Environmental Understanding Vision-Language Model for Embodied Agent

Supplementary Material

We provide additional information and experiments in the supplementary material that could not be included in the main paper due to space limitations. Sec. A describes the benchmark dataset constructed for this work. Sec. B details the prompt templates for all skills, the dataset collection pipeline, and corresponding analyses. Sec. C presents the reward function details used in the GRPO refinement stage. Sec. D outlines the pipeline used by the VLM to perform tasks, along with additional training details. Sec. E provides the prompt templates and descriptions used for skill evaluation. Finally, Sec. F details further experimental results, including LangR [11] validation evaluations, data scaling analysis, and various ablation studies (e.g., memory input and recovery steps). Additionally, it features an extensive analysis of failure cases and the impact of individual skills on task performance.

A. Benchmark Overview

A.1. ALFRED: Action Language From Realistic Environments and Directives

ALFRED [8] is an embodied instruction following benchmark designed to perform a variety of household tasks. Built on the AI2-THOR [5] simulator, the ALFRED benchmark requires agents to interact with various objects within the simulated environment to execute natural language instructions provided by users. Each task is accompanied by a high-level language instruction that states the overall goal (e.g., “Rinse off a mug and place it in the coffee maker”) as well as a low-level language instruction (e.g., “Walk to the coffee maker on the right”) that outlines the subgoals necessary for task completion. ALFRED provides large-scale data consisting of 25,743 English instructions, 8,055 expert demonstrations, and 428,322 image-action pairs. To accomplish the tasks specified in ALFRED, a vision-language model (VLM) should combine multiple skills such as instruction understanding, object recognition, and action generation. Successfully completing a task in ALFRED requires a solid understanding of the instructions as well as strong skills in visual environmental understanding.

A.2. ALFWORLD: Aligning Text and Embodied Environments for Interactive Learning

ALFWORLD [9] is a framework that combines a text-based environment (TextWorld [3]) with an embodied simulator from ALFRED [8], allowing agents to learn abstract language policies and apply them in embodied simulator tasks. For instance, when given a natural language command such as "Put a washed apple in the kitchen fridge", humans plan a rough action sequence based on the goal and refine the plan using additional visual information, such as the layout of the kitchen, before executing the final sequence of actions. Inspired by this cognitive process, ALFWORLD [9] enables agents to first learn abstract language policies in TextWorld [3] and then predict action sequences based on the embodied simulator ALFRED [8], ultimately executing physical interactions.

ALFWORLD [9] provides two primary modes of interaction. First, in the abstract text-based environment powered by TextWorld [3], agents perform various subtasks, such as object retrieval and manipulation, using textual commands, thereby acquiring linguistic priors. Second, in the embodied environment based on the ALFRED [8] simulator, the same tasks require visual input and physical interactions, where agents apply their learned abstract policies to execute actions in a realistic, visually grounded setting. This dual approach facilitates bridging the domain gap between abstract language understanding and concrete execution in an embodied environment. However, ALFWORLD operates at an abstract subgoal level (e.g., Heat an egg), requiring control over low-level action execution.

A.3. LangR: Language Rearrangement

Lang rearrangement (LangR) is a dataset proposed to support Large LAnguage model Reinforcement Learning Policy (LLaRP) [11]. It is introduced to train models for embodied visual tasks that require generalized policies learned through reinforcement learning with large language models. It is built on the Habitat 2.0 simulator and uses the ReplicaCAD [10] and YCB [1] datasets, which provide realistic scanned environments and objects with minimal domain gap from the real world. This allows agents to perform rearrangement tasks in realistic settings.

LangR includes 150,000 training and 1,000 test episodes focused on language-conditioned rearrangement. The benchmark supports complex instruction paraphrasing (e.g., “If the fridge door is open, place the drill on the table; otherwise, place the apple on the sofa”) and demonstrates strong generalization to new tasks requiring novel optimal behaviors. Similar to ALFRED [8], LangR [10] defines a discrete action space of five actions: Navigate, Pick, Place, Open, and Close. Actions are grounded using the planning domain definition language (PDDL), and object interaction depends on success conditions. For example, when the agent is instructed to navigate to a table and no table exists in the scene, the interaction results in a no-op; when a table is present, the agent is teleported to it.

Input prompt #1 (Single image)

<image>
Can you list all the objects visible in the image?

Output #1 (Object recognition)

<Visible objects>

Input prompt #2 (Single image)

<image>
Where is the **<Object>** in the image?

Output #2 (Single object detection)

I can find the <OBJECT>, and its location is described below:
[x1, y1, x2, y2]

Input prompt #3 (Single image)

<image>
In this image, where can I find the **<Object>**?

Output #3 (Multi-object detection)

I can count **<Num>** **<Object>**, and their locations are as described below:
[x1, y1, x2, y2] #1
[x1, y1, x2, y2] #2
...
[x1, y1, x2, y2] #n

Figure 1. **Template for object recognition and detection skill.**

Input prompt (Single image)

<image>
The current sub-goal to achieve is **<Subgoal>**.
The action to perform is **<Action>**, and the object to interact with is **<Object>**.
Ground one bounding box based on the information.

Output

[x1, y1, x2, y2]

Figure 2. **Template for object detection for interaction skill.**

B. Skill Dataset Construction

As described in Sec. 3.1 and Sec. 4.1 of the main paper, we provide the skill instructions for data construction and skill evaluation. We present the complete set of input prompts and output structures used to create datasets for the four skill categories. Sec. B.1 provides the prompts used for each skill and examples of the expected outputs. Sec. B.2 details the procedure for collecting failure data. Finally, Sec. B.3 provides the dataset analyses, including the distribution of the constructed dataset.

B.1. Template of Input Prompt

We provide a comprehensive template for the prompts and outputs that define the skills learned by the VLM. Based on the four primary skills described in Section 3.1 of the main paper, we categorize them into nine detailed templates.

Input prompt (Single image or multi-image)

Given a primary objective and your current surroundings, use your previous decisions and perspectives to determine your next move.
Your Main Goal: **<Main goal>**

Step Details:
<image>Step#1: **<1st subgoal>**
<image>Step#2: **<2nd subgoal>**
...
<image>Step#n: **<nth subgoal>**
<image>Current Step:

Output

<Subgoal>
<Subgoal type>
<Action Sequence>

Figure 3. **Template for subgoal task planning skill.**

The information used in each template is constructed by leveraging memory encoded for the dataset, incorporating observed images and objects from the environment, executed actions, subgoals provided for task completion, and the main goal.

Object Perception. Object perception is categorized into object recognition and object detection for interaction. Figure 1 presents the template used for object recognition. The object recognition corresponds to the results of the first input prompt and its output. To enhance the VLM's ability to recognize multiple objects within an image, we additionally configure the model to generate responses indicating the locations of single and multiple objects when given their names, as shown in the second and third outputs. Object detection for interaction is structured using the template illustrated in Figure 2. The VLM must be capable of selecting objects relevant to achieving a given task. To facilitate this, we incorporate the ongoing subgoal as an additional input, allowing the model to identify the most appropriate object among multiple candidates. Specifically, the model is trained to generate bounding boxes for objects that experts interact with during the trajectory, ensuring that the selected objects align with the task requirements.

Task Planning. Task planning is categorized into subgoal task planning (STP) and step-by-step action planning (SAP). Figure 3 presents the template used for subgoal task planning. We configure the subgoal generation in an interleaved format. The primary goal to be achieved is provided as an input. If subgoal task planning is initiated from the initial state, a single image is used as input. Otherwise, pairs of previously achieved subgoals and corresponding images are provided as input, allowing multiple images to be fed into the VLM. Based on the main goal, the achieved subgoals, their corresponding past images, and the current image, the VLM generates the next subgoal. Additionally, we configure

Input prompt (Single image or multi-image)

Given a primary objective and your current surroundings, use your previous decisions and perspectives to determine your next move.
Current main goal: **<Main goal>**
Achieved sub-goals:
1. **<1st subgoal>**
2. **<2nd subgoal>**
...
n. **<nth subgoal>**

Navigation action space: **<Navigation action space>**
Interaction action space: **<Interaction action space>**

Previous interaction actions:
<Previous action sequence of interaction type>
Previous k step's image, generated action, and location:
<image>, **<Previosu action>**, **<Previous position>**
...
Previous k-3 step's image, generated action, and location:
<image>, <Previous action>, <Previous position>

Current Step: **<Step>**
Current Location: **<Position>**
Current Image: **<image>**
Current visible objects: **<Visible objects>**
Current sub-goal to achieve: **<Current subgoal>**

Generate the next action to achieve the current sub-goal.

Output

<Action>|<Object>

Figure 4. **Template for step-by-step action planning skill.**

Input prompt #1 (Single image, Navigation)

<image>
If I do the **<Action>** in this situation, will it succeed?

Input prompt #2 (Single image, Interaction)

<image>
Will I succeed in performing the **<Action>** with the **<Object>** in the current situation? The **<Object>**'s location is at **[x1, y1, x2, y2]**.

Outputs

Type 1: **<Yes answer>**
Type 2: **<No answer>**

Figure 5. **Template for action success prediction skill.**

Input prompt #1 (Single image, Navigation)

<image>
What do you predict will happen after performing the given **<Action>** is current situation?

Input prompt #2 (Single image, Interaction)

<image>
What do you predict will happen after performing the given **<Action>** with **<Object>** in the current situation? The **<Object>**'s location is at **[x1, y1, x2, y2]**.

Outputs

Type 1: **<Understanding>**
Type 2: **<Unchanged answer>**

Figure 6. **Template for future situation captioning skill.**

Input prompt (Multi-image)

First image: **<image>**
Second image: **<image>**
What action do these images imply was taken?

Output #1 (Navigation)

<Action>
Action Success.

Output #2 (Interaction)

<Action>|<Object>, **[x1, y1, x2, y2]**
Action Success.

Output #3 (Failure)

Nothing happend.
Action Failure.

Figure 7. **Template for action grounding skill.**

the template to predict two types of subgoals (Navigation and Interaction), enabling the VLM to infer the required affordance. Furthermore, we design the model to generate an action sequence necessary to achieve the subgoal, ensuring alignment between the subgoal and the action sequence. Figure 4 presents the interleaved input prompt and the output template used for step-by-step action planning. The provided template takes as input the current main goal, achieved subgoals, the set of executable actions in the environment, past observations from previous steps, current observations, and the subgoal that needs to be achieved. Based on this input, the VLM generates the next action to be taken. This setup allows the VLM to consider past actions and generate actions aligned with the current goal. Since our focus is on interaction, we exclude action generation data related to navigation. Effective navigation requires additional spatial reasoning in the scene, which is not adequately supported by the current template, which primarily relies on recent past information as input. While we believe that VLMs should also be capable of performing navigation, we consider this an open challenge for future research.

