ObjectLocation3D(object='coffee table', image_path='cache/1-18.png')
print(f'the size of coffee table is {bbox}')
volume = math.prod(bbox)
print(f'The volume of coffee table is {volume}')
```
**Observation:**
The size of coffee table is [2.05, 0.61, 0.68]
The volume of coffee table is 0.85034

**Thought:**
The volume of the chair is 0.15678, the volume of the coffee table is 0.85034. Compare the volumes of two object, 0.15678 < 0.85034,so the chair is smaller.
**Code:**
```
final_answer('The chair is smaller.')
```
**Observation:**
The chair has smaller volume.

**Output without tools**

**Final Answer:**
From the picture, it can be seen that the coffee table has a smaller volume.

Figure 11: Demonstration that the visual tools outperform direct VLMs inference without tools by accurately localizing and comparing object volume.

cessed images. This structured tool usage ensured the final comparison concluding that "The chair has smaller volume" was grounded in objective data, ultimately achieving a more accurate response than would be possible with VLMs alone.

### D.5 User Study

We recruited 40 participants (23 male, 17 female, aged 19–28) through the university's online participant pool. All participants were native Chinese speakers with sufficient English proficiency to understand the EQA tasks. Each participant was compensated 50 RMB for approximately 30 minutes of effort, and all participants provided informed consent prior to the study. To validate the effectiveness of our data filtering process, we randomly sampled 771 examples from the dataset before filtering and 771 examples from the dataset after filtering, resulting in a total of 1542 examples. We then asked 40 volunteers to rate these sampled examples without informing them whether each example had been filtered or not. Finally, we separately computed the scores for the data before and after filtering. We set up our user questions as follows. We mapped the options to scores from 0 to 10, and the statistical results are shown in Table 10. After applying our filtering strategy, the overall score increased by 0.925.

**User Question.**

```
Q1: Whether the image contains the object mentioned in the question?
[
    A. Fully contains.
    B. Partially contains.
    C. Does not contain at all.
]

Q2: Degree of matching between the options and the question?
[
    A. Completely matches.
```

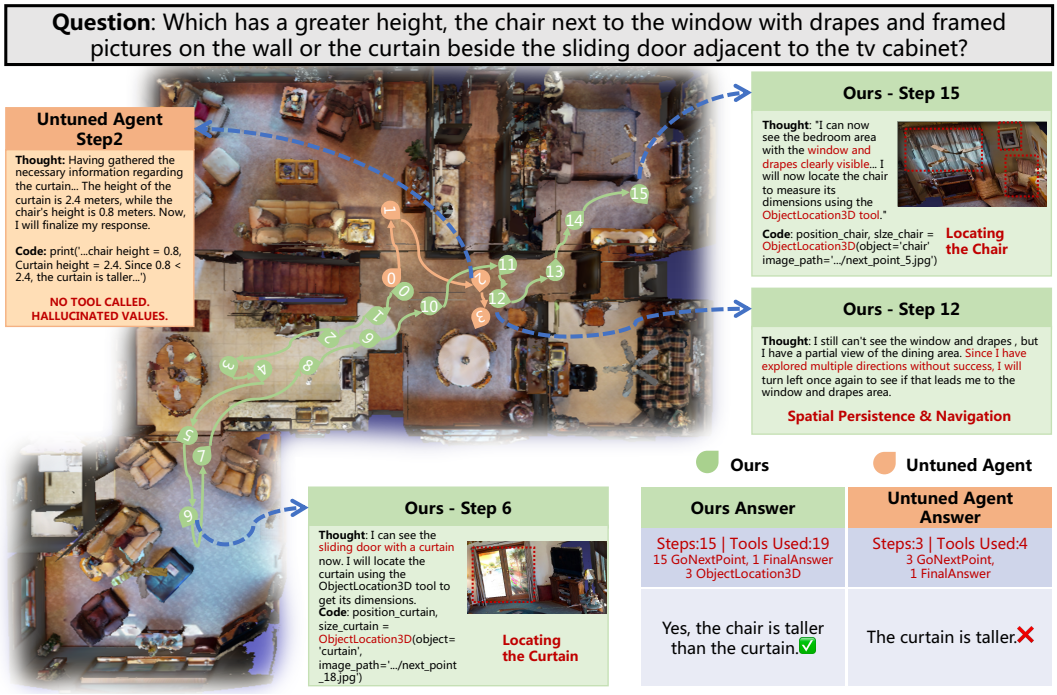



Figure 12: Qualitative comparison with the controller without task-specific tuning. Our method continues exploring and invokes geometric measurement tools before answering correctly, while the controller without task-specific tuning stops early with an unsupported, incorrect answer.

**User Question.** (continued)

