Reinforced Reasoning for Interactive Multi-step Embodied Planning

Di Wu
Tongji University
diwu7012@gmail.com

Jiaxin Fan
Tongji University
2253538@tongji.edu.cn

Junzhe Zang
Tongji University
2250724@tongji.edu.cn

Guanbo Wang
Tsinghua University
wanggb23@mails.tsinghua.edu.cn

Wei Yin
Bank of Communications
yinw_8@bankcomm.com

Wenhao Li[†]
Tongji University
whli@tongji.edu.cn

Bo Jin[†]
Tongji University
bjin@tongji.edu.cn

Abstract

Embodied planning requires agents to make coherent multi-step decisions based on dynamic visual observations and natural language goals. While recent vision-language models (VLMs) excel at static perception tasks, they struggle in interactive environments. In this work, we introduce a reinforcement fine-tuning framework that brings R1-style reasoning enhancement into embodied planning. We adopt an offline reward paradigm to avoid costly online interaction, design a rule-based reward function tailored to multi-step action quality and optimize the policy via Generalized Reinforced Preference Optimization (GRPO). Our approach is evaluated on Embench, a recent benchmark for interactive embodied tasks, covering both in-domain and out-of-domain scenarios. Experimental results show that our method significantly outperforms models of similar or larger scale, including GPT-4o-mini and 70B+ open-source baselines, and exhibits strong generalization to unseen environments. This work highlights the potential of reinforcement-driven reasoning to advance multi-step planning in embodied AI.

1 Introduction

Embodied planning serves as a cornerstone in hierarchical embodied AI systems(38; 58), where intelligent agents must not only perceive their environment but also reason and act within it to accomplish complex, real-world tasks(12). Unlike low-level controllers that govern precise trajectory execution(57; 21), high-level planning is responsible for formulating coherent action sequences that translate complex instructions into manageable sub-tasks(52). While conventional language-based reasoning is confined to static, text-driven contexts(24; 56; 36), embodied planning operates within dynamic, interactive environments that demand sequential decision-making across multiple steps. Despite recent advancements in VLMs have demonstrated impressive capabilities in static understanding tasks(59), they exhibit substantial limitations when applied to **multi-step interactive** embodied planning. Empirical analyses in Figure1 reveal that even state-of-the-art VLMs, which excel in image captioning or visual question answering, struggle to maintain coherent and efficient decision sequences in dynamic environments(55). These shortcomings highlight a critical gap:

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Multi-Turn Interactions in Large Language Models.

effective planning in real-world embodied contexts imposes far greater demands on spatial reasoning, long-horizon coherence, and generalization capability than current VLM architectures can satisfy.

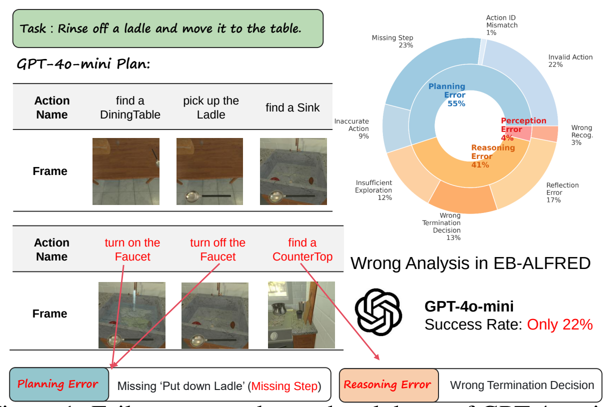


Figure 1: Failure case and error breakdown of GPT-4o-mini in the EB-ALFRED environment. **Left:** A representative task failure. **Right:** Distribution of failure types across EB-ALFRED tasks.

To address reasoning deficiencies, recent research has explored enhancing large models' cognitive abilities through dedicated reasoning frameworks(32). Notably, approaches such as DeepSeek-R1(15) have pioneered reinforcement-driven paradigms that explicitly strengthen a model's reasoning capacity via reward-guided optimization, and have achieved promising results in math and code problems. Extensions of this paradigm into multimodal contexts have begun to emerge(48), tackling tasks such as visual mathematics and diagram-based reasoning(60; 37; 27; 25). However, applying such reasoning-enhancement techniques to embodied planning remains highly challenging and underexplored due to the fundamental differences between embodied tasks and conventional reasoning benchmarks: (1) Embodied planning requires spatial perception and physical commonsense(26), whereas tasks like math or code focus purely on symbolic reasoning without grounding in dynamic environments; (2) The transition from static, single-turn QA to interactive, multi-turn decision-making(50) introduces continuous feedback loops—unlike static tasks, embodied agents must adaptively reason as each action reshapes their environment; (3) Acquiring reward signals for fine-tuning models in embodied planning tasks is inherently challenging, since executing trajectories within interactive environments to obtain feedback as reward is computationally expensive and impractical to scale, especially when bridging to real-world scenarios.

In this work, we bridge the gap by proposing a reinforcement fine-tuning framework that brings R1-style reasoning enhancement into embodied planning, enabling models to make coherent and context-aware decisions in dynamic, interactive environments. To address the challenge of reward acquisition, we introduce an offline reward formulation that scores model-generated trajectories by comparing them to expert demonstrations, then we propose a rule-based reward function that specifically designed for multi-step decision, and optimize the model using Generalized Reinforced Preference Optimization (GRPO) (36) to encourage long-horizon, goal-directed reasoning. Before reinforcement tuning, we apply supervised fine-tuning (SFT)(31) to initialize the model with structured commonsense priors. Recognizing the discrepancy between simplistic text-based simulations and the complexities of real-world physics, we conduct evaluations within Embench(55), an interactive embodied benchmark that faithfully captures environmental dynamics and agent-environment feedback loops. Experimental results demonstrate that our method significantly improves planning performance, yielding more efficient and context-aware action sequences. Moreover, our reinforcement-driven fine-tuning exhibits strong generalization across unseen tasks and environments, underscoring its potential for practical deployment in real-world embodied AI applications.

In summary, our contributions are as follows:

- We are the first to apply reinforcement fine-tuning to optimize a vision-language model for embodied planning, significantly improving the model's ability to perform coherent multi-step reasoning and decision-making in dynamic environments.
- We design a reinforcement fine-tuning strategy tailored for multi-turn embodied planning, featuring an offline reward formulation that avoids costly simulator interaction, a multi-step reward function aligned with long-horizon reasoning, and supporting mechanisms such as online data filtering to ensure training stability.
- We conduct extensive evaluation on Embench, an interactive benchmark for embodied AI, showing that our model not only outperforms comparable-scale models but also surpasses GPT-4o-mini and

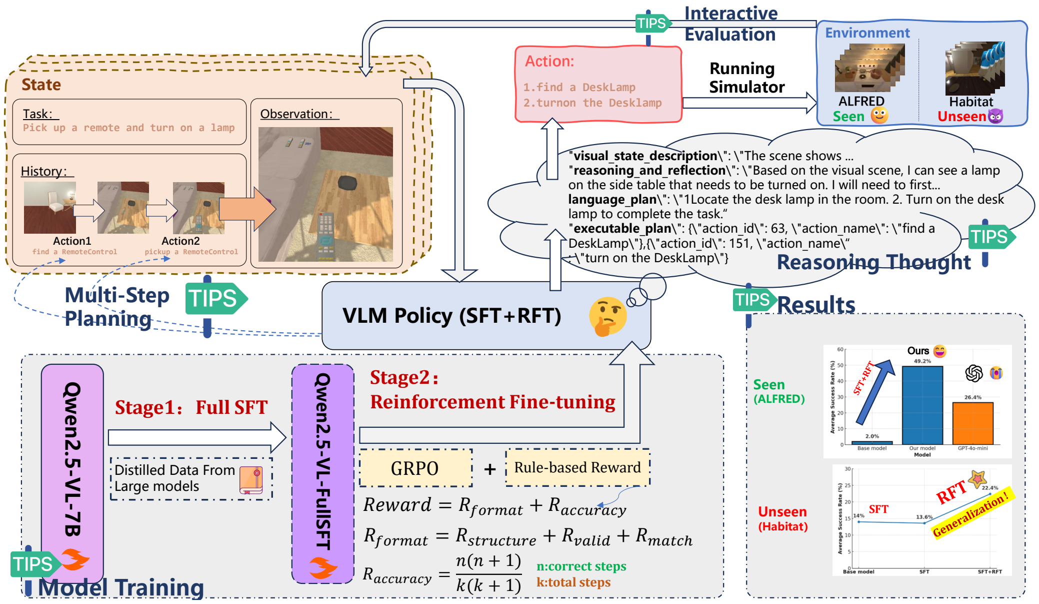


Figure 2: Overview of our proposed framework. We adopt a two-stage training paradigm consisting of supervised fine-tuning (SFT) followed by reinforcement fine-tuning (RFT) to enhance multi-step planning capabilities of the vision-language model. The final model is evaluated on **Embench**, an interactive embodied benchmark, where it achieves strong performance across both **seen** and **unseen** environments.

open-source models with more than 70B parameters. It further demonstrates strong generalization to unseen domains, validating the generality of reinforcement-based adaptation.

2 Methodology

2.1 Problem Definition

We formulate embodied task planning as a multi-turn, partially observable decision-making process, where the agent interacts with an environment through sequential actions based on visual observations. At each time step t, the agent receives an observation $o_t \in \mathcal{O}$ and executes an action $a_t \in \mathcal{A}$, forming a history

$$h_t = \{o_0, a_0, o_1, ..., o_t\}. \tag{1}$$

Given a task instruction $g \in \mathcal{G}$ described by a natural language command L, the task is associated with a set of binary goal-checking conditions $\mathcal{C}(g) = \{c_1, ..., c_k\}$ that must all be satisfied for the task to be considered successful. The agent generates a trajectory

$$e = (g, o_0, a_0, o_1, ..., o_n, a_n), \tag{2}$$

and the reward is defined as

$$r(e) = \mathbb{I}\left[\bigwedge_{c \in \mathcal{C}(g)} c = \text{True}\right], \tag{3}$$

where $\mathbb{I}[\cdot]$ is the indicator function.