Action Understanding. Action understanding is categorized into action success prediction (ASP), future situation captioning (FSC), and action grounding (AG). Figure 5 presents the template used to predict whether an action will succeed or fail. We distinguish templates for both interaction and navigation actions to ensure comprehensive prediction capability. In navigation, since the discrete environment in ALFRED [8] does not require additional interactions with objects or bounding boxes, the prediction focuses solely on movement feasibility. In contrast, interaction actions require bounding box information for interaction targets. To account for this, the VLM receives relevant input information and predicts the likelihood of action success. The output

Input prompt #1 (Single image)

The sub-goal you have is **<Subgoal>**.
Taken actions to achieve this sub-goal: **<Action sequence>**
The visible objects in the current state: **<Current visible objects>**
<image>
Have you achieved it?

Input prompt #2 (Multi-image)

The sub-goal you have is **<Subgoal>**.
Initial image: **<image>**
After **<Action>**#1: **<image>**
After **<Action>**#2: **<image>**
...
After **<Action>**#n: **<image>**
The visible objects in the current state: **<Current visible objects>**
Have you achieved it?

Outputs

Type 1: **<Yes answer>**
Type 2: **<No answer>**

Figure 8. **Template for subgoal recognition skill.**

Input prompt (Multi-image)

Your key objective is **<Main goal>**.
Initial image: **<image>**
<image>Achieved sub-goal#1: **<1st subgoal>**
<image>Achieved sub-goal#2: **<2nd subgoal>**
...
<image>Achieved sub-goal#n: **<nth sub-goal>**

Successfully executed interaction actions in sequence:
<Previous action sequence of interaction type>

The visible objects in the current state:
<Current visible objects>

Have you reached this main goal?

Outputs

Type 1: **<Yes answer>**
Type 2: **<No answer>**

Figure 9. **Template for main goal recognition skill.**

is structured to generate a binary response, either "Yes" or "No". Figure 6 follows a similar structure to action success prediction but differs in its directive. Instead of merely predicting success or failure, this template enables the VLM to describe the resulting changes, allowing for more detailed interpretations of action outcomes. Figure 7 provides an example template for action grounding. It takes the previous and current images as input and generates a description of the performed action. In cases where an interaction action is executed, the model detects the corresponding object and its bounding box. This capability allows the VLM to determine the success of an action autonomously by analyzing the visual differences between images and associating them with the expected interactions.

Goal Recognition. Goal recognition is categorized into

Algorithm 1 Random Exploration Pipeline

$\mathcal{E} \leftarrow$ environment with task $\mathcal{T}$
$\mathcal{N} \leftarrow$ navigation action space
$\mathcal{I} \leftarrow$ interaction action space
$\pi_\theta \leftarrow \pi_\theta(\mathcal{N} \cup \mathcal{I})$ ▷ random exploration policy
$\mathcal{M} \leftarrow \varnothing$ ▷ Memory
$\mathcal{L} \leftarrow$ Location module
$f_0 \leftarrow \mathcal{E}_{\text{init}}$ ▷ First frame
$v_0 \leftarrow \mathcal{E}_{\text{init}}$ ▷ First visible objects
$r_0 \leftarrow$ True ▷ Action result
$p_0 \leftarrow (x_0, y_0, z_0, d_0) = (0, 0, 0, UP)$ ▷ First position
$t = 1$
while $t <$ max step **do** ▷ Random Scene Exploration
  $a_t \leftarrow \pi_\theta(f_{i-1}, v_{i-1}, s_{i-1})$ ▷ Get action
  $\mathcal{E} \leftarrow \mathcal{E}(a_t)$ ▷ Update environment
  $v_t, r_t \leftarrow \mathcal{E}$
  Update state: f_t, r_t, v_t, p_t
  $\mathcal{M} \leftarrow \mathcal{M} \cup (f_t, v_t, a_t, r_t, p_t)$ ▷ Update memory
  $t \leftarrow t + 1$
end while
return $\mathcal{M}$ ▷ Return explored memory

subgoal recognition and main goal recognition. Figure 8 presents the template used for predicting whether a subgoal has been achieved. In the first prompt, the model is given the target subgoal along with a constrained set of inputs, including the sequence of executed actions, the current image, and the objects currently visible. This design helps the VLM focus on multi-step actions while considering the current image to assess subgoal completion. If the number of input images is 8 or fewer, we structure the input as pairs of the target subgoal, post-action images, and executed actions. This setup allows the model to track changes in shorter sequences, enabling a final judgment on whether the subgoal has been achieved. The response is structured as a binary output, either "Yes" or "No". Figure 9 presents the recognition template for achieving the main goal. Unlike subgoal recognition, this template is designed to consider long-horizon changes by relying solely on multiple input images. If the number of input images exceeds 9, only the most recent eight images corresponding to achieved subgoals are used as input. The omitted images are instead represented in textual form, summarizing the completed subgoals. This design ensures that the VLM predicts goal achievement based on significant interaction sequences and the agent's current observations. The output is also structured as a binary response, "Yes" or "No", helping the VLM develop an understanding of goal completion. However, we also acknowledge that this approach remains limited for addressing long-horizon tasks in different environments. Overcoming these limitations remains an open challenge for future research.

Input prompt #1 (Navigation)

You are an artificial intelligence agent in the AI2-THOR environment. You are currently highly curious about the Environment and are exploring it by observing the scenes of the current environment and performing various actions.

Your goal is to explore all of space of the scene. So, you must generate action tokens you wish to perform from the provided list of actions.

Instruction: You have to explore the current environment.

The answer template to be generated is as follows.
Action: action you want to take. (e.g., Action: MoveAhead)

Also please do not create answers other than the template and generate just only one response.
Be sure to create 'Action: ' first.
You can only generate action token in Available actions.
Available actions: **<Selected navigation actions>**

Input prompt #2 (Navigation and interaction)

You are an artificial intelligence agent in the AI2-THOR environment. You are currently highly curious about the Environment and are exploring it by observing the scenes of the current environment and performing various actions.\nYour goal is to interact with objects which in the scene while looking around and moving around. If there are large visible objects in the scene, you can interact with them or if there are no visible objects, you must move to a different location. Also, you must generate action what you want to perform from the provided action lists.

Instruction: You have to explore the current environment.
The answer template to be generated is as follows.\nAction: action you want to take.
Also please do not create answers other than the template and generate just only one response.
Be sure to generate 'Action: ' first.\n
When you generate action, you can only select the action in available interactive actions and available navigation actions.

Available navigation actions:
Available interactive actions:

Interactive action is available only when there are visible objects.
Also, Interactive action requires an object name. If you want to interact with an object, you must specify the object name. (e.g. ToggleObjectOn|'visible object')

Additional prompt (Guideline)

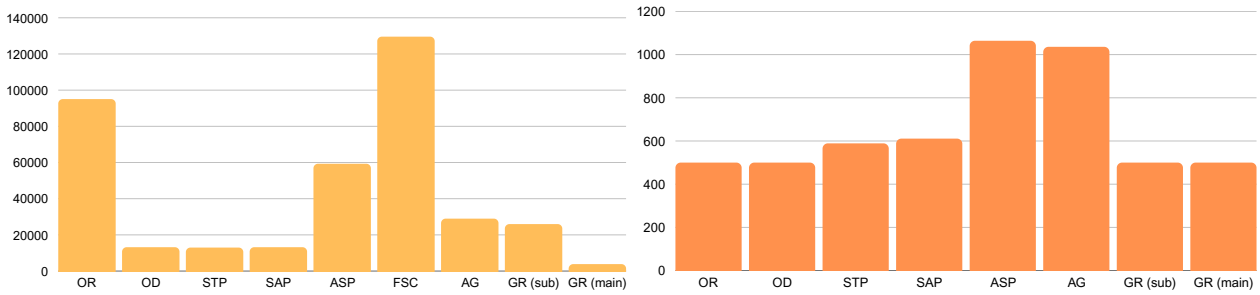
The action **<Previous action>** you previously generated has failed. Please generate a different action.

Figure 10. **Template for random exploration.** We configure prompt 1 to perform navigation randomly, and prompt 2 to perform navigation and interaction randomly. The additional prompt serves as an extra guideline to prevent generating the same action in case of failure, allowing for the collection of diverse action data.

B.2. Random Exploration

Pipeline. To enable the VLM to make informed decisions in action success prediction, it is crucial to collect both success and failure data. To this end, we introduce a random exploration pipeline for failure data collection. In ALFRED [8], the VLM performs random exploration at each scene within the training trajectory, allowing for failure data generation. Each scene undergoes exploration for up to 200 steps, following the procedure outlined in Algorithm 1. The random exploration policy is implemented using a VLM-based approach, with InternVL2.5-8B [2] as the base model. During each exploration trial, we execute 10 navigation actions followed by 30 combined navigation and interaction actions, repeating this process five times to ensure diverse interactions. Throughout the exploration, we maintain a dictionary-based memory that records key environment observations at every step, including images, objects, actions, and whether each action succeeded or failed. In LangR [11], interactions are conducted using a random policy that alternates between one navigation step and three interaction steps, repeated three times, rather than VLM-based random exploration. In addition, the memory representation excludes the position information p_t, since LangR performs teleportation during navigation. These collected data are later used for dataset construction.

Prompt. To enable zero-shot VLM to interact with the envi-



(a) Training dataset distribution.

(b) Test dataset distribution.