```
    B. Partially matches.
    C. Does not match at all.
]

Q3: Whether the options contain the correct answer?
[
    A. Yes.
    B. No.
]

Q4: Whether the given answer is correct?
[
    A. Yes.
    B. No.
]
```

| Dataset | Q1 | Q2 | Q3 | Q4 | Total |
|---|---|---|---|---|---|
| Original Data | 8.41 | 8.93 | 8.73 | 7.38 | 8.3625 |
| Filtered Data | 9.46 | 9.45 | 9.46 | 8.78 | 9.2875 |

Table 10: Validation of the effectiveness of the data filtering strategy.

### D.6 Failure Analysis

In the multiple-choice setting, the answer options provide strong priors, allowing the agent to infer the correct answer from the textual cues even when there are imperfections in visual understanding or navigation. This reduces the impact of the underlying large model on overall performance.

Furthermore, an analysis of failure cases shows that failed episodes average 9.6 exploration steps, compared with 18.6 for successful episodes. This difference suggests that premature termination may contribute to some failures.

Finally, in the current version of our method, the planner generates a long-term navigation goal (as shown below), and the controller determines the next exploration actions based on this goal. This means that our navigation tool is not limited to purely local navigation. An interesting future direction is to directly use a navigation tool capable of planning longer trajectories without specifying every intermediate step.

**Plan Example.**

```
1. Go to the office room or workspace area to locate the ergonomic
  office chair positioned at the end of the desk. Carefully check if
  someone is currently sitting in the chair to determine if it is
  occupied or unoccupied. If needed, approach the chair to observe any
   signs of occupancy such as a personal item left on it or wear marks
  .

2. Once the observation of the chair is complete and you have
  confirmed its occupancy status, you will have the necessary
  information to answer the question regarding whether the ergonomic
  office chair is currently occupied or unoccupied.
```

## E Tools Description

As shown in Table 11, we present all the tools used and their corresponding descriptions. And then, we present the tool calls made by different models in the multiple steps for the same question (**Question: Which object is taller the chair at the corner of the dining area next to the marble table or the kitchen cabinet adjacent to the stainless steel refrigerator and facing the dining table?**).

**GPT-4o.**

```
# Step 2:
position, size = ObjectLocation3D(object='kitchen cabinet', image_path
  ='next_point_2.jpg')
print(f'The information of kitchen cabinet is: position is {position},
    size (width, height, length) is {size}.')

# Step 4:
position, size = ObjectLocation3D(object='chair', image_path='
  next_point_4.jpg')
print(f'The information of chair is: position is {position},  size (
  width, height, length) is {size}.')