We parameterize the policy π_θ using a vision-language model (VLM), which outputs an action distribution conditioned on the observation o_t, history h_t, instruction L, and a fixed prompt template P:

$$a_{t+1} \sim \pi_\theta(\cdot \mid o_t, h_t, L, P). \tag{4}$$

Our objective is to optimize θ such that the expected task success rate of sampled trajectories increases:

$$\max_\theta \ \mathbb{E}_{e\sim\pi_\theta}\left[r(e)\right]. \tag{5}$$

2.2 Reinforcing Reasoning for Embodied Planning with Offline Reward

While reinforcement fine-tuning has proven effective for improving reasoning capabilities in language models, its application to embodied planning poses unique challenges. In particular, acquiring reward signals via online interaction—where the agent executes sampled trajectories in simulators to collect feedback—is prohibitively expensive in embodied settings. Each rollout requires environment resets, step-by-step rendering, and physics simulation, which become costly at scale. Furthermore, exploration under early-stage policies often yields sparse or invalid trajectories, impeding reward signal acquisition. These issues are amplified in real-world deployment scenarios, where environment resets and safety constraints introduce additional friction.

To address these limitations, we adopt an *offline reward optimization* approach that avoids online execution. Inspired by (11), which demonstrates that comparing model-generated plans to high-quality expert trajectories can yield effective and stable policy updates, we adopt a similar offline reward formulation. Instead of collecting interactive feedback, we compute step-wise reward signals by comparing model-generated plans to expert trajectories. This design significantly reduces computational cost, improves reproducibility, and enables large-scale training without the burden of simulator interaction. Our experiments show that such supervision not only avoids the instability of online exploration but also encourages gradual policy alignment with expert behavior, ultimately leading to improved generalization and learning efficiency in multi-turn decision-making tasks.

Offline Expert Trajectory Construction. We construct our offline expert dataset based on the ALFRED benchmark (40), which provides complete ground-truth trajectories for household tasks in simulated environments. Each expert trajectory $e = (g, o_0, a_0, o_1, a_1, ..., o_k, a_k)$ is decomposed into k training samples, specifically, for each step $n \in [1, k]$, we build an input prompt L_n containing the task goal g and the preceding action history $a_{0:n-1}$. The corresponding visual observation o_n is taken from the n-th step, and the target response $\hat{a}_{n:} = \{a_n, ..., a_k\}$ includes all remaining actions. Applying this decomposition to the ALFRED dataset yields 43,898 training samples for reinforcement fine-tuning.

Reward Function Design. We propose a composite reward function that assesses both the structure and correctness of predicted plans. The overall reward is defined as:

$$R(\text{response}, \text{answer}) = R_{\text{format}}(\text{response}) + R_{\text{accuracy}}(\text{response}, \text{answer}), \tag{6}$$

where R_{format} ensures structural validity and task compatibility, while R_{accuracy} evaluates step-wise alignment with expert behavior. Following prior reinforcement fine-tuning practices(27; 44), we set the maximum format reward to 0.5 and the accuracy reward to 1.0.

(1) Format Reward. To encourage valid and interpretable plans, we design a structured format reward inspired by Embench (55), which requires the model's output to include four key sections: `reasoning_and_reflection`, `visual_state_description`, `language_plan`, and `executable_plan`. The reward is composed of three components:

$$R_{\text{format}} = R_{\text{structure}} + R_{\text{valid}} + R_{\text{match}}, \tag{7}$$

Each component reflects a specific aspect of format quality and all three components are weighted proportionally according to a 2:1:1 ratio:

- $R_{\text{structure}}$ rewards the presence of all required top-level fields, ensuring structural completeness.

- R_{valid} measures the proportion of steps that include syntactically correct `action_id` and `action_name` pairs, reflecting output well-formedness.
- R_{match} evaluates the number of actions that align with a predefined schema, ensuring semantic correctness and avoiding hallucinated actions.

(2) Accuracy Reward with Multi-step Allocation. We compare the predicted sequence $\hat{a} = \{a_1, ..., a_k\}$ with the ground-truth expert actions $a^* = \{a_1^*, ..., a_k^*\}$ using prefix matching. To reflect long-horizon planning quality, we define a progressive reward allocation curve that assigns higher reward to longer correct prefixes. Let n be the number of consecutive matches such that $a_i = a_i^*$ for all $i \in [1, n]$. The reward is defined as:

$$R_{\text{accuracy}} = \frac{n(n+1)}{k(k+1)}, \tag{8}$$

where k is the length of the reference sequence. This formulation allocates progressively larger reward as more steps are correctly predicted in sequence, encouraging the model to maintain long-horizon consistency.

2.3 Training Pipeline and Details

We adopt a two-stage training paradigm to effectively equip vision-language models (VLMs) with long-horizon planning capabilities: supervised fine-tuning (SFT) for initialization, followed by reinforcement fine-tuning (RFT) for optimization as the main part.

2.3.1 Stage 1: Supervised Fine-tuning (SFT).

To bootstrap embodied reasoning and spatial grounding, we distill outputs from a proprietary model (Gemini-2.0-flash) on ALFRED-style tasks. Given a task goal g and visual history h_t, we construct a prompt $p = \texttt{Prompt}(g, h_t)$ and record Gemini's plan $\hat{a}_{t+1}$, forming a dataset $\mathcal{D}_{\text{SFT}} = \{(p_i, \hat{a}_i)\}_{i=1}^{N}$. The VLM policy π_θ is then trained via maximum likelihood:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(p,\hat{a})\sim\mathcal{D}_{\text{SFT}}} \left[\log \pi_\theta(\hat{a} \mid p)\right]. \tag{9}$$

This stage aligns the model with commonsense patterns and structural conventions seen in expert demonstrations, serving as a strong prior for downstream reinforcement learning.

2.3.2 Stage 2: Reinforcement Fine-tuning (RFT).

While SFT improves task-specific performance, it often lacks the reasoning generalization needed for unseen scenarios. To address this, we introduce reinforcement fine-tuning for interactive multi-turn embodied planning tasks. Building on the offline reward paradigm described in the previous section, we optimize the policy using GRPO (36). To ensure training stability and sample diversity, we further incorporate an online data filtering strategy that selectively retains informative prompt-response groups for gradient updates.

Optimization via GRPO. We employ Group Relative Policy Optimization (GRPO) (36), a stable and efficient method for reward-guided training. Given a prompt x, the model samples G candidate responses $\{y_1, ..., y_G\} \sim \pi_\theta(\cdot \mid x)$, each scored by reward $r_i = R(y_i)$. The relative advantage is computed as:

$$A_i = \frac{r_i - \text{mean}(\{r_j\})}{\text{std}(\{r_j\})}, \tag{10}$$

and the GRPO loss encourages high-reward responses while constraining deviation from a reference policy:

$$\mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\, \mathbb{E}_{\{y_i\}\sim\pi_\theta} \left[\frac{1}{G}\sum_{i=1}^{G}\left(\text{clip}\left(\frac{\pi_\theta(y_i \mid x)}{\pi_{\text{old}}(y_i \mid x)},\ 1-\epsilon,\ 1+\epsilon\right)\cdot A_i - \beta\cdot\mathcal{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})\right)\right]. \tag{11}$$

Model	EB-ALFRED (Seen)						EB-Habitat (Unseen)					
	Avg	Base	Com	Cplx	Visual	Spatial	Avg	Base	Com	Cplx	Visual	Spatial
Closed-Source MLLMs												
Claude-3.5-Sonnet	65.2	70	62	72	62	60	70.4	96	68	74	74	40
Gemini-2.0-flash	50.8	58	58	50	46	42	38.4	76	30	30	30	26
GPT-4o	54.8	62	52	68	44	48	53.6	82	34	62	58	32
GPT-4o-mini	26.4	32	24	32	20	24	36.8	68	38	28	28	22
Open-Source MLLMs												
LLaMA-3.2-90B	35.2	38	34	44	28	32	45.6	94	24	50	32	28
LLaMA-3.2-11B	15.2	24	8	16	22	6	26.8	62	16	24	14	18
Qwen2.5-VL-72B	40.8	50	42	42	36	34	41.2	72	28	42	40	24
Qwen2.5-VL-7B	2.0	4	2	2	2	0	14	38	4	12	4	12
InternVL2.5-78B	36.8	38	34	42	34	36	53.2	80	42	56	58	30
InternVL2.5-8B	3.6	2	0	12	0	4	19.6	48	6	16	10	18
Open-Source Reasoning MLLMs												
R1-VL-7B	2	2	2	6	0	0	8.4	24	0	4	6	8
MM-Eureka-Qwen-7B	3.2	6	4	4	2	0	19.2	40	16	14	10	16
Open-Source Embodied MLLMs												
RoboBrain	0.4	2	0	0	0	0	17.6	38	6	18	8	18
Tapa	0.0	0	0	0	0	0	0.0	0	0	0	0	0
Open-Source Embodied + Reasoning MLLMs												
Ours(base)	2.0	4	2	2	2	0	14	38	4	12	4	12
Ours(SFT)	22	34	22	24	12	18	13.6	34	2	10	10	12
Ours(SFT+RFT)	49.2	60	60	48	38	40	22.4	56	8	18	16	14

Table 1: Side-by-side comparison: left EB-ALFRED (Seen) vs. right EB-Habitat (Unseen). **Abbreviations:** Com = Common, Cplx = Complex

Online Data Filtering. To ensure informative and stable gradients, we incorporate an online filtering strategy during RFT, inspired by PRIME (10) and MM-Eureka (27). For each prompt group, we discard uninformative samples by measuring how many responses achieve full reward:

$$C_x = \left| \{ y^{(i)} \mid r^{(i)} = 1 \} \right|. \tag{12}$$

Only groups with C_x within a predefined range are retained to maintain a balanced contrastive learning signal. Accepted samples are buffered, and GRPO is performed periodically over the collected data. This stabilizes training by avoiding reward degeneracy and encouraging consistent policy improvement.