Figure 11. **Training and test dataset distributions.**

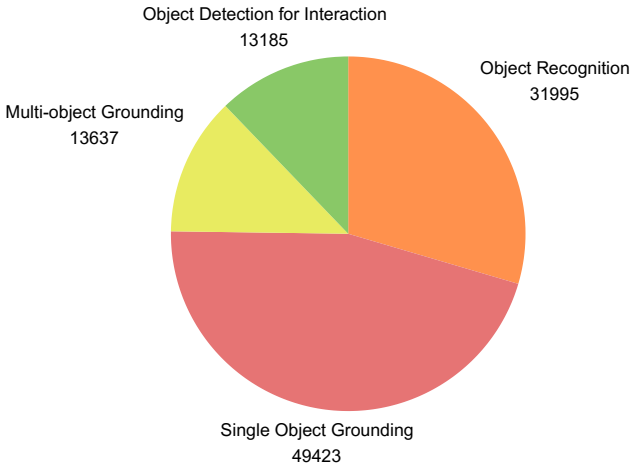


Figure 12. **Distribution of object perception.**

ronment in ALFRED [8], we utilize the prompts shown in Figure 10. In this setup, available actions are filtered based on the location module, which excludes previously visited locations from the navigation action space, ensuring that the model selects only unvisited areas. This approach prevents revisiting the same locations and promotes broader exploration. If all nearby locations have already been visited, the model is prompted to select from the remaining navigation action space. The second prompt incorporates interactive actions, allowing the VLM to perform both navigation and interaction rather than limiting it to movement alone. Additionally, we introduce guideline prompts to prevent the model from repeatedly generating the same action when it encounters a failure in the environment. This mechanism ensures that the VLM explores different actions across various locations and facilitates failure data collection.

B.3. Dataset Analyses

Figure 11a presents a graph illustrating the training and test data distribution for our four core skills. We collect both success and failure data through the ALFRED [8] benchmark dataset and our random exploration pipeline. The total number of training samples amounts to 382,065, which is used for fine-tuning the VLM. Figure 11b shows the benchmark data distribution used for skill evaluation in VLMs. To ensure a balanced evaluation across all skills, we evaluate a total of 5,300 samples. Specifically, for action understanding, we collected nearly 1,000 samples to ensure comprehensive coverage of different action types. Additionally, we provide the validation datasets used not only for training and testing but also partially during the GRPO [7] refinement stage. The ALFRED [8] validation set contains 858k samples, and the LangR [11] validation set contains 2.81M samples. Since the validation data is constructed using the same procedure as the training data, the distribution of skills and actions closely matches that of the training set.

Object Perception. Object recognition (OR) consists of 95,055 samples, while object detection (OD) includes 13,185 samples, bringing the total number of object prediction samples to 108,240. Figure 12 presents the additional data distribution for single-object detection and multi-object detection, which are subcategories of OR. Since object detection for interaction is task-oriented, its dataset size matches that of SP within the trajectory. For OR, we set a 1:2 ratio between object recognition and object detection, randomly selecting object information from the trajectory memory to construct the dataset.

Task Planning. Subgoal task planning (STP) consists of 12,981 samples, while step-by-step action planning (SAP) includes 13,185 samples. Task planning is constructed based on expert demonstrations from the ALFRED [8]. Within each trajectory, annotations with the highest scores are randomly selected to form the dataset. For SAP, we ensure consistency by fixing the subgoal instruction for each action, maintaining a structured and uniform dataset throughout the planning process.

Action Understanding. Action success prediction (ASP) consists of 59,316 samples, future situation captioning (FSC) includes 129,567 samples, and action grounding (AG) contains 28,972 samples. Figure 13 illustrates the distribution of successful data in ASP, while Figure 14 presents the distribution of failure data collected through random exploration in

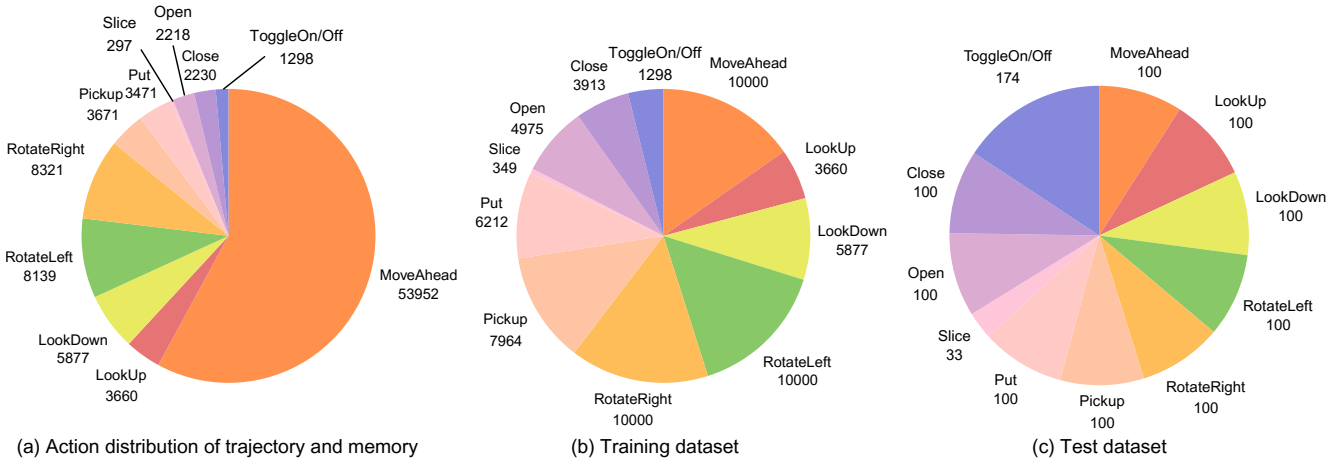


Figure 13. **Distribution of action success prediction.**

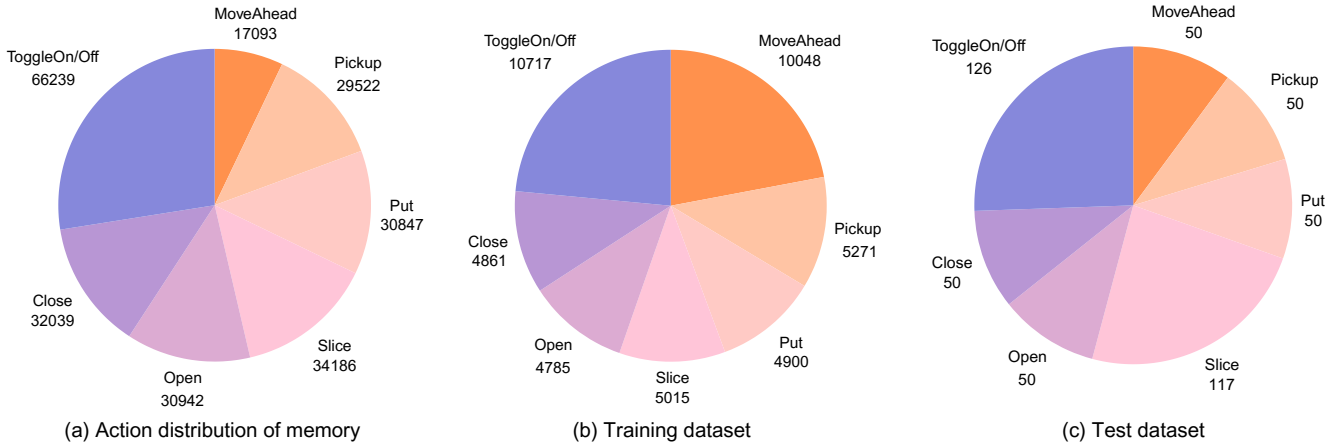
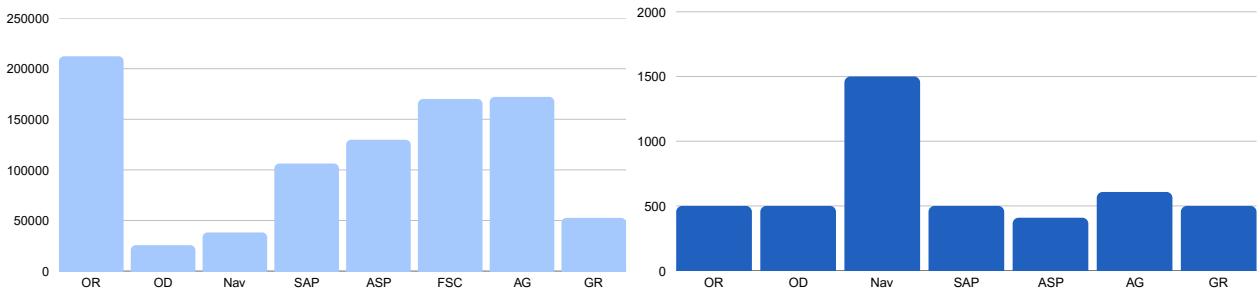


Figure 14. **Distribution of action failure prediction.**

ASP. Figure 13a shows the distribution of image-action pair data provided by ALFRED [8], along with the successful data collected from random exploration. We observe that navigation actions account for approximately 86% of the successful action distribution. To address the imbalance issue, we apply random sampling from both trajectory annotations and random exploration memory, limiting the maximum sample size to 10,000 per category, as shown in Figure 13b. Other navigation actions that only alter the agent's position are excluded, as their failure occurrences are close to zero.

For the skill evaluation dataset used in Sec. 4.2 of the main paper, Figure 13c illustrates the action distribution of the constructed dataset for ASP. To ensure a balanced evaluation across all actions, we limit the maximum number of samples per action to 100. However, due to the limited number of successful instances available in the ALFRED trajectories, SliceObject contains only 33 samples. Figure 14c shows the failure data distribution collected through random exploration, which is further adjusted to match the balanced distribution in Figure 13c. By applying these balancing procedures, our skill evaluation avoids bias toward any specific action and ensures fair assessment across all action categories.

Goal Recognition. We utilize the main goal and multiple subgoals provided in the expert demonstrations from ALFRED to collect 3,842 samples for main goal recognition and 25,962 samples for subgoal recognition. Since goal recognition is a binary classification task requiring a "Yes" or "No" response, we construct a balanced dataset with a 5:5 ratio. For cases where the goal is not achieved, subgoals related to interaction can be clearly identified, whereas those related to navigation are more challenging to distinguish. Accordingly, for navigation, we treat samples as failures when the agent does not reach the target location within half of the required action sequence steps. For interaction, if a subgoal is labeled as "No", we construct failure samples by pairing



(a) LangR training dataset distribution. (b) LangR test dataset distribution.

Figure 15. **LangR training and test dataset distributions.**

actions and image observations from timesteps prior to the next subgoal step. For the main goal, failure samples are built from timesteps except the final step, allowing the model to learn to output no when the goal has not been achieved.

LangR dataset distribution. We also present the distributions of both the skill evaluation benchmarks and the training dataset used in LangR [11]. Figure 15 illustrates the resulting dataset distribution obtained by applying our construction method to the LangR benchmarks.

C. GRPO Refinement Stage

C.1. Preliminary

Group relative policy optimization (GRPO) [7] samples multiple responses for each training instance and assigns rewards to each response according to the reward function. These rewards allow the model to estimate the relative advantage:

$$\hat{A}_{i,t} = R_i - \frac{mean(\mathbf{R})}{std(\mathbf{R})} \tag{1}$$

Where r_i denotes the reward assigned to each sampled response, and t is a token.