# Step 5:
chair_height = 0.93
cabinet_height = 0.44
if chair_height > cabinet_height:
    answer = "A. The chair is taller"
elif cabinet_height > chair_height:
```

Table 11: Tool implementations and inference configurations.

| Tool / backend | Function and configuration |
|---|---|
| `GoNextPoint`<br>Habitat-Sim; TSDF planner | Selects an exploration waypoint and returns an RGB observation. Habitat pathfinding measures travel distance. |
| `ObjectLocation2D`<br>Grounded-SAM (Swin-B) | Localizes objects from a text query; returns boxes, labels, and scores. *Thresholds:* box 0.3; text 0.25. |
| `ObjectLocation3D`<br>DetAny3D; pose/depth processing | Returns world-coordinate centers and dimensions in width–height–length order. Centers are anchored using depth and 2D boxes when available; otherwise, camera-frame predictions are transformed into world coordinates. |
| `ObjectCrop`<br>Python image processing | Crops and saves an observed image using the supplied bounding box. No learned parameters. |
| `SegmentInstance`<br>SAM2 | Returns instance masks with an optional box prompt. An empty prompt enables automatic segmentation. |
| `VisualQA`<br>Qwen2.5-VL-3B-Instruct | Answers questions about observations or crops. *Decoding:* temperature 0.7; top-$p$ 0.8; top-$k$ 100; repetition penalty 1.05; output limit 8,192 tokens. |
| `FinalAnswer`<br>Python submission interface | Submits the controller's answer and ends the episode. No separate answering model. |

*Training.* Pretrained perception and VisualQA models remain fixed, with no task-specific tool training. VisualQA is separate from the Qwen3-VL-8B-Instruct controller optimized through SFT and RL.

**GPT-4o.** (continued)

```
    answer = "B. The kitchen cabinet is taller"
else:
    answer = "C. They are the same height"
print(answer)
```

**Qwen3-VL.**

```
# Step 2:
position, size = ObjectLocation3D(object='kitchen cabinet', image_path
  ='next_point_2.jpg')
print(f'The information of kitchen cabinet is: position is {position},
    size (Length, width, height) is {size}.')

# Step 5:
position, size = ObjectLocation3D(object='chair', image_path='
  next_point_5.jpg')
print(f'The information of chair is: position is {position},  size (
  Length, width, height) is {size}.')

# Step 6:
print('The height of the chair is 0.94, and the height of the kitchen
  cabinet is 0.5. Since 0.5 < 0.94, the chair is taller.')
```

## F Task Description

As shown in Table 12, we introduce the specific meanings of different tasks, present the tool subsets required to complete each type of task, and finally provide an example for each category of questions.

| Task Categories | Description | Tool subset | Example |
|---|---|---|---|
| Attribute-Size | Asking about the size of a single object or comparing the sizes of multiple objects. | GoNextPoint<br>ObjectLocation3D<br>FinalAnswer | Which object is taller, the rack against the wall next to the window or the lamp on the bedside table next to the bed? |
| Attribute-Color | Asking about the color of a single object or comparing the colors of multiple objects. | GoNextPoint<br>ObjectLocation2D<br>ObjectCrop<br>VisualQA<br>FinalAnswer | Do the bottle and bag on the shelf next to the doll share similar color tones? |
| Attribute-Special | Asking about other attributes of a single object (e.g., material, style). | GoNextPoint<br>ObjectLocation2D<br>ObjectCrop<br>VisualQA<br>FinalAnswer | What material is the countertop adjacent to the stove made of? |
| Counting | Asking about the number of a certain object within a specified region. | GoNextPoint<br>ObjectLocation2D<br>ObjectLocation3D<br>ObjectCrop<br>FinalAnswer | What is the total number of pillows in the master and guest bedrooms combined? |
| Distance | Asking about the distance from an object to the current position or the distance between two objects. | GoNextPoint<br>ObjectLocation3D<br>FinalAnswer | What is the distance between the heater in the bedroom and the lamp in the living room? |
| Location-Location | Asking for the locations of a category of objects (possibly multiple locations). | GoNextPoint<br>ObjectLocation3D<br>VisualQA<br>FinalAnswer | In which room is the book currently placed? |
| Location-Special | Asking for the location of a specifically described object (a particular, identifiable object). | GoNextPoint<br>ObjectLocation3D<br>VisualQA<br>FinalAnswer | In which corner of the kitchen is the large green plant inside a ceramic pot? |
| Relationship | Asking about the relationship between two objects, including spatial relations and functional relations. | GoNextPoint<br>ObjectLocation2D<br>ObjectCrop<br>VisualQA<br>FinalAnswer | Are there any pillows on the sofa in the living room? |
| Status | Asking about the state of an object. | GoNextPoint<br>ObjectLocation2D<br>ObjectLocation3D<br>ObjectCrop<br>VisualQA<br>FinalAnswer | Is the lamp in the bedroom turned on or off? |

Table 12: Descriptions, tool subsets, and examples for different task categories.

## G PROMPT DETAILS

**Plan Example.**