3 Experiments

We conduct a series of experiments to evaluate the effectiveness of our proposed reinforcement fine-tuning (RFT) framework for multi-step embodied planning. Specifically, we aim to answer the following key questions:

- **(Q1)** *How well does our method perform in interactive benchmarks for multi-step embodied task planning?* (Section 3.1)
- **(Q2)** *Is reinforcement fine-tuning necessary and uniquely beneficial, especially compared to supervised fine-tuning?* (Section 3.2)
- **(Q3)** *Does each component of the RFT framework contribute effectively to the final performance?* (Section 3.3)

3.1 Experiment Results in Embench

3.1.1 Experimental Settings

Benchmark Most prior works in embodied planning reduce evaluation to static visual question answering, which fails to capture the interactive and sequential nature of real-world decision-making.

To address this gap, we adopt **Embench**(55), a benchmark designed for evaluating multimodal agents in dynamic, interactive environments.

Embench provides a unified framework across four embodied settings and supports over 1,100 tasks involving manipulation, navigation, and spatial reasoning. We evaluate on two environments: **EB-ALFRED**, built on ALFRED(40) and AI2-THOR(22), and **EB-Habitat**, based on Habitat 2.0's rearrangement tasks(35). The benchmark organizes tasks into different subsets. Among them, the *Base* set forms the core task pool, while the *Common Sense*, *Complex Instruction*, *Spatial Awareness*, *Visual Appearance* are constructed via prompt-level augmentation that increases reasoning or perception difficulty, such as adding commonsense constraints or syntactic complexity. Notably, our RL fine-tuning is conducted solely on the *Base* set without any prompt augmentation, demonstrating its ability to generalize beyond the training distribution.

All models generate step-by-step plans from egocentric inputs and execute them in simulation. Since our training data is collected from the ALFRED , EB-Habitat serves as an fully **out-of-domain** setting for generalization evaluation. More details are provided in Appendix.

Baselines We compare our method against a range of baselines, including: (1) proprietary models such as Claude-3.5-Sonnet(3), Gemini-2.0-flash(4), GPT-4o(2), and GPT-4o-mini(1); (2) open-source general VLMs like LLaMA-3.2-Vision-11B(5), Qwen2.5-VL-7B(7), and InternVL2.5-8B(9); (3) reasoning-oriented models such as MM-Eureka(27) and R1-VL(60); and (4) embodied VLMs including RoboBrain(19) and TAPA(52). For evaluation, we convert visual inputs into text for TAPA due to its lack of vision capabilities. Further details on each baseline are provided in Appendix.

Evaluation Metrics We follow the original Embench (55) to use **task success rate** as the primary evaluation metric. A task is marked as successful only if all predefined goal-checking conditions are satisfied at the end of execution.

To support multi-turn planning, Embench adopts an iterative evaluation protocol where the model generates a new action sequence based on the latest observation at each round. The environment executes the actions and returns updated states until task success or step limit is reached.

3.1.2 Main results

In-Domain Results We conduct comprehensive in-domain evaluations on the EB-ALFRED environment. As shown in Table 1, our proposed model achieves a task success rate of 43.6%, significantly outperforming GPT-4o-mini (22.0%) and much larger models such as Qwen2.5-VL-72B (33.7%) and LLaMA3.2-90B-Vision-Ins(32.0%).

Several key observations emerge from the results: (1) Our two-stage training pipeline (SFT + RFT) leads to consistent performance gains in embodied task planning for both base and other advanced tasks. (2) Existing open-source reasoning models and embodied VLMs perform poorly in Embench. While reasoning models produce verbose intermediate steps, they struggle to execute correct action sequences. Similarly, embodied VLMs lack the generalization ability to transfer to Embench tasks.

Variant	EB-ALFRED (Seen)		EB-Habitat (Unseen)	
	Avg	Base	Avg	Base
Base	2	4	14	38
SFT only	22	34	13.6	30
RFT only	10.4	18	17.6	40
RFT → SFT	33.2	40	11.4	30
SFT → SFT	37.6	50	11.6	22
SFT → RFT (ours)	**49.2**	**60**	**22.4**	**56**

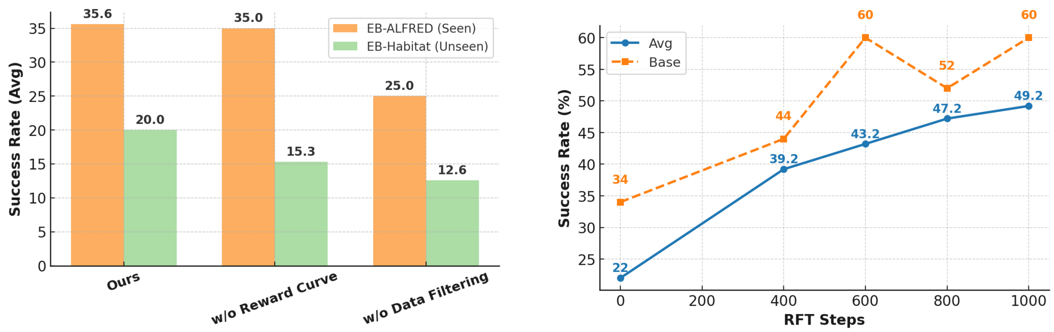
(a) Ablation study on training stages in EB-ALFRED and EB-Habitat.

Model	Overall Acc	SpatialMap	MazeNav	SpatialGrid
Base	0.475	0.696	0.256	0.542
SFT only	0.488	0.682	**0.328**	0.524
SFT+RFT	**0.503**	**0.748**	0.260	**0.605**

(b) Visual reasoning accuracy on spatial VQA subsets.

Table 2: RFT Generalization Experiment

Out-of-Domain Results To evaluate generalization, we tested our models in the EB-Habitat environment, which differs from ALFRED in terms of scenes, objects, action space, and task types. As shown in the right part of Table 1, our method exhibits strong out-of-domain performance, outperforming all baseline models of similar 7B size, including general-purpose, reasoning-augmented, and embodied VLMs.The result highlight Reinforcement fine-tuning leads to substantial improvements even in completely unseen environments.



(a) Ablation on reward curve and data filtering modules.

(b) Performance evolution with increasing RFT steps.

Figure 3: Ablation study on the RFT module. (a) Module-level analysis shows that removing either the reward curve or data filtering leads to significant degradation. (b) Performance consistently improves with more RFT steps, with Base tasks saturating earlier while generalization emerges in other subsets.

3.2 RFT Generalizes While SFT Overfits

Is Reinforcement Fine-Tuning Necessary? A key question is whether the performance gain of GRPO-based reinforcement fine-tuning (RFT) stems from the optimization process itself, or merely from exposure to additional trajectory data. To investigate this, we compare five training strategies: (1) **Base**: the original Qwen2.5-VL-7B model without any tuning; (2) **SFT only**: supervised fine-tuning (SFT) on distilled trajectories; (3) **RFT→SFT**: first applying RFT, then re-align with SFT; (4) **SFT→SFT**: conducting SFT , followed by additional SFT using the same trajectories during RFT. This variant isolates the effect of data exposure from optimization. and (5) **SFT→RFT (ours)**: our proposed pipeline with SFT followed by GRPO-based RFT.

As shown in Table 2a, our **SFT → RFT** pipeline achieves the best performance across both seen and unseen environments. While **SFT → SFT** brings moderate gains over **SFT only** on seen tasks, it surprisingly degrades performance in unseen domains—exposing the limitations of supervised fine-tuning. In contrast, our **SFT → RFT** approach not only boosts more in-domain accuracy but also enhances generalization, confirming the necessity of offline reward-driven optimization beyond simple trajectory exposure.

Does RFT Overfit to Embodied Benchmarks? To further evaluate the generalization capability of RFT, we assess whether fine-tuning on Embench harms the model's performance on its original training domains besides embodied task planning. Specifically, we evaluate on **SpatialEval** (46), a benchmark designed to assess general spatial understanding across three diverse tasks: spatial maps, maze navigation, and spatial grids.

As shown in Table 2b, the SFT-RFT model not only avoids degradation on general spatial reasoning tasks but also improves performance on spatial map and spatial grid tasks. This indicates that our reinforcement-based fine-tuning pipeline promotes structured reasoning without overfitting to the embodied benchmark. The structured action plans and reward-aligned outputs learned through RFT appear to benefit broader visuospatial understanding.

3.3 Ablation Study on RFT Module

Beyond the primary comparison between supervised fine-tuning (SFT) and reinforcement fine-tuning (RFT), we further conduct ablation studies to dissect the internal design choices of our RFT stage. Specifically, we investigate two key modules: (1) the *reward allocation curve*, and (2) the *data filtering mechanism*. These components are designed to enhance the learning signal and stabilize policy optimization.

The reward allocation curve applies a non-linear weighting over step-wise rewards, emphasizing later steps within a trajectory. This encourages the model to optimize for long-horizon strategies and complete task execution rather than short-sighted local successes. Meanwhile, the data filtering mechanism discards trivial or infeasible trajectories based on reward thresholds, reducing the variance of training samples and preventing overfitting to noisy or uninformative cases.