The relative advantage is estimated by computing normalized rewards across all tokens. Using these advantages, then the objective to be maximized is as follows:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x\sim P(X),\,\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(Y|x)}$$
$$\frac{1}{|G|}\sum_{i=1}^{|G|}\frac{1}{|\hat{y}_i|}\sum_{t=1}^{|\hat{y}_i|}\left[\min\left(r_{i,t}\hat{A}_{i,t}, c_{i,t}\hat{A}_{i,t}\right) - \beta\mathcal{D}_\theta\right], \tag{2}$$

Where

$$r_{i,t} = \frac{\pi_\theta(y_{i,t}|x, y_{i,<t})}{\pi_{\theta_{old}}(y_{i,t}|x, y_{i,<t})} \tag{3}$$
$$c_{i,t} = \text{clip}(r_{i,t}, 1-\epsilon, 1+\epsilon) \tag{4}$$
$$\mathcal{D}_\theta = \mathbb{D}_{KL}[\pi_\theta \,||\, \pi_{\text{ref}}] \tag{5}$$

We first apply supervised fine-tuning (SFT) to equip the VLM with the four proposed skills, and subsequently perform a GRPO refinement stage to correct wrong decisions that appear in the model's skill outputs. In addition, rather than targeting cases where the VLM completely fails to produce meaningful outputs, we focus on ambiguous responses, where the model alternates between correct and incorrect outputs across sampled predictions. As described in Sec. 3.3 of the main paper, we filter data based on this behavior to encourage the model to generate consistent correct outputs. Following observations from DAPO [13], exploration cases in which the model assigns extremely low probability to the correct answer show little improvement, which motivates our filtering design. This strategy further allows efficient training even with relatively small datasets.

C.2. Reward funtion

We provide additional details of the reward model described in Sec. 3.3. Our overall reward function total R_{total} is defined as follows:

$$R_{total} = R_{OP} + R_{TP} + R_{AU} + R_{GR} \tag{6}$$

Where the four component reward functions R_{OP}, R_{TP}, R_{AU}, R_{GR} correspond to the four core skills: object perception (OP), task planning (TP), action understanding (AU), and goal recognition (GR), respectively. Since each input instance corresponds to a single skill, the rewards for all other skills are set to zero, ensuring that only the reward function of the associated skill is applied.

To mitigate cases where the model's responses deviate from the predefined output structure of each skill during training, we add the following template reward to four skill reward functions:

$$r_{tem} = \begin{cases} 0 & \text{if template is correct} \\ -1 & \text{if template is incorrect} \end{cases} \tag{7}$$

Object Perception Reward. Object perception consists of two sub-skills: object recognition (OR) and object detection (OD). The OD skill is further decomposed into single-object detection and multi-object detection. Accordingly,

the reward for object perception is computed based on each corresponding sub-skill as follows:

$$R_{OP} = \begin{cases} R_{OR} & \text{if skill is object recognition} \\ R_{OD} & \text{if skill is object detection} \end{cases} \tag{8}$$

For OR, we define the reward function R_{OR} using the Jaccard index, which penalizes incorrect predictions even when all ground-truth objects are included. This encourages more precise recognition of visible objects, as follows:

$$R_{OR} = \begin{cases} -0.5, & \text{if } Y \text{ is not } \varnothing \text{ and } \hat{Y} \text{ is } \varnothing \\ -0.5, & \text{if } Y \text{ is } \varnothing \text{ and } \hat{Y} \text{ is not } \varnothing \\ 1, & \text{if } Y \cup \hat{Y} \text{ is } \varnothing \\ score_{OR}, & \text{else} \end{cases} \tag{9}$$

$$score_{OR} = \frac{Y \cap \hat{Y}}{Y \cup \hat{Y}} \tag{10}$$

Where Y denotes the list of visible objects obtained as ground-truth during skill data construction, and $\hat{Y}$ represents the object list predicted by the model.

When no objects are visible, we assign a reward of -0.5 if the model incorrectly predicts objects, and vice versa. If both the ground-truth and the model predict that no objects are visible, the reward is set to 1. For all other cases, the reward is scaled such that it reaches 1 only when the predicted number of objects exactly matches the ground-truth. This computation also accounts for multi-object cases by weighting the contribution of each object according to the number of objects involved.

For OD, we define the reward function R_{OD} using intersection over union (IoU), where a more accurate bounding box prediction yields a value closer to 1, and the reward range lies between 0 and 1. Depending on whether the task involves single-object or multi-object detection, R_{OD} is defined as follows:

$$R_{OD} = \begin{cases} \frac{1}{\hat{o}} \sum_{i=1}^{\hat{o}} r_{OD_i}, & \text{if } o = \hat{o} \\ -1 + \frac{1}{\hat{o}} \sum_{i=1}^{\hat{o}} r_{OD_i}, & \text{if } o \neq \hat{o} \end{cases} \tag{11}$$

Where o is the number of ground-truth bounding boxes, and $\hat{o}$ is the number of predicted bounding boxes.

For the multi-object case, when the number of predicted bounding boxes differs from the ground-truth, we compute the IoU reward by pairing each predicted box with the closest ground-truth box. Specifically, we form all possible pairs between ground-truth and predicted boxes, greedily select the pair with the highest IoU, remove the matched boxes, and repeat this procedure until no boxes remain. We also scale the reward using two thresholds, which determine whether the predicted IoU is sufficiently high for interaction. In addition, to focus emphasis on accurately detecting smaller objects, we apply an extra weighting factor as follows:

$$\alpha = 1.5 - \left(\frac{(x_{max} - x_{min})(y_{max} - y_{min})}{w_{img} h_{img}}\right) \tag{12}$$

$$r_{OD} = \begin{cases} \alpha, & \text{if } IoU > \tau_{\text{upper}} \\ \alpha(s_1 IoU - s_2), & \text{if } \tau_{\text{lower}} \leq IoU \leq \tau_{\text{upper}} \\ 0, & \text{if } IoU < \tau_{\text{lower}} \end{cases} \tag{13}$$

Where w_{img} and h_{img} denote the image width and height, and x_{min}, x_{max}, y_{min}, y_{max} represent the coordinates of the predicted bounding box. α is an extra weighting factor, while τ_{upper} and τ_{lower} are threshold values used for scaling the reward range. s_1 and s_2 are additional scaling parameters that normalize the reward to the range $[0, 1]$. We set s_1 to 1.5, s_2 to 0.2, τ_{upper} to 0.8, and τ_{lower} to 0.4. All images are resized to 448×448 for evaluation.

The weighting factor α increases toward 1.5 when the target object occupies a smaller region in the image, thereby enlarging the reward scaling range, and decreases toward 0.5 for larger objects. Since the VLM shows substantial performance gains on large objects after SFT, we apply this weighting to encourage the model to focus more on accurately detecting small objects.

Task Planning Reward. Task planning skill consists of two sub-skills: step-by-step action planning (SAP) and subgoal task planning (STP). To assign rewards for these two sub-skills, we define the task-planning reward R_{TP} as follows:

$$R_{TP} = \begin{cases} R_{SAP} & \text{if skill is SAP} \\ R_{STP} & \text{if skill is STP} \end{cases} \tag{14}$$

Where

$$R_{SAP} = \begin{cases} 1 & \text{if Action and Object are correct} \\ 0.5 & \text{if Action correct and Object incorrect} \\ 0.5 & \text{if Action incorrect and Object correct} \\ -1 & \text{if Action and Object are incorrect} \end{cases} \tag{15}$$

We define R_{SAP} by comparing the predicted action and object with the ground-truth for the next step. A reward of 1 is assigned when both are correct, –1 when both are incorrect, and 0.5 when either the action or the object is correct. This design encourages the model to correctly predict both components of the next-step instruction.

STP generates three elements for each subgoal which are the text, the planning type, and the action sequence. Since the action sequences implicitly overlap with the content of the text, we define the reward using the accuracy of the planning type and the accuracy of the action sequence. The

reward R_{STP} is defined as follows:

$$R_{STP} = \begin{cases} r_{type} & \text{if } \mathbf{a} \text{ is navigation} \\ \frac{1}{2}(r_{type} + \frac{\gamma}{N}\sum_{i=1}^{N} R_{SAP_i}) & \text{if } \mathbf{a} \text{ is interaction} \end{cases} \tag{16}$$

Where

$$r_{type} = \begin{cases} 1 & \text{if TP or TN} \\ -1 & \text{if FP or FN} \end{cases} \tag{17}$$

$$\gamma = \frac{\max(|A|, |\hat{A}|) - err}{\max(|A|, |\hat{A}|)} \tag{18}$$

$$err = \max(0, |\hat{A}| - |A|) + \max(0, |A| - |\hat{A}|) \tag{19}$$

$\mathbf{a}$ is the planning type, which is the affordance of the given goal. TP, TN, FP, and FN denote true positive, true negative, false positive, and false negative, respectively. r_{type} is the planning type reward, which is set to 1 when a matches the ground truth and -1 otherwise. When $\mathbf{a}$ is an interaction type, the actions belong to the action sequence and follow the same definition as SAP, so we additionally compute R_{SAP}. γ is the penalty for the length of the sequence that is predicted incorrectly. We exclude the navigation type because we do not predict actions for navigation.

These reward functions, defined by R_{SAP} and R_{STP}, guide the VLM to generate more precise actions for the next plan and correct the affordance required for the next step.

Action Understanding Reward. Action understanding skill consists of three components which are action success prediction (ASP), future situation captioning (FSC), and action grounding (AG). Because FSC is defined through a failure template, it uses the same reward function as ASP. The reward R_{AU} is defined as follows:

$$R_{AU} = \begin{cases} R_{ASP} & \text{if skill is ASP} \\ R_{AG} & \text{if skill is AG} \end{cases} \tag{20}$$

Where

$$R_{ASP} = \begin{cases} 1 & \text{if TP or TN} \\ -1 & \text{if FP or FN} \end{cases} \tag{21}$$

We treat the output as a binary classification ("Yes" or "No") and assign a reward of 1 when the predicted label matches the ground truth, and –1 otherwise.