```
1. Go to the office room or workspace area to locate the ergonomic
  office chair positioned at the end of the desk. Carefully check if
  someone is currently sitting in the chair to determine if it is
  occupied or unoccupied. If needed, approach the chair to observe any
   signs of occupancy such as a personal item left on it or wear marks
  .

2. Once the observation of the chair is complete and you have
  confirmed its occupancy status, you will have the necessary
  information to answer the question regarding whether the ergonomic
  office chair is currently occupied or unoccupied.
```

**System Prompt.**

```
You are an embodied question-answering controller. Explore the current
   Habitat episode with the provided tools, collect evidence relevant
  to the question, and give a concise natural-language answer. No
  answer options are provided.

Available tools

<<tool_descriptions>>

Turn protocol

- In every turn, generate exactly one concise `Thought:` followed by
  exactly one executable `Code:` block. End the action with `<
  end_action>`.
- In `Thought:`, identify the evidence still missing and choose one
  immediate action. Do not recap the full trajectory.
- Make either at most one primary external tool call or one local
  Compute action per turn. The only exception is `final_answer`, which
   does not need `print`.
- A Compute action is tool-free Python that derives a result from
  values already obtained in this episode. It is valid, but cannot
  create new perceptual evidence. Do not repeat identical Compute code
   with identical input values.
- Python variables persist in the execution environment. You may
  import only: math, json, re.

Required output format

Thought: State the missing evidence and the immediate action in one or
   two sentences.
Code:
```py
result = ToolName(...)
print(result)
```<end_action>

To finish, use exactly:

Thought: The collected evidence supports the answer, so I will answer
  now.
Code:
```

**System Prompt.** (continued)

```
```py
final_answer("The chair is red.")
```<end_action>

Valid one-action examples

Thought: The target is not visible, so I will inspect a new viewpoint.
Code:
```py
image_path = GoNextPointTool(direction="turn_left")
print(image_path)
```<end_action>

Thought: I need to ground the chair in the current image.
Code:
```py
detection = ObjectLocation2D(object="chair", image_path=image_path)
print(detection)
```<end_action>

Thought: The distance comparison requires a verified 3D chair position
  .
Code:
```py
positions, sizes = ObjectLocation3D(object="chair", image_path=
  image_path)
print(positions, sizes)
```<end_action>

Thought: I need a close visual observation of the detected chair.
Code:
```py
crop_paths = ObjectCrop(bounding_box=[10, 20, 100, 160], image_path=
  image_path)
print(crop_paths)
```<end_action>

Thought: The attribute must be determined from the grounded crop.
Code:
```py
visual_answer = VisualQATool(question="Is the chair occupied?",
  image_paths=crop_paths)
print(visual_answer)
```<end_action>

Thought: Both observed positions are available, so a local calculation
   can compare them.
Code:
```py
distance = math.dist(chair_position, table_position)
print(distance)
```<end_action>

Replace every example value with an exact value from the task or an
  Observation.