Figure 3a reports the results under the EB-ALFRED (Seen) and EB-Habitat (Unseen) settings. Both ablated variants exhibit clear performance drops: removing the reward curve decreases success rates on EB-Habitat from 20.0 to 15.3, while removing data filtering reduces EB-ALFRED from 35.6 to 25.0. This confirms that both modules play complementary roles in improving generalization and robustness. We note that these comparisons are conducted at 600 RFT steps, rather than the full 1000 steps for time limit.

In addition to module-level ablations, we also analyze how performance evolves with different numbers of RFT steps. As shown in Figure 3b, the *Avg* success rate exhibits a consistent upward trend throughout training, rising from 22.0 at initialization to 49.2 after 1000 steps, which indicates that reinforcement fine-tuning steadily enhances overall policy competence. For the *Base* subset, performance quickly saturates around 600 steps, however, with further training, we observe continued improvements in other subsets such as *Visual*, *Spatial*, and *Complex* tasks, which drive the overall increase in the *Avg* score. This emergent transfer is particularly noteworthy given that our RFT training data exclusively contains *Base*-type tasks, highlighting the generalization capability induced by our reinforcement optimization process.

4 Limitation and Future Work

While our work demonstrates the effectiveness of GRPO-based reinforcement fine-tuning using offline rule-based rewards, further investigation is required to understand its theoretical foundations and explore potential improvements. Moreover, despite the practical advantages of avoiding expensive online rollouts, purely offline reward signals may limit the model's capacity to explore novel behaviors beyond the expert data distribution. Bridging this gap—by integrating the stability of offline supervision with the exploration capabilities of online interaction—represents a promising direction for future research.

In addition, our current focus lies on high-level embodied planning, producing structured action sequences that can guide downstream control modules. Although our method demonstrates strong performance and generalization in simulated benchmarks, it has not yet been deployed on real-world robotic platforms. Extending this framework to physical agents and integrating it with low-level control systems is an important step toward realizing embodied intelligence in practical applications.

5 Conclusion

In this paper, we tackle the challenge of enabling vision-language models to perform robust multi-step planning in dynamic embodied environments. To this end, we propose a reinforcement fine-tuning framework driven by structured offline rewards, which enhances reasoning and decision-making under long-horizon, interactive settings. Our method leverages rule-based feedback to guide Generalized Reinforced Preference Optimization (GRPO), enabling the agent to learn directly from expert trajectory without relying on costly online interaction or human preferences.

We validate our method on Embench, a comprehensive benchmark for interactive embodied planning, demonstrating that our model significantly outperforms both proprietary and open-source baselines of comparable or larger scale. Beyond in-domain performance, our approach shows strong generalization to out-of-distribution tasks and unseen environments — a benefit not observed in supervised fine-tuning alone. These results highlight the promise of reinforcement-driven reasoning as a scalable and effective direction for advancing embodied intelligence.

References

[1] GPT-4o mini: advancing cost-efficient intelligence. URL: `https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/`.

[2] Hello GPT-4o | OpenAI. URL: `https://openai.com/index/hello-gpt-4o/`.

[3] Introducing claude 3.5 sonnet \ anthropic. URL: `https://www.anthropic.com/news/claude-3-5-sonnet`.

[4] Introducing gemini 2.0: our new AI model for the agentic era. URL: `https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/`.

[5] Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. URL: `https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/`.

[6] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances. URL: `http://arxiv.org/abs/2204.01691`, `arXiv:2204.01691[cs]`, `doi:10.48550/arXiv.2204.01691`.

[7] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[8] Yaran Chen, Wenbo Cui, Yuanwen Chen, Mining Tan, Xinyao Zhang, Dongbin Zhao, and He Wang. RoboGPT: an intelligent agent of making embodied long-term decisions for daily instruction tasks. URL: `http://arxiv.org/abs/2311.15649`, `arXiv:2311.15649[cs]`, `doi:10.48550/arXiv.2311.15649`.

[9] 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.

[10] Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

[11] Zhirui Deng, Zhicheng Dou, Yutao Zhu, Ji-Rong Wen, Ruibin Xiong, Mang Wang, and Weipeng Chen. From novice to expert: Llm agent policy optimization via step-wise reinforcement learning. *arXiv preprint arXiv:2411.03817*, 2024.

[12] Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. A survey of embodied ai: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(2):230–244, 2022.

[13] Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL: `https://github.com/huggingface/open-r1`.

[14] Xian Fu, Min Zhang, Peilong Han, Hao Zhang, Lei Shi, Hongyao Tang, et al. What can vlms do for zero-shot embodied task planning? In *ICML 2024 Workshop on LLMs and Cognition*, 2024.

[15] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[16] Jian Hu, Xibin Wu, Zilin Zhu, Weixun Wang, Dehao Zhang, Yu Cao, et al. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

[17] Yingdong Hu, Fanqi Lin, Tong Zhang, Li Yi, and Yang Gao. Look before you leap: Unveiling the power of GPT-4v in robotic vision-language planning. URL: `http://arxiv.org/abs/2311.17842`, `arXiv:2311.17842[cs]`, `doi:10.48550/arXiv.2311.17842`.

[18] Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025.

[19] Yuheng Ji, Huajie Tan, Jiayu Shi, Xiaoshuai Hao, Yuan Zhang, Hengyuan Zhang, Pengwei Wang, Mengdi Zhao, Yao Mu, Pengju An, Xinda Xue, Qinghang Su, Huaihai Lyu, Xiaolong Zheng, Jiaming Liu, Zhongyuan Wang, and Shanghang Zhang. RoboBrain: A unified brain model for robotic manipulation from abstract to concrete. URL: `http://arxiv.org/abs/2502.21257`, `arXiv:2502.21257[cs]`, `doi:10.48550/arXiv.2502.21257`.

[20] Byeonghwi Kim, Jinyeon Kim, Yuyeong Kim, Cheolhong Min, and Jonghyun Choi. Context-aware planning and environment-aware memory for instruction following embodied agents. URL: `http://arxiv.org/abs/2308.07241`, `arXiv:2308.07241[cs]`, `doi:10.48550/arXiv.2308.07241`.

[21] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, et al. Openvla: An open-source vision-language-action model. In *8th Annual Conference on Robot Learning*.

[22] 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.

[23] Xinhao Li, Ziang Yan, Desen Meng, Lu Dong, Xiangyu Zeng, Yinan He, Yali Wang, Yu Qiao, Yi Wang, and Limin Wang. Videochat-r1: Enhancing spatio-temporal perception via reinforcement fine-tuning. *arXiv preprint arXiv:2504.06958*, 2025.

[24] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

[25] Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual-rft: Visual reinforcement fine-tuning. *arXiv preprint arXiv:2503.01785*, 2025.

[26] Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied ai. *arXiv preprint arXiv:2405.14093*, 2024.

[27] Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, et al. Mm-eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2503.07365*, 2025.

[28] Chancharik Mitra, Brandon Huang, Trevor Darrell, and Roei Herzig. Compositional chain-of-thought prompting for large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14420–14431, 2024.

[29] Debjyoti Mondal, Suraj Modi, Subhadarshi Panda, Rituraj Singh, and Godawari Sudhakar Rao. Kam-cot: Knowledge augmented multimodal chain-of-thoughts reasoning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 18798–18806, 2024.

[30] Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhai Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng Dai, Yu Qiao, and Ping Luo. EmbodiedGPT: Vision-language pre-training via embodied chain of thought. URL: `http://arxiv.org/abs/2305.15021`, `arXiv:2305.15021[cs]`, `doi:10.48550/arXiv.2305.15021`.

[31] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[32] Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. Reasoning with large language models, a survey. *arXiv preprint arXiv:2407.11511*, 2024.

[33] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

[34] Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, and Ian Reid. SayPlan: Grounding large language models using 3d scene graphs for scalable robot task planning.

[35] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347, 2019.

[36] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[37] Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, et al. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615*, 2025.

[38] Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

[39] Suyeon Shin, Sujin jeon, Junghyun Kim, Gi-Cheon Kang, and Byoung-Tak Zhang. Socratic planner: Inquiry-based zero-shot planning for embodied instruction following. URL: `http://arxiv.org/abs/2404.15190`, `arXiv:2404.15190[cs]`.

[40] 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.

[41] Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B. Tenenbaum, Leslie Kaelbling, and Michael Katz. Generalized planning in PDDL domains with pretrained large language models. 38(18):20256–20264. URL: `https://ojs.aaai.org/index.php/AAAI/article/view/30006`, `doi:10.1609/aaai.v38i18.30006`.

[42] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: Generating situated robot task plans using large language models. URL: `http://arxiv.org/abs/2209.11302`, `arXiv:2209.11302[cs]`, `doi:10.48550/arXiv.2209.11302`.

[43] Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization for llm agents. *arXiv preprint arXiv:2403.02502*, 2024.

[44] Huajie Tan, Yuheng Ji, Xiaoshuai Hao, Minglan Lin, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Reason-rft: Reinforcement fine-tuning for visual reasoning. *arXiv preprint arXiv:2503.20752*, 2025.

[45] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

[46] Jiayu Wang, Yifei Ming, Zhenmei Shi, Vibhav Vineet, Xin Wang, Sharon Li, and Neel Joshi. Is a picture worth a thousand words? delving into spatial reasoning for vision language models. *Advances in Neural Information Processing Systems*, 37:75392–75421, 2024.

[47] Siyin Wang, Zhaoye Fei, Qinyuan Cheng, Shiduo Zhang, Panpan Cai, Jinlan Fu, and Xipeng Qiu. World modeling makes a better planner: Dual preference optimization for embodied task planning. URL: `http://arxiv.org/abs/2503.10480`, `arXiv:2503.10480[cs]`, `doi:10.48550/arXiv.2503.10480`.

[48] Yaoting Wang, Shengqiong Wu, Yuecheng Zhang, Shuicheng Yan, Ziwei Liu, Jiebo Luo, and Hao Fei. Multimodal chain-of-thought reasoning: A comprehensive survey. *arXiv preprint arXiv:2503.12605*, 2025.