For AG, we define the reward based on the model's ability to identify which action was executed between the previous and current image observations. The reward is assigned as follows:

$$R_{AG} = \begin{cases} r_{AG} & \text{if TN or FN or FP} \\ \frac{1}{2}(r_{AG} + R_{SAP}) & \text{if Navigation} \\ \frac{1}{3}(r_{AG} + R_{SAP} + R_{OD}) & \text{if Interaction} \end{cases} \tag{22}$$

Where

$$\mathrm{r}_{AG} = \begin{cases} 1 & \text{if TP or TN} \\ -2 & \text{if FP or FN} \end{cases} \tag{23}$$

The action-success reward r_{AG}, the action–object correctness reward R_{SAP}, and, if the action is an interaction, the bounding box reward R_{OD}. The final reward is computed by averaging over the number of reward functions. TP correspond to cases where the action is successfully executed, and a base reward of 1 is assigned. Additional rewards are incorporated depending on the action type, such as including R_{SAP} and, for interaction actions, R_{OD}. TN cases where no visual change occurs and the model correctly predicts failure, also receive a reward of 1. In contrast, FP and FN incur a larger penalty of -2, as these represent severe errors: claiming success when no visual change occurs, or predicting failure when a clear visual change is present.

These reward functions allows the model to attend not only to future changes but also to past environmental transitions, thereby enhancing its responses.

Goal Recognition Reward. Although goal recognition consists of two types, main goal and subgoal recognition, the output format for both is binary ("Yes" or "No"). Therefore, similar to the reward definitions used for r_{type}, R_{ASP}, and r_{AG}, we define the goal-recognition reward R_{GR} as follows:

$$\mathrm{R}_{GR} = \begin{cases} 1 & \text{if TP or TN} \\ -1 & \text{if FP or FN} \end{cases} \tag{24}$$

We assign a reward of 1 for correct cases (TP and TN) and a reward of -1 for incorrect cases (FP and FN). This encourages the VLM to make more consistent and accurate judgments about whether a goal has been achieved.

D. Task Evaluation Setup

D.1. Task Evaluation Interaction Pipeline

As mentioned in Sec. 4.1 in main paper, we provide a ALFRED [8] task evaluation pipeline for VLM to directly interact with the environment using its skills. To evaluate whether the learned skills improve a VLM's task performance, we use ALFRED, built on AI2-THOR [5], which provides clearly defined state changes for robust task evaluation. Since our focus is on an end-to-end agent, we avoid pre-designed or hand-engineered components and allow the VLM to operate directly from instructions and environment observations. Unlike previous studies [6, 12] that pre-plan interactions before execution, our approach enables the VLM to generate actions dynamically based on its goals and current state. To support this dynamic decision-making, we introduce an instruction-following pipeline that executes actions, updates memory, and performs tasks interactively, as shown in Algorithm 2.

We categorize the given subgoals into two types: **Navigation** and **Interaction**. Once a subgoal is given, in the

Algorithm 2 Instruction Following Pipeline

initiailze: $\mathcal{E} \leftarrow$ environment with task $\mathcal{T}$, $\mathcal{I} \leftarrow$ interaction action space, $\mathcal{M} \leftarrow \varnothing$, $\mathcal{L} \leftarrow$ Location module, $\mathcal{G} \leftarrow$ main goal, $\mathcal{S} \leftarrow$ step-by-step subgoals
$\pi_{\text{exp}} \leftarrow \pi_{\text{exp}}(\mathcal{G}, \mathcal{S})$ ▷ Navigation expert
$\pi_\theta \leftarrow \pi_\theta(\mathcal{G}, \mathcal{S})$ ▷ Init agent
$f_0 \leftarrow \mathcal{E}_{\text{init}}$ ▷ First frame
$v_0 \leftarrow OR(\pi_\theta, f_0)$ ▷ First visible objects
$p_0 \leftarrow \mathcal{L}_{\text{init}}$ ▷ First position
$\mathcal{M} \leftarrow \mathcal{M} \cup \{f_0, v_0, \varnothing, \varnothing, p_0\}$
$t \leftarrow 0$
while $\mathcal{G}$ not achieved **do**
  $sg \leftarrow \mathcal{S}(t)$
  $\sigma \leftarrow \pi_\theta(sg, \mathcal{M})$ ▷ Subgoal type
  $\delta \leftarrow$ False ▷ Subgoal success status
  if σ is interaction **then**
    while δ not achieved **do**
      Execute interaction step:
        $a_t, o_t \leftarrow SP(\pi_\theta, \mathcal{M}, \mathcal{I})$
        $b_t \leftarrow OD(\pi_\theta, f_t, a_t, o_t)$
        $\mathcal{E} \leftarrow \mathcal{E}(a_t, b_t)$
        $r_t \leftarrow \pi_\theta(f_{t-1}, f_t)$
        $t = t + 1$
        Update state: $f_t, r_t, v_t, p_t, \mathcal{M}$
      if r_t got failure **then**
        Execute recovery step:
          $a_t, o_t \leftarrow Recovery(\pi_\theta, \mathcal{M}, \mathcal{I})$
          Repeat OD and states update
      end if
      $\delta \leftarrow GP_{\text{sub}}(sg, \mathcal{M}, f_t, v_t)$
    end while
  else
    Execute navigation step:
      $\mathcal{E} \leftarrow \mathcal{E}(\pi_{\text{exp}}, \varnothing)$
      $t = t + 1$
      $r_t \leftarrow \pi_\theta(f_{t-1}, f_t)$
      Update state: $f_t, r_t, v_t, p_t, \mathcal{M}$
  end if
end while

navigation phase, the agent follows planning domain definition language (PDDL) expert actions while gathering environmental information. In the interaction phase, the agent generates actions to achieve subgoals, detects an object with a bounding box, and interacts with the environment. Also, action success is determined by whether the image changes. This process repeats until the given subgoal is completed. Through this approach, the agent utilizes the proposed skills to perceive the environment and generate actions accordingly, making decisions at each step and handling situations flexibly. This ensures that VLM-based agent can evaluate instruction-following tasks.

D.2. Training Details

We provide additional training details for the two stages used to train the VLM: SFT and GRPO.

SFT stage. We full fine-tune all VLMs for 1 epoch each using 8 A100 80GB GPUs, following the their training scripts [2, 14, 15]. The vision encoder is frozen, while the MLP and LLM components are fine-tuned. We set the batch size to 128, the max sequence length to 8192 under a cosine learning-rate schedule with bf16 precision. For InternVL3-8B and InternVL2.5-4B and 8B, we use a warmup ratio of 0.03, with learning rates of 2e-5 and 4e-5, respectively. Qwen2.5-VL-3B and 7B follows the same optimization setup but with a warmup ratio of 0.1 and a learning rate of 1e-5. All models are trained using DeepZero stage-3 optimization.

GRPO stage. We fine-tune InternVL2.5-8B and InternVL3-8B with LoRA [4] for 5 out of 10 epochs due to early stopping, using 2 A100 80GB GPUs, following DAPO [13] hyperparameter setting. We set the batch size to 64 and the maximum sequence length to 8192. Both VLMs are trained with the liger kernel enabled, bf16 precision, a LoRA rank of 8, a temperature of 1.0, top-p of 0.99, 8 sampled responses per input, a learning rate of 1e-6, $\epsilon_{\text{high}} = 0.3$, $\epsilon_{\text{low}} = 0.2$ from DAPO [13], a warmup ratio of 0.05, and DeepZero stage-3 optimization.

Input prompt #1 (Object Recognition)

You are an agent in a 3D environment simulator. Generate responses according to the following format:

Example:
n object1, m object2, ...

Precede each object with the number of times it appears in the image.
Generate the name of the object based on what is visible.
Perform grounding for all visible objects, specifying their count and names.

Object space:
You can only generate objects from the following predefined set: **<Object space>**

Do not generate objects outside this predefined object space.
Only generate object counts and names, following the given example format.

Input prompt #2 (Object Detection for Interaction)

You are an agent in a 3D environment simulator. Generate responses according to the following format:

Example:
[x1, y1, x2, y2]

x1, y1, x2, y2 are all numerical values representing the bounding box of an object.
The valid range for these values is 0 to 1000.
Only generate the bounding box coordinates as shown in the example.
Do not include any additional text or explanations.

Figure 16. **Additional instruction for object perception.**

Input prompt (Goal Prediction)

You are an agent in a 3D environment simulator. Generate responses only as "Yes" or "No" according to the given instructions.

Figure 17. **Additional instruction for goal recognition.**

Input prompt #1 (Action Prediction, Navigation)

You are an agent in a 3D environment simulator. Based on the given image, action, and, object. predict whether the action was successful.
Your response must be only "Yes" or "No".

Input prompt #2 (Action Prediction, Interaction)

You are an agent in a 3D environment simulator. Based on the given image, action, object, and bounding box, predict whether the action was successful.
Your response must be only "Yes" or "No".

[x1, y1, x2, y2] are all numerical values representing the bounding box of an object.
The valid range for these values is 0 to 1000.
0 represents the first coordinate of the image, and 1000 represents the last coordinate of the image.

Input prompt #3 (Action Grounding)

You are an agent in a 3D environment simulator. Based on two given images, you must determine which action was performed with an object.

Navigation action space:
MoveAhead, LookUp, LookDown, RotateLeft, RotateRight.
Interaction action space:
PickupObject, PutObject, SliceObject, OpenObject, CloseObject, ToggleObjectOn, ToggleObjectOff.
Object space: **<Object space>**
Response format:
Example 1 (Navigation action): NAVIGATION_ACTION
Example 2 (Interaction action with an object):
INTERACTION_ACTION|OBJECT

Rules:
Only respond with actions and objects from the provided action and object spaces.
Navigation actions must not include an object.
Interaction actions must always include an object.
Do not generate any additional text beyond the required format.

Figure 18. **Additional instruction for action prediction.**

Input prompt #1 (Subgoal Task Planning)

You are an agent in a 3D environment simulator. Generate only one sentence as a sub-goal to execute the next step based on the given main objective.

Input prompt #2 (Step-by-step Action Planning)

You are an agent in a 3D environment simulator. Generate one action and one object required to achieve the current sub-goal.

Example:
ACTION|OBJECT

The response must contain only the action and the object name in the given format.
The ACTION and OBJECT must be generated based on the given instructions.
Do not generate responses that fall outside the specified spaces.

Figure 19. **Additional instruction for subgoal task planning.**

E. Skill Evaluation Setup

E.1. Prompt for Skill Evaluation

Figures 16, 17, 18, and 19 illustrate the additional instructions included in skill prompts for object prediction, goal recognition, task planning, and action prediction, respectively, to facilitate skill evaluation as described in Section 4.2 of the main paper. Without training, VLMs lack an understanding of the environment, which may lead them to generate responses in a free-text format rather than in the structured output required for evaluation. To address this, we incorporate additional prompt instructions to guide the model in producing outputs that align with the required skill format. In Figure 16 and Figure 18, **<Object space>** refers to a list of all detectable objects in the environment, which is included in the prompt as a comma-separated input.