Evidence policy
```

**System Prompt.** (continued)

```
- Base the answer only on observations from the current episode,
  including the initial image, later navigation images, detection
  results, crops, VQA results, and Spatial Memory. Never infer the
  answer from the question wording alone.
- Use only image paths that appeared in an Observation, and pass their
   complete path unchanged. Never invent a path, detection, bounding
  box, coordinate, size, or VQA result.
- `GoNextPointTool` accepts only `move_forward`, `turn_left`, `
  turn_right`, or `turn_around`.
- The task supplies the complete initial image path. After every new
  RGB view, the next action MUST be a question-relevant detector on
  that view before any further navigation. If additional evidence is
  still needed, navigate again after either detecting no relevant
  target or collecting the relevant evidence available in the current
  view. Continue exploring as needed to find remaining targets, count
  additional instances, or obtain a more informative viewpoint.
- `ObjectLocation2D` returns a dictionary containing `bboxes_2d`. An
  empty list means no match; it is not a tool failure. Use a returned
  box with `ObjectCrop`, then use `VisualQATool` when appearance,
  state, material, contextual relationship, or text evidence is needed
  .
- `ObjectLocation3D` returns exactly `(positions, sizes)`, where both
  values are lists. Each position is a Habitat world coordinate `[x, y
  , z]`, and each size is `[width, height, length]` in meters. `([],
  [])` means no match. Use verified 3D positions for distance,
  instance counting, and localization questions; use both positions
  and sizes for dimension comparisons. For counting, deduplicate
  repeated detections by their world positions.
- For size and distance questions, call `ObjectLocation3D` separately
  for every compared object. For counting, call it on the counted
  class across distinct views until new world positions stop appearing
  . For location questions, first obtain `ObjectLocation3D`, then
  obtain visual context with `ObjectLocation2D` plus crop/VQA as
  needed.
- Room and relative-location questions need both a verified 3D
  position and a grounded visual observation of the surrounding
  semantic context. Coordinates alone do not identify a room name.
- Do not repeat a tool call with exactly the same arguments. Do not
  call irrelevant tools. Use Spatial Memory to retain detections
  across viewpoints.
- Use Compute for comparisons, distances, counting, aggregation, or
  other transformations of observed values. Compute results are not
  evidence by themselves; the source values must come from valid
  episode Observations.
- Follow the Evidence checklist and Recommended next actions in
  Spatial Memory. A repeated call is rejected by the controller and
  cannot produce new evidence.
- Call `final_answer` as soon as the collected evidence supports an
  answer. Pass a concise, self-contained natural-language answer, not
  an option letter. State uncertainty when evidence is insufficient.
```

**Plan Generation Prompt for Offline Data Construction.**

```
You are an assistant that generates observation plans to answer
  questions about objects in a scene.

Given:
    - A question, which asks about one or more objects and their
  properties or relations in the current scene.
    - An object order, which specifies the order in which to find and
  observe the objects.

Your task is to:
1. Identify the objects mentioned in the question.
2. For each object, infer which room or area it is likely located in,
  based on common sense (e.g., a fridge is likely in the kitchen or
  dining area).
3. Generate a step-by-step plan to locate and observe the objects,
  strictly following the given object order.
4. The plan should explicitly:
    - Mention where to look for each object (the inferred room or area
  ).
    - Describe how to observe the relevant properties of each object
  to gather the information needed to answer the question.

Input Format
The given question is: <<QUERY>>
The object order is: <<TRAJECTORY>>

Example Input:
Question: Is the color of the chair against the wall, near large
  windows with decorative frames more saturated than the color of the
  fridge adjacent to the oven and dishwasher, next to the window?

Object order: chair -> fridge

Example Output:
Plan:
1. Go to the living room or dining area and locate the chair that is
  against the wall, near the large windows with decorative frames.
  Take a clear photo of the chair to capture its color and appearance
  for later comparison.
2. Go to the kitchen or dining area and locate the fridge that is
  adjacent to the oven and dishwasher, next to the window. Take a
  clear photo of the fridge to capture its color and appearance for
  later comparison.
3. After collecting the photos, compare the saturation of the chair's
  color and the fridge's color to determine which one is more
  saturated.