[49] Yiqi Wang, Wentao Chen, Xiaotian Han, Xudong Lin, Haiteng Zhao, Yongfei Liu, Bohan Zhai, Jianbo Yuan, Quanzeng You, and Hongxia Yang. Exploring the reasoning abilities of multimodal large language models (mllms): A comprehensive survey on emerging trends in multimodal reasoning. *arXiv preprint arXiv:2401.06805*, 2024.

[50] Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

[51] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[52] Zhenyu Wu, Ziwei Wang, Xiuwei Xu, Jiwen Lu, and Haibin Yan. Embodied task planning with large language models. *arXiv preprint arXiv:2307.01848*, 2023.

[53] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025.

[54] Zhiyuan Xu, Kun Wu, Junjie Wen, Jinming Li, Ning Liu, Zhengping Che, and Jian Tang. A survey on robotics with foundation models: toward embodied ai. *arXiv preprint arXiv:2402.02385*, 2024.

[55] Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, et al. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *arXiv preprint arXiv:2502.09560*, 2025.

[56] Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.

[57] Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *8th Annual Conference on Robot Learning*, 2024.

[58] Jianke Zhang, Yanjiang Guo, Xiaoyu Chen, Yen-Jen Wang, Yucheng Hu, Chengming Shi, and Jianyu Chen. Hirt: Enhancing robotic control with hierarchical robot transformers. *arXiv preprint arXiv:2410.05273*, 2024.

[59] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[60] Jingyi Zhang, Jiaxing Huang, Huanjin Yao, Shunyu Liu, Xikun Zhang, Shijian Lu, and Dacheng Tao. R1-vl: Learning to reason with multimodal large language models via step-wise group relative policy optimization. *arXiv preprint arXiv:2503.12937*, 2025.

[61] Wenqi Zhang, Mengna Wang, Gangao Liu, Xu Huixin, Yiwei Jiang, Yongliang Shen, Guiyang Hou, Zhe Zheng, Hang Zhang, Xin Li, et al. Embodied-reasoner: Synergizing visual search, reasoning, and action for embodied interactive tasks. *arXiv preprint arXiv:2503.21696*, 2025.

[62] Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*, 2023.

[63] Baining Zhao, Ziyou Wang, Jianjie Fang, Chen Gao, Fanhang Man, Jinqiang Cui, Xin Wang, Xinlei Chen, Yong Li, and Wenwu Zhu. Embodied-r: Collaborative framework for activating embodied spatial reasoning in foundation models via reinforcement learning. *arXiv preprint arXiv:2504.12680*, 2025.

[64] 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.

A Appendix Contents

- Section B: Related Work
- Section C: Additional Details of SFT Training Stage
- Section D: Additional Details of RFT Training Stage
- Section E: Additional Details for Evaluation
- Section F: Case study and Visualization

B Related Work

B.1 Embodied Task Planning

Embodied task planning focuses on decomposing high-level natural language instructions into executable sequences of sub-tasks, enabling agents to perform complex behaviors in interactive environments. With the emergence of large language and vision-language models(53; 54), researchers have explored using pretrained LLMs or VLMs to generate plans from textual and visual observations, typically relying on carefully crafted prompts(39; 34; 17; 20; 42; 14) or auxiliary tools(34; 6; 41) to provide necessary planning cues. While simple and data-efficient, such methods often struggle with spatial grounding and temporal coherence in visually rich environments. Advanced methods have tried to fine-tune LLMs or VLMs to improve planning performance. Several works have employed supervised fine-tuning pipelines(52; 8; 19), while others adopt preference optimization methods(47; 43) such as Direct Preference Optimization (DPO)(33) to better align model behavior with expert planning preferences.

Despite these advances, most existing methods operate in static or offline settings, where plans are generated without actual interaction with the environment. In this work, we address this limitation by evaluating our model in interactive environments(55) bridging the gap between static planning capabilities and dynamic embodied execution.

B.2 Vision-Language Model Reasoning

Reasoning in vision-language models (VLMs) involves drawing inferences from both textual and visual inputs, often requiring spatial, temporal, or causal understanding(48; 49). A common approach is Chain-of-Thought (CoT) prompting(51), where the model generates intermediate steps to clarify its reasoning. In multimodal settings, Multimodal Chain-of-Thought (MCoT) extends this idea by integrating visual inputs like images and videos into the reasoning process(62; 29; 28).

More recently, R1-style reinforcement learning(15; 36) has emerged as an effective framework for enhancing reasoning capabilities. These methods optimize reasoning quality through reward-guided learning, enabling models to self-correct and generate more detailed reasoning processes. Originally developed for text-based reasoning, R1 approaches have since been extended to multimodal domains, including image-based QA(25; 37; 60), visual math problems(27; 18; 45), and video reasoning(23). In the context of embodied AI, some early studies(63; 61; 44) have applied R1-based training to question answering tasks, however, they primarily focus on short-horizon QA tasks. In contrast, our work is the first to adopt R1-style reinforcement fine-tuning for long-horizon embodied planning, aiming to improve structured decision-making across multiple interactive steps.

C Additional Details of SFT Training Stage

C.1 Training Dataset Details

To initialize the model with structured planning priors, we construct a supervised fine-tuning (SFT) dataset via response distillation from a high-capacity proprietary model. Specifically, we query *Gemini-2.0-flash* on a subset of EB-ALFRED tasks from the Embench benchmark and collect its response traces. For each task prompt, we record the full model output, including both intermediate reasoning and executable plan.

To ensure diversity and robustness, we repeat this process across three separate sampling rounds. The final dataset comprises a total of 4,245 training samples. Each sample includes a natural language instruction, egocentric image observations, and the model's full response in a structured format. An example training instance is shown in figure C.1

SFT dataset example


```
{
 "messages": [
  {
   "role": "user",
   "content": {EB-ALFRED prompt}
  },
  {
   "role": "assistant",
   "content": "{"reasoning_and_reflection": "The user wants me to rinse
   off a ladle and move it to the table. First I need to find the ladle,
    pick it up, find the sink, put the ladle down in the sink, find the
   faucet, turn on the faucet to rinse the ladle, turn off the faucet,
   pick up the ladle, find the table, and put the ladle on the table.",
    "visual_state_description": "I can see a garbage can and a fridge in
    the current view.",  "language_plan": "First, locate a ladle and
   pick it up. Then, find a sink and put the ladle down in the sink.
   Next, find a faucet, turn it on to rinse the ladle, and then turn it
   off. After that, pick up the rinsed ladle, find a dining table, and
   place the ladle on the table.",  "executable_plan": [   {    "action_
   id": 64,    "action_name": "find a Ladle"   },   {    "action_id":
   109,    "action_name": "pick up the Ladle"   },   {    "action_id":
   79,    "action_name": "find a Sink"   },   {    "action_id": 133,
   "action_name": "put down the object in hand"   },   {    "action_id":
    2,    "action_name": "find a Faucet"   },   {    "action_id": 155,
     "action_name": "turn on the Faucet"   },   {    "action_id": 156,
     "action_name": "turn off the Faucet"   },   {    "action_id": 64,
     "action_name": "find a Ladle"   },   {    "action_id": 109,    "
   action_name": "pick up the Ladle"   },   {    "action_id": 18,    "
   action_name": "find a DiningTable"   },   {    "action_id": 133,    "
   action_name": "put down the object in hand"   }]  }"
   }
 ],
 "images": [
  "example.png"
 ]
},
```


RFT dataset example


```
{
 {
   "id": "trial_T20190909_062150_965386_remain_0",
   "question": "{Our_RFT_prompt}",
   "answer": "['Goto handtowelholder', 'Pickup handtowel', 'Goto
   garbagecan', 'Put handtowel']",
   "message": "[{\"role\": \"system\", \"content\": \"Solve the question.
    The user asks a question, and you solves it. You first thinks about
   the reasoning process in the mind and then provides the user with the
    answer.\"}, {\"role\": \"user\", \"content\": [{\"type\": \"image\",
    \"image\": \"example.jpg\"}, {\"type\": \"text\", \"text\": \"{Our_
   RFT_prompt}\"}]}]"
 }
},
```

C.2 Training Hyperparameters

We perform full-parameter supervised fine-tuning on the Qwen2.5-VL-7B model using the `LLaMA-Factory`(64) framework. The training is conducted on 4 NVIDIA A100 40GB GPUs for approximately 8 hours. All hyperparameters are summarized in Table 3.

Component	Setting	Component	Setting
	Model Configuration		
image_max_pixels	262144	freeze_vision_tower	true
freeze_language_model	false	freeze_multi_modal_projector	true
deepspeed config	ds_z3_config.json		
	Dataset Configuration		
dataset	alfred_sft	template	qwen2_vl
cutoff_len	2048	max_samples	1000
overwrite_cache	true	preprocessing_workers	16
dataloader_workers	4		
	Training Configuration		
stage	sft	finetuning_type	full
do_train	true	num_train_epochs	3.0
learning_rate	1e-5	per_device_batch_size	1
grad_accum_steps	2	lr_scheduler	cosine
warmup_ratio	0.1	bf16	true
ddp_timeout	180000000		

Table 3: Detailed hyperparameters used in supervised fine-tuning.

Figure 4: Summary of SFT training results.

Metric	Value
Epochs	3.0
Total FLOPs	3.13e13
Training Loss	0.252
Runtime (s)	21111.79
Samples/sec	0.142
Steps/sec	0.018

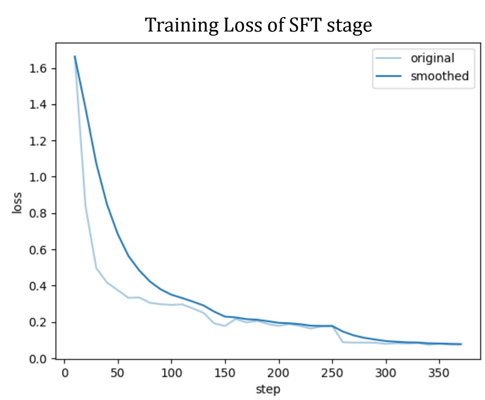


Figure 5: Training loss curve during SFT stage.