Input prompt (Single image)

You must identify the target object that should be picked up to complete the given instruction.

[Interactable Object Space]
<PICK_OBJECT_SPACE>

[Objects previously picked up during task completion]
<PREVIOUS_OBJECT>

[Previous your environmental understanding]
<PREVIOUS_UNDERSTANDING>

Current image: **<image>**
Your assigned task is '**<INSTRUCTION>**'.

What is the target object you should pick up to complete the instruction?

Output

<TARGET_OBJECT>

Figure 20. **Additional instruction for setting the target object.**

Input prompt (Single image)

You must answer the destination to find the target object.

[Navigation Space]
<NAVIGATION_SPACE>

Current image: **<image>**
Target object to find: **<TARGET_OBJECT>**
Where should I go?

Output

<NAVIGATION_SPACE>

Figure 21. **Additional instructions for finding the target object.**

E.2. Navigation Skills for Expansion on LangR

We additionally propose three navigation skills to support the language rearrangement task in LangR [11]. Due to the nature of the benchmark introduced in LangR, the navigate action functions teleportation, allowing us to implement custom navigation policies. To achieve the given Pick&Place goal, we decompose the navigation process into three skills: setting a target object, finding the target object, and moving

Input prompt (Single image)

You must answer the destination to place the holding object.

[Navigation Space]
<NAVIGATION_SPACE>

Current image: **<image>**
Your assigned task is '**<INSTRUCTION>**'.
Current holding object: **<HOLDING_OBJECT>**
Where should I go?

Output

<TARGET_LOCACTION>

Figure 22. **Additional instructions for destination navigation.**

the target object to the destination. Detailed instructions and templates for each skill are provided in Figures 20, 21, and 22.

Target Object Selection. In the Pick&Place task, identifying which object to pick according to the instruction is essential. Figure 20 presents a template for generating the target object based on the instruction. Following the 70 predefined object spaces in the LangR benchmark [11], we additionally provide the agent with the set of possible pickable object candidates in the prompt. To better process the task, we also include visual captions such as FSC, enabling the model to reason about the current scene. The target object to be generated is constrained to be one of the 70 predefined object spaces, consistent with the setting described in LangR benchmark skills.

Find Target Object. Once the target object is determined, the agent must search the scene to locate it. Figure 21 shows the instruction template that guides the agent to find the object using the predefined destination spaces in the LangR benchmark [11] as input for navigation. The previously selected target object is also given as input. Based on this, we define an additional policy that allows the agent to recognize and pick up the target object using an object perception skill when it becomes visible upon arrival at a destination.

Navigate to Target Destination. Figure 22 presents the template for generating the target destination, conditioned on the object that has been picked (**<HOLDING_OBJECT>**). This guides the model to infer where the object should be placed according to the instruction. Once the agent successfully navigates to the target destination, it proceeds to the interaction phase, where additional actions may be performed to complete the instruction-specific object manipulation.

F. Additional Experiment Results

We provide additional results as mentioned in Sec. 4.3 of the main paper, including failure cases, analyses of how removing individual skills affects task performance, and ablation studies on memory input and recovery step sampling count. Sec. F.1 presents extended task evaluation results, including ablations on memory input and recovery step sampling count. Sec. F.2 provides qualitative analyses showing how removing each individual skill influences performance across different tasks. Finally, Sec. F.3 presents a detailed analysis of Ours failure cases.

F.1. Additional Task Evaluation Results

Task Evaluation with an Additional InternVL2.5-8B Backbone. Table 1 presents the task performance of behavior cloning (BC), SFT, and GRPO for a weaker InternVL2.5-8B backbone, compared to the InternVL3-8B backbone evaluated in Sec. 4.2 of the main paper. Our approach improves the average task success rate by 40.79% over the BC baseline after SFT, and the additional GRPO refinement stage yields a further 2.1% improvement. While BC slightly outperforms SFT on the Clean task by 2.04%, GRPO improves performance by 6.13% over BC and increases success rates across all tasks except Pick Two. Although Pick Two decreases by 1.85% after GRPO, the result still remains significantly higher than the BC success rate of 18.52%. These results demonstrate that our method effectively enhances environmental understanding even in weaker backbones and can substantially improve performance on long-horizon tasks such as Cool and Heat, where BC struggles.

Comparison of Recovery Methods Using the Additional InternVL2.5-8B Backbone. Table 2 presents a comparison of recovery methods in both the SFT and GRPO stages using the InternVL2.5-8B backbone. Even with this weaker backbone, our sampling-based recovery step consistently outperforms the external environment-feedback method, demonstrating its effectiveness. Interestingly, the recovery step in the SFT stage yields about a 1.4% higher success rate than in the GRPO stage. This may occur because the GRPO refinement stage fine-tunes the model to assign stronger confidence to the next selected actions, which can reduce the diversity of alternative actions considered during the recovery step.

Task Performance Comparison of SFT with All Validation Data and GRPO Stage. Table 3 compares the performance of the SFT stage using the full validation dataset with that of the GRPO stage on two backbones, InternVL2.5-8B and InternVL3-8B. In the SFT stage, only about 30% of the total data (382,065 samples) is used for training, whereas the SFT-Full stage uses the entire validation set, totaling approximately 1.2M samples, to expose the model to a broader range of scenes and skills. However, the GRPO stage outperforms SFT-Full by 31.94% and 0.93% in average task success rate on InternVL2.5-8B and InternVL3-8B, respectively, despite using only about 10k samples from the 858k sample validation set. These results demonstrate that the GRPO refinement stage is effective and that correcting inconsistent responses can achieve superior performance compared to simply increasing the amount of supervised data.

Additional Skill Ablation Studies on InternVL2.5-8B and

Table 1. **Comparison of task success rates with instruction-following InternVL2.5-8B VLM Agents.**

VLM Agent	Task Success Rate						
	Avg.	Look	Pick	Pick Two	Clean	Cool	Heat
BC (InternVL2.5-8B)	38.23	81.48	63.41	18.52	59.18	0.00	0.00
Ours (SFT)	79.02	90.74	78.05	**81.48**	57.14	**95.24**	84.62
Ours (GRPO)	**81.12**	**92.59**	**81.71**	79.63	**63.27**	**95.24**	**87.18**

Table 2. **Comparison of recovery methods for task evaluation.** We evaluate task performance by comparing different recovery methods. *Env Feedback* adds external environment feedback when it failed.

Recovery Method	Success Rate		Goal Condition	
	SFT	GRPO	SFT	GRPO
Ours (InternVL2.5-8B)	79.02	81.12	85.74	87.02
w/ Env feedback	80.42	82.05	86.79	87.65
w/ Recovery Step	**83.92**	**82.52**	**88.00**	**88.00**

Table 3. **Comparison of SFT-Full and GRPO task success rate in instruction-following VLM agents.** We compare task success rates between SFT-Full and GRPO. SFT-Full refers to a model trained using the entire validation skill dataset during supervised fine-tuning.

VLM Agent	Task Success Rate						
	Avg.	Look	Pick	Pick Two	Clean	Cool	Heat
Ours (InternVL2.5-8B, SFT-Full)	49.18	**96.30**	75.61	68.52	37.76	36.51	0.00
Ours (InternVL2.5-8B, GRPO)	**81.12**	92.59	**81.71**	**79.63**	**63.27**	**95.24**	**87.18**
Ours (InternVL3-8B, SFT-Full)	84.85	87.04	**86.59**	**92.59**	69.39	**98.41**	84.62
Ours (InternVL3-8B, GRPO)	**85.78**	**90.17**	85.37	85.19	**74.49**	**98.41**	**87.18**

Table 4. **Ablation study on our proposed skills for task evaluation with InternVL2.5-8B and Qwen2.5-VL-7B backbones.** We evaluate task performance by conducting an ablation study on the proposed skills that impact environmental understanding. BC denotes the behavior cloning baseline, SR denotes the average task success rate, GC refers to goal condition rate, FSC refers to future situation captioning, AG refers to action grounding, AU refers to action understanding, TP to subgoal task planning, and GR_{Main} to main goal recognition.

VLM Agent	InternVL2.5-8B		Qwen2.5-VL-7B	
	SR	GC	SR	GC
BC	38.23	50.74	63.64	72.26
Ours	**79.02**	**85.74**	**78.32**	**84.19**
w/o FSC	73.66	79.76	73.89	80.65
w/o AG	71.10	77.51	69.46	74.32
w/o AU	75.29	82.13	77.16	84.03
w/o TP	54.55	65.97	74.59	81.00
w/o GR_{Main}	61.07	71.79	70.40	77.12

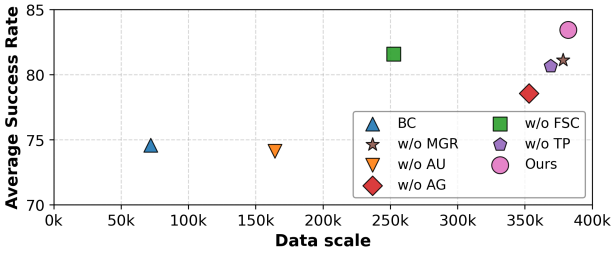


Figure 23. **Performance comparison across ablated skill variants.**

Qwen2.5-VL-7B. Table 4 shows the performance differences between the BC baseline, Ours, and Ours with each skills ablated. In all cases, even when a skill is ablated, the performance remains higher than the BC baseline. However, removing any of the key skills, FSC, AG, AU, TP, or GR_{Main}, consistently reduces performance for both backbones. Notably, TP and GR_{Main} have the strongest impact on average task performance for InternVL2.5-8B, while AG and GR_{Main} are the most influential for Qwen2.5-VL-7B. These results demonstrate that, despite architectural differences between backbones, our approach effectively enables VLMs to function as embodied agents, with each skill contributing meaningfully to overall task performance.