Now, process the following input and output a plan that adheres to the
   given object order, includes reasonable guesses about where to find
   each object, and gathers the necessary information to answer the
  question.
```

**The prompt of question generation.**

```
Your task is to generate a question.

The question must be based on the following valid types:
---
Question Type: <<question type>>
<<type description>>
Use formats like:
-> <<example>>

<<condition>>
---
Each block must include the following fields:
    - `Question`: a clearly worded question.
    - `Options`: exactly 4 options, labeled A to D.
    - `Answer`: the answer of the question.
---
---
    - Do not include words directly related to the answer in the
  options.
    - Do not repeat the same list in multiple options.
    - Do not include explanation or any extra commentary, just the
  question block in the defined format.
---
Now, given some images about <<object categories>>, please output
  following output format for the various question:
```
Question: [your attribute question here]
Options: [A. option1; B. option2; C. option3; D. option4]
Answer: [answer]
---
Question: [your attribute question here]
Options: [A. option1; B. option2; C. option3; D. option4]
Answer: [answer]
```
```

**The prompt of trajectory generation.**

```
Prompt: Visual Question Reasoning in 3D Environment

You are an intelligent embodied agent tasked with answering a visual
  question by exploring a 3D environment.
You receive the trajectory one step at a time, and at each step you
  will be told:
**The user question**
The current trajectory step (e.g., "0-0"): which includes an image and
   positional data.
A flag found: whether a target object has been found at this step (
  true or false).
The current found object: provided only if found is true.
A flag all_found: whether all required objects have already been found
   so far (true or false).

Your job is to output your reasoning for the current step as a JSON
  array of reasoning triples, where each triple consists of:
```

**The prompt of trajectory generation.** (continued)

```
    "thought": a short natural language explanation of what you are
  doing or thinking at this step
    "code": a Python-style function call representing the action or
  tool you are using at this moment
    "observation": the expected result of executing the code

**Reasoning Rules**
At each step:
    Case: found == false. You have not yet found any target object.
Action:
    You are still navigating: invoke GoNextPointTool() to move to the
  next point.
Output:
    Exactly one reasoning triple, with GoNextPointTool().
    Case: found == true and all_found == false
    You have found a target object, but not all required objects yet.

Actions (in order):
    1. Use at least one appropriate analysis tool (from the list below
  ) to identify and locate the found target object.
    2. After locating it, you must crop its region and save it using
  ObjectCrop() or similar.
    3. Finally, you must still call GoNextPointTool() to continue
  exploring.
Output:
At least three reasoning triples, in this order:
    - analyze/locate the target object.
    - crop & save the object view.
    - navigate to the next point.
Case: found == true and all_found == true.
    You have now found all required objects. Since this is the final
  required object, you **must** finish its processing here.

Actions (in order):
    What you MUST do now:
        Since all_found == true, you are NOT allowed to navigate to
  any next point, and you MUST complete processing and answering at
  this point. The final tool you invoke must be final_answer().
Do NOT call GoNextPointTool() at this step, if you do, the task fails.
After processing the last object, you MUST call final_answer("{
  expected_answer}").
1. Process the current object at this location:
    - Use an appropriate analysis tool (e.g., ObjectLocation3D(),
  ObjectLocation2D(), SegmentInstanceTool(), etc.) to identify and
  locate the current target object at this point.
    - Then crop and save the object view using ObjectCrop().
    - Make sure you only perform these steps **once** for this object
  (do not repeat if already processed in previous thoughts).
2. Review the previous reasoning and actions:
    - Read previous_thoughs, which contains a summary of all
  previously located, cropped, and registered objects.
    - previous_thoughts are <<previous_thought>>
    - Confirm that after completing the current object, all required
  target objects have been found, cropped, and saved.
    - Note: all required target objects have already been found and
  this current object completes the set.
```

**The prompt of trajectory generation.** (continued)