C.3 Training Results

We record the final metrics and loss curve from the supervised fine-tuning process, as shown in Figure 5. The table summarizes key training statistics after 3 epochs of full-parameter tuning.

D Additional Details of RFT training stage

D.1 Training Dataset Details

We construct our reinforcement fine-tuning (RFT) dataset based on the ALFRED benchmark, following the decomposition and formatting strategy described in Section 2. Notably, we do not reuse the SFT-distilled dataset for reinforcement fine-tuning. This decision is motivated by two key considerations: (1) the distilled data may contain suboptimal trajectories, introducing noise into the learning signal; (2) the distilled instruction format is tightly coupled with the benchmark evaluation prompts, whereas our constructed dataset introduces instruction variations that encourage greater policy generalization and better isolate the impact of reinforcement learning.

The resulting dataset contains 43,898 samples, each formatted to include a natural language instruction, a visual observation, and a ground-truth action sequence used for reward computation. We provide a full example of a training sample from the RFT dataset for reference in figureB.2

D.2 Training Hyperparameters

We implement reinforcement fine-tuning using the `OpenRLHF`(16) framework, adopting the Generalized Reinforced Preference Optimization (GRPO) algorithm(36) to optimize policy learning from structured reward feedback. A full list of training hyperparameters is provided in Table 4.

Hyperparameter	**Value**	**Hyperparameter**	**Value**
ref_num_nodes	1	vllm_num_engines	8
ref_num_gpus_per_node	8	actor_num_gpus_per_node	8
actor_num_nodes	1	vllm_tensor_parallel_size	1
vllm_gpu_memory_utilization	0.65	vllm_enable_sleep	True
vllm_sync_backend	nccl	temperature	1.0
max_epochs	1	max_episodes	10
prompt_max_len	3000	max_samples_len	10000
generate_max_len	4096	advantage_estimator	group_norm
zero_stage	3	actor_learning_rate	1e-6
init_kl_coef	0.0	n_samples_per_prompt	8
micro_train_batch_size	1	micro_rollout_batch_size	2
train_batch_size	128	rollout_batch_size	128
freeze_prefix	visual	enable_accuracy_filter	True
accuracy_lower_bound	0.1	accuracy_upper_bound	0.9

Table 4: Hyperparameter configuration used during reinforcement fine-tuning.

D.3 Training Log and Result

We record the reinforcement fine-tuning process using several key indicators, as visualized in Figure 6.

The *total reward* refers to the combined score of the format reward and the accuracy reward. Due to the use of an online filtering strategy during training, we distinguish between two types of accuracy reward: *accuracy reward (filtered)*, which reflects the reward from selected high-quality samples that pass the filtering criteria, and *accuracy reward (original)*, which represents the average reward across all generated responses prior to filtering.

We also report two types of length statistics: *response length*, which quantifies the number of tokens generated by the model for each output, and *total length*, which denotes the combined token length of the input prompt and generated response.

E Additional Details for Evaluation

E.1 Detailed Introduction to EmbodiedBench

EmbodiedBench is a comprehensive interactive benchmark designed to evaluate vision-language agents in embodied planning scenarios. Unlike static visual question answering settings, EmbodiedBench offers dynamic, simulation-based environments where agents must generate and execute multi-step plans grounded in first-person visual observations and natural language instructions. The benchmark spans four embodied environments and supports over 1,100 diverse tasks with hierarchical action levels, covering both high-level planning and low-level control.

In our work, we focus on two high-level planning environments within EmbodiedBench:

EB-ALFRED. EB-ALFRED is built upon the ALFRED dataset (40) and implemented on top of the AI2-THOR simulator (22). It supports eight core skill types such as *pick up*, *put down*, *find*,

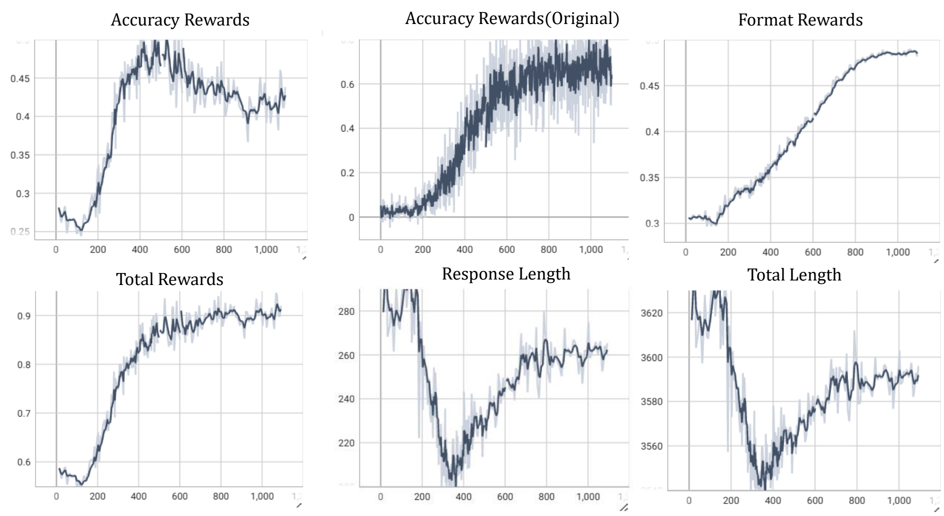


Figure 6: Training curve during reinforcement fine-tuning. The figure shows the progression of total reward, filtered and unfiltered accuracy reward, and generation length statistics.

open/close, and *turn on/off*. The environment provides egocentric visual inputs and textual feedback (e.g., success/failure messages), enabling agents to adaptively plan and act. Compared to the original ALFRED setup, EB-ALFRED enhances object diversity and simulator robustness. Specifically, it supports multiple object instances of the same type, merges redundant actions (e.g., unified *put down*), and dynamically adjusts the action space size (ranging from 171 to 298). These improvements provide a more realistic and flexible environment for assessing embodied planning capabilities.

EB-Habitat. EB-Habitat extends the Language Rearrangement benchmark (35), based on the Habitat 2.0 simulator. It focuses on five high-level skills: *navigation*, *pick*, *place*, *open*, and *close*. Unlike ALFRED, navigation in EB-Habitat is constrained to receptacle-type targets, requiring more sophisticated exploration and scene understanding. The environment includes 282 instruction templates and places more emphasis on spatial reasoning and location-aware planning, making it a complementary testbed for generalization.

Task Subsets. To enable fine-grained capability analysis, Embench introduces six distinct task subsets. Due to space limitations, we omit the subset "Long Horizon" from the main table and report its results in the Appendix.

- **Base:** Evaluates standard task-solving skills under low to medium complexity, testing general planning competence.
- **Common Sense:** Assesses agents' ability to reason over implicit object references and everyday knowledge.
- **Complex Instruction:** Presents long, noisy or ambiguous contexts to evaluate the agent's ability to extract user intent.
- **Spatial Awareness:** Requires understanding object relationships in space, such as relative positions or arrangements.
- **Visual Appearance:** Involves identifying objects via attributes like color or shape, testing fine-grained visual recognition.
- **Long Horizon:** Contains tasks demanding long sequences of actions (often exceeding 15 steps), stressing planning depth and temporal consistency.

Each subset is designed to probe a specific capability of embodied reasoning, such as commonsense inference, spatial understanding, or long-horizon planning. In our experiments, we evaluate model performance across all six subsets to provide a fine-grained analysis. As shown in Table 5, these categories span a wide range of reasoning challenges. Notably, since our reinforcement fine-tuning

Table 5: Examples of each task type from EB-ALFRED and EB-Habitat.

Task Subset	ALFRED Example	Habitat Example
Base	Put washed lettuce in the refrigerator.	Move one of the pear items to the indicated sofa.
Common Sense	Place washed leafy green vegetable in a receptacle that can keep it fresh.	Prepare for a game by delivering something to play with to the TV stand.
Complex Instruction	Place the washed lettuce in the refrigerator. This way, it's ready for any delightful recipe ideas you have.	When you find the fridge door open, go ahead and move one bowl to the sofa; otherwise, transport one hammer to the sofa.
Spatial Awareness	Put two spray bottles in the cabinet under the sink against the wall.	Move a spatula from the right counter to the right receptacle of the left counter.
Visual Appearance	Put a knife in a blue container onto the black table in the corner.	Deliver a small red object with green top to the indicated large gray piece of furniture.
Long Horizon	Pick up knife, slice apple, put knife in bowl, heat apple slice in microwave, put apple slice on table.	Move the rubrics cube to the left counter; the towel to the left counter, and the bowl to the brown table.

dataset only includes *Base* tasks, we observe a significantly larger performance gain in this category, whereas improvements in other subsets are relatively modest. This highlights the need for more diverse training data to support generalizable planning across varied task types.

Overall, Embench provides a rigorous, scalable, and diagnostic framework for benchmarking embodied agents across diverse real-world challenges. In our setup, we use EB-ALFRED for in-domain training and evaluation, while EB-Habitat serves as an out-of-domain testbed to examine generalization performance.

E.2 Detailed Introduction to Baselines

To comprehensively evaluate our proposed method, we compare it against a diverse set of baselines, covering both proprietary and open-source models, as well as models specifically optimized for multimodal reasoning and embodied planning.

(1) *Closed-source models*: we include several leading proprietary vision-language models as strong general-purpose baselines, including Claude-3.5-Sonnet(3), Gemini-2.0-flash(4), GPT-4o(2), and GPT-4o-mini(1).