Impact of Recovery Step Count on Task Success Rates. Table 5 shows how varying the sampling count n in the recovery step influences task performance. When $n = 1$, fewer alternative actions are explored, resulting in lower average task success compared to $n = 5$ and $n = 10$. With $n = 5$ and $n = 10$, the average success rate improves by up to 2.33%. However, the effect differs by task type. For the long-horizon Heat task, which requires more low-level interactions, $n = 5$ does not yield improvement, whereas $n = 10$ increases the success rate by 2.56% due to the broader set of explored alternatives. In contrast, for the Pick Two task, $n = 5$ improves performance by 1.85%, but $n = 10$ leads to a 1.86% decrease, indicating that sampling too many alternatives can increase the chance of selecting an incorrect one. These results highlight the importance of choosing an appropriate sampling count n for effective

Table 5. **Comparison of task success rates with different recovery step counts.** We compare task success rates by varying the number of sampling steps used in the recovery process.

VLM Agent	Task Success Rate						
	Avg.	Look	Pick	Pick Two	Clean	Cool	Heat
Ours (InternVL3-8B, SFT)	83.45	**90.74**	**86.59**	75.93	65.31	**98.41**	91.03
w/ Recovery Step ($n = 1$)	84.85	**90.74**	**86.59**	75.93	71.43	**98.41**	91.03
w/ Recovery Step ($n = 2$)	85.31	**90.74**	**86.59**	75.93	73.47	**98.41**	91.03
w/ Recovery Step ($n = 5$)	**85.78**	**90.74**	**86.59**	**77.78**	**74.49**	**98.41**	91.03
w/ Recovery Step ($n = 10$)	**85.78**	**90.74**	**86.59**	74.07	**74.49**	**98.41**	**93.59**

Table 6. **Comparison of task success rates by varying memory length in SAP.** We compare task success rates while varying the previous-memory length k provided to the SAP skill, examining how different temporal contexts influence action generation and overall task performance.

VLM Agent	Task Success Rate						
	Avg.	Look	Pick	Pick Two	Clean	Cool	Heat
Ours (InternVL3-8B, SFT)	**83.45**	**90.74**	86.59	**75.93**	**65.31**	**98.41**	**91.03**
w/ SAP ($k = 3$)	82.75	**90.74**	**87.80**	**75.93**	**65.31**	**98.41**	85.90
w/ SAP ($k = 2$)	82.75	**90.74**	**87.80**	**75.93**	63.27	**98.41**	88.46
w/ SAP ($k = 1$)	82.28	**90.74**	**87.80**	**75.93**	62.24	**98.41**	87.18
w/ SAP ($k = 0$)	82.75	88.89	**87.80**	**75.93**	**65.31**	**98.41**	87.18

Table 7. **Comparison of success rates on Habitat Rearrangement task.**

VLM Agents	Task Success Rate
BC (InternVL3-8B)	43.60
Ours (InternVL3-8B)	**61.20**

recovery.

Comparison of Task Success Rates by Varying Memory Length in SAP. Table 6 presents an additional analysis of how task performance is affected when varying the memory size k used as input to the SAP skill during task execution. Adjusting the number of stored images alters the amount of past execution information available to the agent, and removing too much of this information leads to repeated actions. To isolate the effect of visual history, we vary only the number of stored images k while preserving all other memory components. The default setting is $k = 4$, and $k = 0$ indicates that no past images are used.

We observe that reducing k consistently lowers average task success rates. In particular, tasks requiring continuous low-level interactions, such as Heat, rely heavily on past visual observations, and performance degrades as these observations are removed. In contrast, the Pick task shows only a modest 1.21% performance drop, suggesting that simple single-step interactions do not depend strongly on historical visual information. These findings suggest the importance of effectively incorporating past visual context, and exploring architectures that more effectively leverage historical image information may further improve performance.

Analysis on Data Scaling and Skill Synergy. To examine the relationship between data scale and performance, we analyze the average success rate across varying data volumes (Fig. 23). Both our proposed EUEA framework and the behavior cloning (BC) baseline utilize the identical number of original scenes, tasks, and expert trajectories; the difference in total data size arises from the skill-specific augmentation. As shown in Fig. 23, scaling up the data does not inherently guarantee better results. For example, the *w/o AU* variant uses more than twice the data of the BC baseline ($\sim$160k vs. $\sim$75k) but shows similar performance. Additionally, the *w/o FSC* variant outperforms the *w/o AG* and *w/o TP* models despite using significantly less data ($\sim$250k vs. $\sim$370k). These findings imply that the performance gains of EUEA result from skill synergy rather than a mere increase in data size, highlighting the efficiency of our formulation even in data-limited scenarios.

Evaluation on Habitat Rearrangement Task. To demonstrate the broader applicability and generalization capabilities of our approach, we evaluated EUEA on the Habitat Rearrangement task (Table 7). The evaluation was conducted on 1,000 episodes sampled from 42 unseen scenes, excluding cases that required interacting with invisible objects. For this setup, we additionally extended the VLM to handle navigation directly. As shown in Table 7, our method outperforms

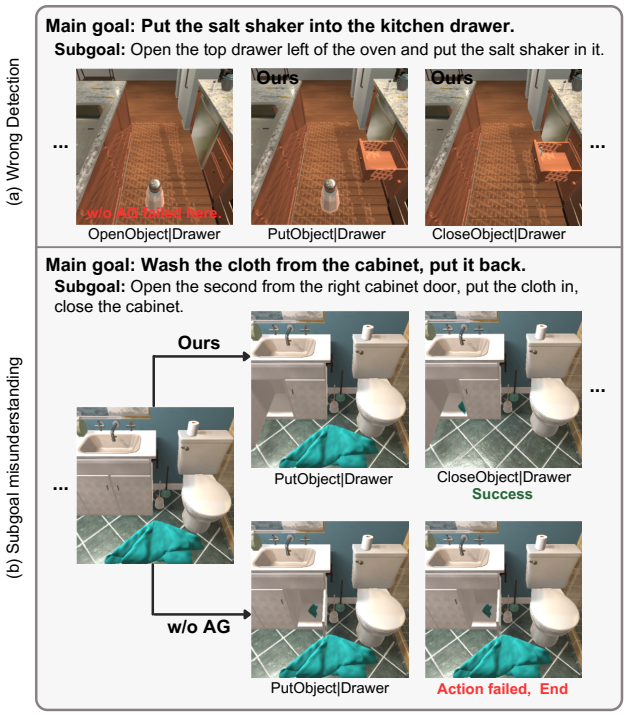


Figure 24. **Qualitative results on action grounding skill ablation.**

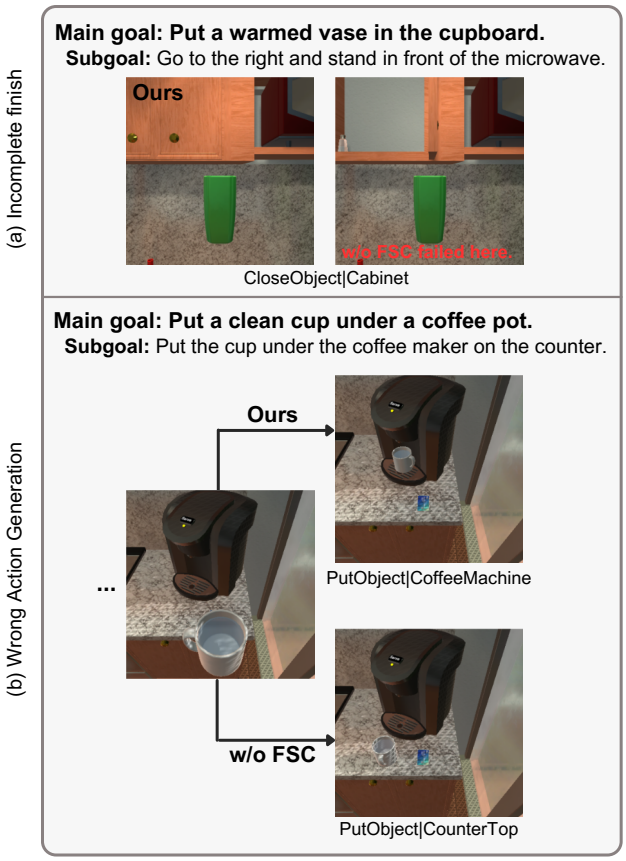


Figure 25. **Qualitative results on future situation captioning skill ablation.**

the behavior cloning (BC) baseline by an absolute margin of 17.6% (61.20% vs. 43.60%). This significant improvement indicates that the EUEA framework is effective and generalizable across different embodied benchmarks.

F.2. Qualitative Results for Skill Ablation Studies

We use InternVL3-8B as our main backbone, as in the main paper, and provide qualitative results that illustrate how task execution is affected when individual skills are removed, compared with the Ours SFT stage.

Figure 24 shows the effect of removing the Action Grounding (AG) skill. In Figure 24a, the agent without AG fails to complete the task because it cannot predict the correct bounding box while executing the OpenObject with Drawer action. In contrast, Ours successfully identifies the correct interaction target, completes the subgoal, and ultimately achieves the main goal.

Figure 24b presents another failure mode unrelated to detection errors: the agent generates an incorrect bounding box due to misunderstanding the current subgoal, leading to a sequence of failed interactions and eventual task failure. These cases highlight that although AG primarily captures what action occurred between two frames, and provides bounding boxes for interaction, it also affects the bounding box generation required for subsequent steps.

Figure 24 shows the effect of removing the Future Situation Captioning (FSC) skill. In Figure 25a, Ours correctly executes the subgoal "Go to the right and stand in front of the microwave." by picking the green vase from the cabinet and closing it. Without FSC, however, the agent fails to detect the bounding box required for the final CloseObject with Cabinet action. This omission prevents smooth progression to the next subgoal, causes the agent to collide with the open cabinet (e.g., LookUp), and eventually leads to a failure cascade when the agent later attempts to reopen the cabinet.

Figure 25b shows another example where the FSC-removed agent incorrectly predicts the object for a place action. The subgoal requires placing the cup into the coffee maker, and while our model succeeds, the FSC-removed variant misinterprets "under the coffee maker" and places the cup on the counter instead, resulting in task failure. These examples indicate that FSC supports anticipating future scene changes, helping the agent select correct objects and avoid errors.