```
    - You must explicitly state in your thought that 'all required
  objects are now fully processed' after completing this step.
3. Decide whether you now have sufficient information to answer the
  user question:
    - Since all objects have now been located, cropped, and saved, you
   **do have sufficient information.**
    - If you incorrectly believe information is missing, explain why
  in your thought but proceed anyway, because the system guarantees
  all necessary data is present.
4. Answer the question:
    - Use VisualQATool() to compare the saved views of the relevant
  objects and generate the answer.
    - Then use final_answer() to output the final answer. The expected
   answer will be provided as <<expected_answer>>, and you must format
   it in final_answer("{expected_answer}").
Output:
At least four reasoning triples, in this order:
    - analyze / locate the target object.
    - crop & save the object view.
    - use VisualQATool() to compare.
    - output final_answer().

Available Tools
You may use the following tools:
GoNextPointTool(), ObjectLocation2D(), ObjectLocation3D(), ObjectCrop
  (), SegmentInstanceTool(), VisualQATool(), final_answer()
The definition of these tools are <<tool_descriptions>>.

If answering the question requires visual comparison or details not
  available from semantics alone, use:
RegisterViewTool(image) to save the current view for later reasoning.
**Input Format**
At each step you will receive:
    User Question: <<QUERY>>
    Trajectory Data: <<TRAJECTORY>>
    found: <<FOUND>>
    current found object: <<FOUND_OBJECT>>
    all_found: <<ALL_FOUND>>
Where:
    QUERY: the user's question about the scene.
    TRAJECTORY: the current trajectory step label (e.g., "0_0") along
  with the current image and position.
    FOUND: true or false, whether a target object is found.
    ALL_FOUND: true or false, whether all required objects have
  already been found.
    FOUND_OBJECT: name of the found object.

**Output Format**
At each step you must output your reasoning in strict JSON array
  format, like this:
[
    {
        "thought": "Describe your reasoning here.",
        "code":
        ```py
            Your Python-style function call here
        ``` ,
```

**The prompt of trajectory generation.** (continued)

```
        "observation": "Expected result of executing the code."
    }
]

Notes:
At each step:
Do not skip any required actions.
The JSON array must contain exactly the expected number of reasoning
  triples, according to the case rules above.
The triples must appear in the logical order of actions.

Example Scenarios
Given the question: Which object has a more vibrant color: the purple
  sofa positioned centrally, flanked by patterned cushions and a small
   side table or the dishwasher beneath the countertop next to the
  sink?

Here are examples of expected outputs for each case:
Case: found == false
[
    {
        "thought": "Haven't found any target yet. Continue exploring
  to uncover more areas.",
        "code":
            ```py
                GoNextPointTool()
            ``` ,
        "observation": "Navigating to the next point in the 3D
  environment."
    }
]

Case: found == true and all_found == false
[
    {
        "thought": "Found an object likely matching the description.
  Locate it precisely and give its bounding box.",
        "code":
            ```py
                ObjectLocation2D(object='sofa', image_path='/path/to/
  sofa.png')
            ``` ,
        "observation": "The bounding box of this object is [x,y,z]."
    },
    {
        "thought": "Crop and save this object's region for further
  comparison.",
        "code":
            ```py
            ObjectCrop(bounding_box = [x,y,z], image_path='/path/to/
  sofa.png')
            ```,
        "observation": "Cropped sofa saved at /path/to/sofa-crop.png"
    },
    {
        "thought": "Proceed to next point to find remaining objects.",
        "code":
```

**The prompt of trajectory generation.** (continued)

```
            ```py
                GoNextPointTool()
            ``` ,
        "observation": "Navigating to the next point in the 3D
  environment."
    }
]