(2) *Open-source general VLMs*: we evaluate widely adopted open-source VLMs trained for generic multimodal tasks, such as LLaMA-3.2-Vision-11B(5), Qwen2.5-VL-7B(7) and InternVL2.5-8B(9).

(3) *Open-source reasoning VLMs*: we further include two representative models that have been explicitly optimized for multimodal reasoning, including MM-Eureka(27) and R1-VL(60).

MM-Eureka extends rule-based reinforcement learning to multimodal reasoning, enabling models to improve through reward-driven optimization without supervised fine-tuning. It reproduces key behaviors from language-only RL systems, such as reflection and reward-aligned response growth, achieving strong data efficiency and reasoning performance.

R1-VL enhances step-by-step reasoning in multimodal LLMs via StepGRPO, a reinforcement learning framework with dense, rule-based rewards for accuracy and logical consistency. It surpasses imitation learning by guiding models to self-correct flawed reasoning, achieving superior results on multiple benchmarks.

We also attempted to evaluate other open-source reasoning models, such as VisualRFT(25) and Open-R1(13). However, their inference speed was prohibitively slow, resulting in impractically long evaluation time on interactive benchmarks. Additionally, their final planning performance remained poor for embodied planning scenarios.

(4) *Embodied VLMs*: we also include RoboBrain(19) and TAPA(52), two representative open-source large models designed for embodied tasks.

Model	EB-ALFRED (Seen)													
	Avg		Base		Common		Complex		Visual		Spatial		Long	
	PR	ES	PR	ES	PR	ES	PR	ES	PR	ES	PR	ES	PR	ES
Closed-Source MLLMs														
Claude-3.5-Sonnet	70.11	14.9	72.67	12.2	65.83	12.74	73.33	11.48	65.5	14.02	68.83	16.96	74.5	21.98
Gemini-2.0-flash	57.13	16.5	61.83	13.96	60.67	14.0	55.33	15.16	55.33	15.26	46.67	17.04	63.0	23.56
GPT-4o	61.78	16.77	65.67	12.54	57.17	16.1	74.67	13.92	58.33	15.2	52.33	17.58	62.5	25.43
GPT-4o-mini	30.42	19.69	36.33	17.32	29.83	18.06	38.0	17.74	27.33	18.48	31.0	19.9	20.0	26.62
Open-Source General MLLMs														
Qwen2.5-VL-7B	6.86	9.4	5.67	8.78	4.0	4.2	5.0	5.28	5.33	7.16	0.67	8.26	20.5	22.72
InternVL2.5-8B	5.78	7.87	6.17	8.2	0.67	4.9	16.0	8.92	4.0	6.78	6.33	7.52	1.5	10.92
Open-Source Reasoning MLLMs														
R1-VL-7B	2.78	4.01	3.0	3.22	3.0	2.06	6.0	1.7	0.67	1.62	0.0	2.66	4.0	12.78
MM-Eureka-Qwen-7B	6.59	8.48	8.67	7.64	5.33	5.04	8.67	9.72	3.67	6.46	0.67	6.58	12.5	15.42
Open-Source Embodied MLLMs														
RoboBrain	1.22	6.7	3.33	6.1	0.67	6.3	0.67	3.68	0.67	7.56	0	6.36	2.0	10.22
Tapa	0	0.03	0	0.06	0	0	0	0	0	0.04	0	0.08	0	0
Open-Source Embodied + Reasoning MLLMs														
Ours (Base)	6.86	9.4	5.67	8.78	4.0	4.2	5.0	5.28	5.33	7.16	0.67	8.26	20.5	22.72
Ours (SFT only)	23.8	15.06	39	13.14	26.6	13.04	27.6	12.56	19.3	14.12	14.3	15.16	16.5	22.38
Ours (SFT+RFT)	53.03	17.63	70.3	13.72	65	15.3	59.9	16.12	48.5	17.06	43	16.88	31.5	26.7

Table 6: Progress Rate (PR) and Environment Steps (ES) on EB-ALFRED (Seen)

TAPA is the first model specifically optimized for embodied multi-step planning, but it lacks visual perception capability; thus, we convert visual observations into textual descriptions for evaluation.

RoboBrain is a state-of-the-art VLM for embodied scenarios that integrates robotic and general multimodal data through a multi-stage training pipeline,leveraging long-horizon video and high-resolution image supervision to enhance manipulation and planning performance.

While there exist other VLMs designed for embodied settings, many of them are unavailable for public use, such as ReasonRFT(44), Embodied-R(63), and Embodied-Reasoner(61). Other models, such as EmbodiedGPT(30) and TAPA(52), exhibit poor generalization to new task distributions, achieving near-zero scores on Embench tasks and revealing a lack of transferable planning capabilities.

E.3 Experiment Results using supplementary metrics

In addition to task success rate, we provide supplementary evaluation results using two additional metrics: **Progress Rate (PR)** and **Environment Steps (ES)**.

Progress Rate (PR) quantifies the degree to which the agent completes the task, measured as the proportion of goal conditions satisfied by the final environment state. This metric provides a finer-grained signal than binary success, especially for partially completed tasks.

Environment Steps (ES) refers to the number of actions executed in the environment before task termination. A lower ES generally indicates more efficient planning and fewer redundant or failed actions.

Complete results across these metrics are reported in Appendix Tables 6 and 7.

Model	EB-Habitat (Unseen)													
	Avg		Base		Common		Complex		Visual		Spatial		Long	
	PR	ES	PR	ES	PR	ES	PR	ES	PR	ES	PR	ES	PR	ES
Closed-Source MLLMs														
Claude-3.5-Sonnet	70.9	10.7	98	6.54	69.5	10.46	75.5	10.6	75.1	10.74	45.2	9.44	62.1	16.42
Gemini-2.0-flash	38.5	13.41	76.5	8.56	31.5	12.9	34	15.66	32.7	13.7	37	12	19.8	17.66
GPT-4o	60.8	14.32	85.3	9.76	34	14.74	67.5	13.34	64.3	13.82	46.3	14.78	67.2	19.5
GPT-4o-mini	44.2	18.8	73.6	10.96	46	18.78	40.5	19.76	36.8	21.76	47.5	18.86	20.6	22.7
Open-Source General MLLMs														
Qwen2.5-VL-7B	19.05	12.58	44.5	10.64	6.5	14.9	17	11.12	6.4	14.12	28.8	11.74	11.1	12.94
InternVL2.5-8B	26	16.77	52.9	13.1	13	19.1	22	16.48	21.6	18.36	35.4	18.24	11.1	15.32
Open-Source Reasoning MLLMs														
R1-VL-7B	8.06	5.08	24.6	5.9	0	3.78	4	4.38	6	1.8	11.8	7.78	2	6.88
MM-Eureka-Qwen-7B	22.03	13.53	40.5	10.24	20.5	15.78	19	11.34	15.9	15.66	31.3	13.74	5	14.4
Open-Source Embodied MLLMs														
RoboBrain	20.18	10.68	39.1	8.08	9.5	9.08	21	11.3	12.9	13.9	31.1	11.48	7.5	10.24
Tapa	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Open-Source Embodied + Reasoning MLLMs														
Ours (Base)	19.05	12.58	44.5	10.64	6.5	14.9	17	11.12	6.4	14.12	28.8	11.74	11.1	12.94
Ours (SFT only)	20.05	12.40	38.75	10.62	7	12.3	19.5	12.76	16	11.24	34.6	15.26	4.5	12.26
Ours (SFT+RFT)	27.18	13.31	58.75	8.72	15	14.98	23	13.3	20	13.36	37	13.78	9.33	15.76

Table 7: Progress Rate (PR) and Environment Steps (ES) on EB-Habitat (Unseen)

F Case study and Visualization

F.1 Case Study

To better understand how our model performs embodied multi-step planning, we present detailed case studies illustrating its behavior and reasoning process. Specifically, we compare the outputs of our reinforcement-tuned model with the base Qwen2.5-VL model to highlight improvements in planning coherence and action correctness, we also present full multi-step execution trajectories from our model to show how it plans and interacts with the environment to complete specific tasks.

Figure 7 and Figure 8 show side-by-side comparisons between the two models in the EB-ALFRED and EB-Habitat environments, respectively. We observe that the base model often produces incomplete or illogical plans, while our model generates more structured and context-aware action sequences, along with interpretable reasoning steps.

Figure 9, Figure 10,Figure 11 and Figure 12 further visualize full planning trajectories executed by our model in representative tasks from EB-ALFRED and EB-Habitat. These examples demonstrate the model's ability to maintain long-horizon coherence, correctly interpret dynamic observations, and recover from intermediate failures.

F.2 Prompt

In this section, we document the full prompt formats used in both evaluation and training stages, including for EB-ALFRED, EB-Habitat, and our reinforcement fine-tuning (RFT) process.

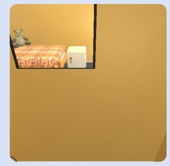
EB-ALFRED Prompt. The EB-ALFRED prompt is used for evaluating models within the EB-ALFRED environment of Embench. Our SFT stage also adopts this prompt format.

EB-Habitat Prompt. This prompt format is used in Embench's EB-Habitat environment, which differs from EB-ALFRED in simulator, object distribution, and language patterns.

RFT Training Prompt. During reinforcement fine-tuning, we adopt a custom prompt format. While still grounded in the same simulation environment, our RFT prompts include modifications in action representation and instruction phrasing. These differences help introduce broader data diversity and encourage the model to learn a more generalizable planning policy.