Figure 26 illustrates the effect of removing the Task Planning (TP) skill, specifically the subgoal task planning (STP) skill that predicts subgoal-level plans. Without STP, the agent frequently repeats previous actions due to weakened planning capabilities and fails to reach the intended sub-

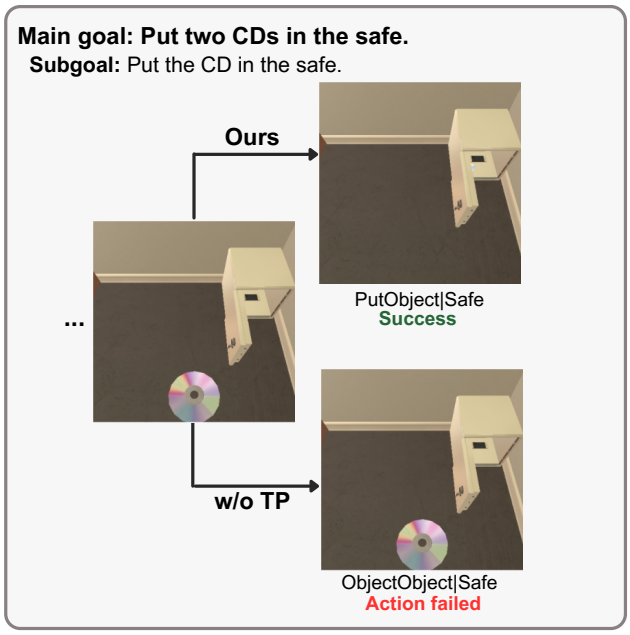


Figure 26. **Qualitative results on task planning skill ablation.**

goal. This suggests that STP skill is necessary for improving the precision of step-by-step action planning (SAP). Under the same conditions, Ours completes the task successfully, whereas the TP-ablated model fails.

We further observe that removing main-goal recognition degrades the agent's ability to track subgoal progress. Figure 27 demonstrates that when GR_{Main} is removed, the agent misjudges a completed subgoal as a failure at time step $t = 37$, triggering an unnecessary PutObject with Desk action. This additional action disrupts the correct subgoal order and results in overall task failure. In contrast, Ours correctly recognizes the successful completion of subgoals 1 and 2 and completes the final task. These findings suggest that main-goal recognition is closely linked to stable subgoal recognition and essential for long-horizon policies.

We additionally analyze failures from action understanding (AU) ablation and from the BC baseline. Removing AU produces failure patterns that mirror those observed in both AG and FSC ablations. This is consistent with the fact that AU subsumes the capabilities of AG and FSC, and therefore its removal results in the largest performance degradation among all skills. Removing BC reveals failures spanning all major categories, AG, FSC, TP, and GR_{Main}, indicating that our improving environmental understanding approach significantly affects task execution quality and is crucial for performance gains.

F.3. Failure Cases

We provide analysis of failure cases that occur in the Ours SFT stage, even when a recovery step is applied.

Qualitative Failure Analysis. Figure 28 shows cases in which Ours executes actions consistent with the given subgoal, yet still fails because the human-annotated subgoals in ALFRED [8] are not consistently aligned with the actual task conditions defined by ALFRED [8]. This indicates that annotation errors can influence task outcomes and also suggests that Ours exhibits a degree of language generalization beyond the dataset's imperfections.

Qualitative Analysis of Failure Cases. We additionally examine failure cases that Ours fails to solve. Figures 29(a), (b), and (c) illustrate scenarios where the model misinterprets the subgoal and generates an incorrect target object for the action; confuses objects with similar appearance (e.g., selecting the wrong container instead of the saltshaker); or fails to detect a bounding box required to perform the interaction. These observations highlight that, beyond environmental understanding, future work should explore more explicit reasoning over interaction preconditions, improve object recognition precision, and incorporate adjustments to navigation viewpoints to make interaction steps.

References

[1] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *arXiv preprint arXiv:1502.03143*, 2015. 1

[2] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024. 5, 11

[3] Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. In *Computer Games: 7th Workshop, CGW 2018, Held in Conjunction with the 27th International Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, July 13, 2018, Revised Selected Papers 7*, pages 41–75. Springer, 2019. 1

[4] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 11

[5] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017. 1, 10

[6] So Yeon Min, Devendra Singh Chaplot, Pradeep Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov. Film: Following instructions in language with modular methods. *arXiv preprint arXiv:2110.07342*, 2021. 10

[7] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li,

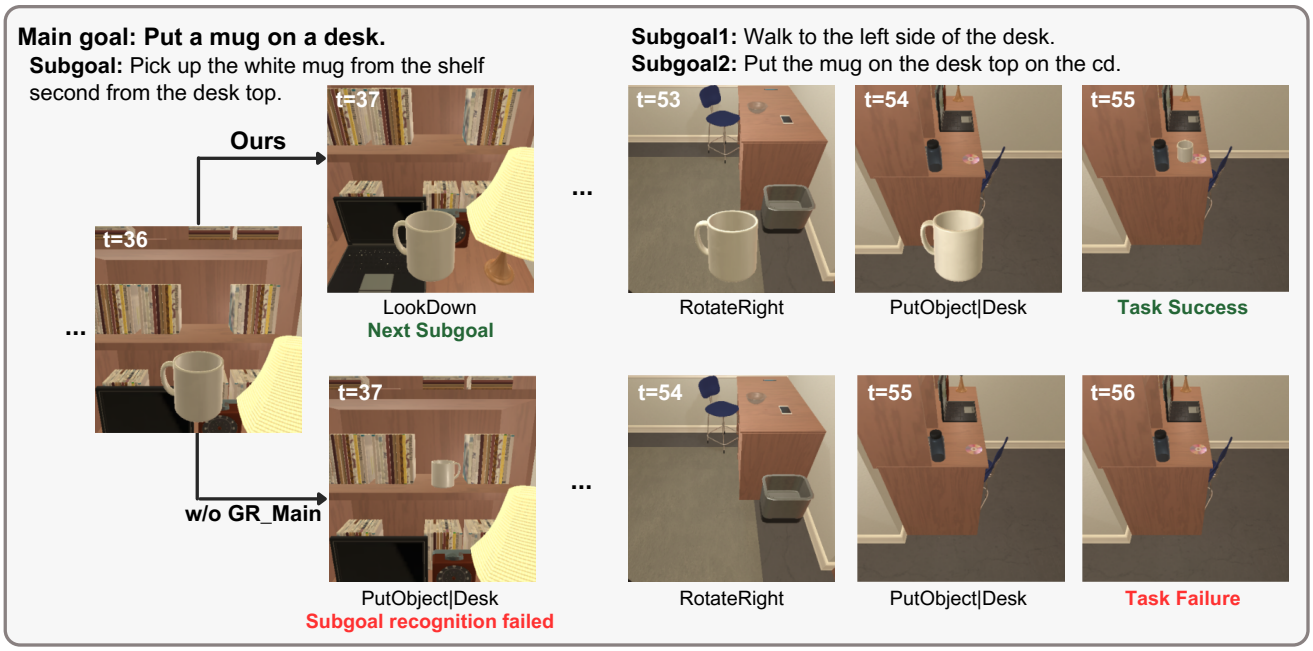


Figure 27. **Qualitative results on main goal recognition skill ablation.**

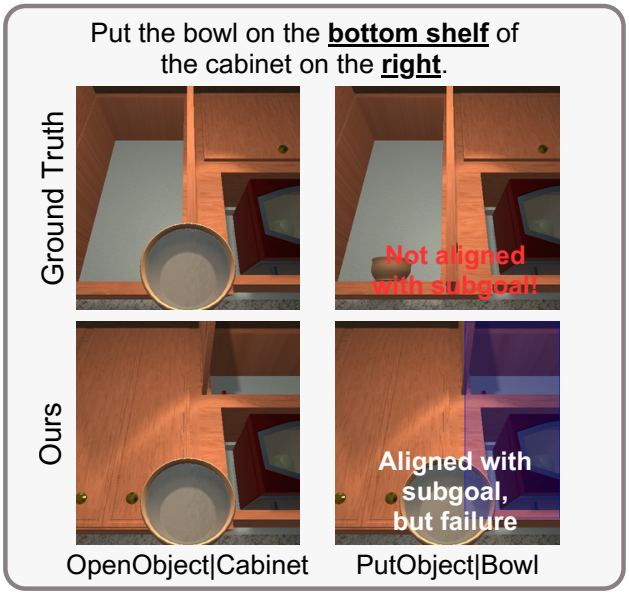


Figure 28. **Qualitative result for failure case.**

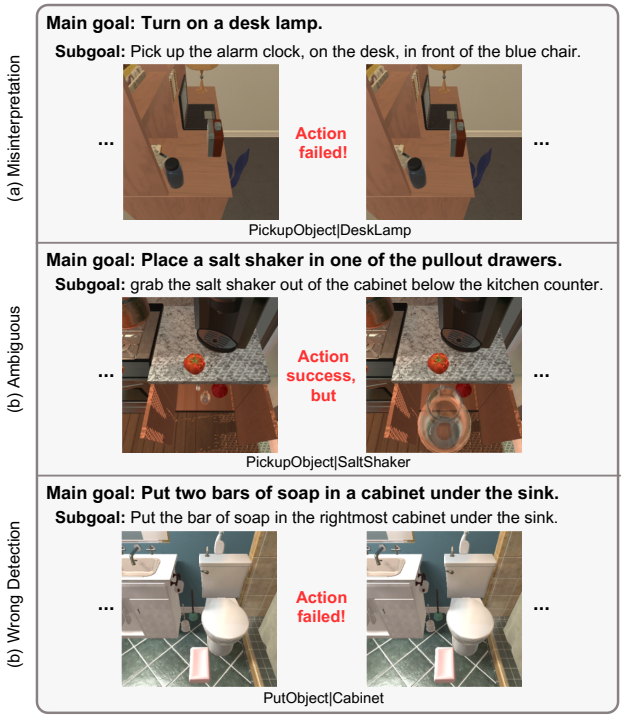


Figure 29. **Additional qualitative results for failure cases.**

Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. 6, 8

[8] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020. 1, 3, 5, 6, 7, 10, 17

[9] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020. 1

[10] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in neural information processing systems*, 34:251–266, 2021. 1

[11] Andrew Szot, Max Schwarzer, Harsh Agrawal, Bogdan Mazoure, Rin Metcalf, Walter Talbott, Natalie Mackraz, R Devon Hjelm, and Alexander T Toshev. Large language models as generalizable policies for embodied tasks. In *The Twelfth International Conference on Learning Representations*, 2023. 1, 5, 6, 8, 12, 13

[12] Xinyu Xu, Shengcheng Luo, Yanchao Yang, Yong-Lu Li, and Cewu Lu. Disco: Embodied navigation and interaction via differentiable scene semantics and dual-level control. In *European Conference on Computer Vision*, pages 108–125. Springer, 2024. 10

[13] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at scale, 2025. *URL https://arxiv.org/abs/2503.14476*, 2025. 8, 11

[14] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*, 2024. 11

[15] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Yuchen Duan, Hao Tian, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025. 11