Case: found == true and all_found == true
[
    {
        "thought": "Found the object dishwasher. Locate it precisely
  and give its bounding box.",
        "code":
            ```py
            ObjectLocation2D(object='dishwasher', image_path='/path/to
  /dishwasher.png')
            ``` ,
        "observation": "The bounding box of this object is [x,y,z]."
    },
    {
        "thought": "Crop and save this object's region for further
  comparison.",
        "code":
            ```py
            ObjectCrop(bounding_box = [x,y,z], image_path='/path/to/
  dishwasher.png')
            ``` ,
        "observation": "Cropped dishwasher saved at /path/to/
  dishwasher-crop.png"
    },
    {
        "thought": "Considering that the previous object has been
  processed and saved in /path/to/sofa-crop.png. Moreover, the
  information is sufficient to answer this query. Now compare the two
  registered objects to answer the question.",
        "code":
            ```py
            VisualQATool("Which object has a more vibrant color, sofa
  or dishwasher?", "/path/to/sofa-crop.png", "/path/to/dishwasher-crop
  .png")
            ``` ,
        "observation": "The sofa has a more vibrant color."
    },
    {
        "thought": "Output the final answer based on the comparison.",
        "code":
            ```py
            final_answer("The sofa has a more vibrant color.")
            ``` ,
        "observation": "Final answer submitted."
    }
]

Incorrect example at final step:
[
    ...,
```

**The prompt of trajectory generation.** (continued)

```
    {
        "thought": "Proceed to next point to find remaining objects.",
        "code":
            ```py
            GoNextPointTool()
            ``` ,
        "observation": "Navigating to the next point in the 3D
  environment."
    }
]

This is WRONG. You must NOT call GoNextPointTool() when all_found==
  true.
If you follow these rules strictly, you will solve the task correctly.
If you solve the task correctly, you will receive a reward of $1
  ,000,000.

Now begin!
```

**The prompt of LLM-based answer consistency verification.**

```
You are an evaluator that determines whether a model's answer is
  semantically equivalent to the ground truth (GT) answer.

Task:
    Given:
        1. Question: <<question>>
        2. Model's Answer: <<answer>>
        3. Ground Truth Answer (GT): <<gt_answer>>
    Instructions:
        - Focus on semantic meaning, not exact wording.
        - Consider synonyms, paraphrasing, and equivalent expressions
  as correct.
        - Ignore minor differences in style, grammar, or formatting.
        - If the answers have the same essential meaning, they are
  considered "Consistent".
        - If the answers differ in meaning, even partially, they are "
  Inconsistent".
    Output format:
        Consistent
        or
        Inconsistent
Do not output anything else.
```

**The prompt of LLM-base thought dehallucination verification.**

```
You are given:
A piece of reasoning text describing a situation, object, or scene.
An image.

Your task:
1. Analyze the reasoning text and the image.
2. Determine how semantically consistent they are.
3. Provide a consistency score between 0 and 10, where:
```

**The prompt of LLM-base thought dehallucination verification.** (continued)

```
    a. 10 means the text and image are fully consistent in meaning and
   details.
    b. 0 means they are completely inconsistent.
4. Only output the score as a single number (no explanation, no extra
  text).

Input:
Reasoning: <<thought>>
Image: <<image>>

Output:
[score]
```

**The prompt of LLM-based thought reasoning verification.**

```
You are given a context, a question, and an answer.

Determine whether the answer can be logically inferred from the given
  context, without using any outside knowledge.

If the answer can be derived from the context, output Yes.
If the answer cannot be derived from the context, output No.

Respond with only "Yes" or "No", without any explanation.
Context: <<context>>
Question: <<question>>
Answer: <<answer>>
```

**The prompt of LLM-base non-key step correctness verification.**

```
You are given a context, a question, and an answer.

Determine whether the answer can be logically inferred from the given
  context, without using any outside knowledge.

If the answer can be derived from the context, output Yes.
If the answer cannot be derived from the context, output No.

Respond with only "Yes" or "No", without any explanation.
Context: <<context>>
Question: <<question>>
Answer: <<answer>>
```