Instruction

You are a home-operating robot. Given a task, you must use a set of actions to achieve the desired outcome. The available action id and action names are: {}. Guidelines: {} Now the human instruction is: **Put the books on the desk.**The action history: Step {}, action id {}, {}, env feedback: {} You need to describe current visual state from the image, summarize interaction history and environment feedback, and reason why the last action or plan failed and did not finish the task, output your new plan to achieve the goal from current state. At the end, output the excutable plan with action ids(0 ~ {len(self.actions)-1}) from the available actions.The output json format should be {'visual_state_description':str, 'reasoning_and_reflection':str, 'language_plan':str, 'executable_plan':List[{'action_id':int, 'action_name':str}...]}



Qwen2-VL-7B-Instruct

```
"reasoning": "{
    \"visual_state_description\": \"The image shows a room with a bed, a dresser, and a safe. The bed has a blanket and a teddy bear on it. The dresser has a few items on it, including a book. The safe is closed and located near the dresser.\" ,

    \"reasoning_and_reflection\": \"The task is to put the books on the desk. The first step is to find the books. The books are located on the dresser. Next, pick up the books. The books are not inside a closed receptacle, so the 'pick up' action is valid. Then, put down the books on the desk. The desk is visible in the image, so the 'put down' action is valid. The final step is to close the safe. The safe is closed, so the 'close' action is valid.\" ,

    \"language_plan\": \"First, locate the books on the dresser. Once found, pick up the books. Next, search for the desk and place the books on it. Finally, find the safe and close it.\",

    \"executable_plan\": [
        {\"action_id\": 177, \"action_name\": \"find a Book_2\"},
        {\"action_id\": 111, \"action_name\": \"pick up the Book_2\"},
        {\"action_id\": 18, \"action_name\": \"find a Desk\"},
        {\"action_id\": 133, \"action_name\": \"put down the object in hand\"},
        {\"action_id\": 48, \"action_name\": \"find a Safe\"},
        {\"action_id\": 136, \"action_name\": \"close the Safe\"}
    ]
}"
```

The photo of the room does not show a dresser or any books

Although a safe appears in the scene, there is no indication that the task is related to the safe

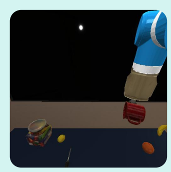
Ours

```
"reasoning": "{
    \"visual_state_description\": \"The image shows a room with a bed, a nightstand, and a wall. There are no visible books or desks in the image.\",

    \"reasoning_and_reflection\": \"The task is to put books on a desk. However, there are no books or desks visible in the image. This suggests that the books might be located elsewhere in the room or not present at all.\",

    \"language_plan\": \"First, locate the books. Once found, pick them up. Next, find the desk and place the books on it.\",

    \"executable_plan\": [
        {\"action_id\": 47, \"action_name\": \"find a Book\"},
        {\"action_id\": 94, \"action_name\": \"pick up the Book\"},
        {\"action_id\": 47, \"action_name\": \"find a Book\"},
        {\"action_id\": 94, \"action_name\": \"pick up the Book\"},
        {\"action_id\": 49, \"action_name\": \"find a Desk\"},
        {\"action_id\": 133, \"action_name\": \"put down the object in hand\"}
    ]
}"
```

Figure 7: Case comparison (Our model vs Base model) in EB-ALFRED

Instruction

You are a home-operating robot. Given a task, you must use a set of actions to achieve the desired outcome. The available action id and action names are: {}. Guidelines: {} Now the human instruction is: **Find a mug on the right counter and move it to the TV stand**. The action history: Step {}, action id {}, {}, env feedback: {} You need to describe current visual state from the image, summarize interaction history and environment feedback, and reason why the last action or plan failed and did not finish the task, output your new plan to achieve the goal from current state. At the end, output the excutable plan with action ids(0 ~ {len(self.actions)-1}) from the available actions.The output json format should be {'visual_state_description':str, 'reasoning_and_reflection':str, 'language_plan':str, 'executable_plan':List[{'action_id':int, 'action_name':str}...]}



Qwen2-VL-7B-Instruct

```
"action_description": "place at the table 2",

"reasoning": "{
    \"visual_state_description\": \"The robot is currently at the right counter in the kitchen. There is a mug on the right counter, but the robot cannot pick it up because it is not near the robot.\",

    \"reasoning_and_reflection\": \"The last action failed because the robot cannot pick up an object that is not near it. To find the mug, the robot needs to navigate to the right counter and then pick it up.\",

    \"language_plan\": \"The robot should first navigate to the right counter, then pick up the mug, and finally move to the TV stand and place the mug there.\",

    \"executable_plan\": [
        {\"action_id\": 10, \"action_name\": \"navigate to the right counter in the kitchen\"},
        {\"action_id\": 44, \"action_name\": \"pick up the mug\"},
        {\"action_id\": 8, \"action_name\": \"navigate to the TV stand\"},
        {\"action_id\": 51, \"action_name\": \"place at the TV stand\"}
    ]
}"
```

The action differs from the plan. Table 2's source is unsubstantiated, and repeated attempts in subsequent steps caused failure.

Ours

```
"action_description": "place at the TV stand",

"reasoning": "{
    \"visual_state_description\": \"The image shows a table with various items including a mug, a red cup, a sausage, a pair of scissors, and a small orange container. A blue and white object is partially visible on the right side.\",

    \"reasoning_and_reflection\": \"The last action involved navigating to the right counter in the kitchen, which was successful as indicated by the feedback. However, the task is to find a mug on the right counter and move it to the TV stand. Since the right counter is already visited, the next step is to identify the mug and pick it up.\",

    \"language_plan\": \"The task is to find a mug on the right counter and move it to the TV stand. The right counter has been navigated to, and the mug needs to be picked up and placed on the TV stand.\",

    \"executable_plan\": [
        {\"action_id\": 10, \"action_name\": \"navigate to the right counter in the kitchen\"},
        {\"action_id\": 44, \"action_name\": \"pick up the mug\"},
        {\"action_id\": 8, \"action_name\": \"navigate to the TV stand\"},
        {\"action_id\": 52, \"action_name\": \"place at the TV stand\"}
    ]
}"
```

Figure 8: Case comparison (Our model vs Base model) in EB-Habitat

Instruction	Examine a tv remote next to the light of a tall lamp.

"reasoning": "{"visual_state_description": "The image shows a room with a wooden table, a box on top of it, and a window with sunlight coming through. There is a tall lamp near the window, but no TV remote is visible in the scene.", "reasoning_and_reflection": "The task requires examining a TV remote under the light of a tall lamp. However, there is no TV remote visible in the image. The lamp is present, but the TV remote is not. This suggests that the TV remote might be out of view or not in the current scene.", "language_plan": "First, locate the TV remote. Once found, pick it up. Next, find the tall lamp and turn it on to examine the remote.", "executable_plan": [{"action_id": 68, "action_name": "find a RemoteControl"}, {"action_id": 116, "action_name": "pick up the RemoteControl"}, {"action_id": 27, "action_name": "find a FloorLamp"}, {"action_id": 153, "action_name": "turn on the FloorLamp"}]}"

Action 1

"env_step": 1
"env_feedback": "Last action executed successfully."
"action_id": 68
"action_description": "find a RemoteControl"

Action 2

"env_step": 2
"env_feedback": "Last action executed successfully."
"action_id": 116
"action_description": "pick up the RemoteControl"

Action 3

"env_step": 3
"env_feedback": "Last action executed successfully."
"action_id": 27
"action_description": "find a FloorLamp"

Action 4

"env_step": 4
"env_feedback": "Last action executed successfully."
"action_id": 153
"action_description": "turn on the FloorLamp"

Task Success

Figure 9: Our model's Full Trajectory execution in EB-ALFRED

Instruction Retrieve a apple and place it into the predetermined sofa.

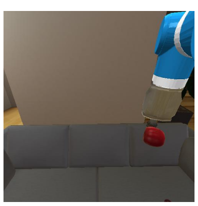
"reasoning": "{"visual_state_description": "The image shows a living room with a gray sofa, a coffee table, a staircase, and a blue object resembling a person. The apple is not visible in the image.", "reasoning_and_reflection": "The task is to retrieve an apple and place it on the sofa. However, the apple is not visible in the image. This suggests that the apple might be located elsewhere in the house, possibly in the kitchen or another room. The current plan should involve navigating to the kitchen to find the apple and then placing it on the sofa.", "language_plan": "The task is to find an apple and place it on the sofa. Since the apple is not visible in the current room, the robot should navigate to the kitchen to search for the apple.", "executable_plan": [{"action_id": 10, "action_name": "navigate to the right counter in the kitchen"}, {"action_id": 33, "action_name": "pick up the apple"}, {"action_id": 12, "action_name": "navigate to the sofa"}, {"action_id": 56, "action_name": "place at the sofa"}]}"

Action 1

"env_step": 1
"env_feedback": "Last action executed successfully."
"action_id": 10
"action_description": "navigate to the right counter in the kitchen"

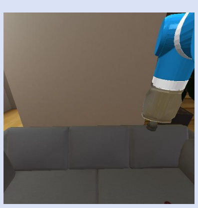
Action 2

"env_step": 2
"env_feedback": "Last action executed successfully and you are holding apple."
"action_id": 33
"action_description": "pick up the apple"



Action 3

"env_step": 3
"env_feedback": "Last action executed successfully."
"action_id": 12
"action_description": "navigate to the sofa"



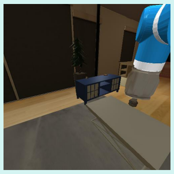
Action 4

"env_step": 4
"env_feedback": "Last action executed successfully and you are holding nothing."
"action_id": 56
"action_description": "place at the sofa"

Task Success

Figure 10: Our model's Full Trajectory execution example 1 in EB-Habitat

Instruction	Move a ball to the sofa.



"reasoning": "{"visual_state_description": "The image shows a living room with a sofa, a TV stand, and a table. A ball is located on the floor near the TV stand.", "reasoning_and_reflection": "The task is to move a ball from the floor to the sofa. The ball is clearly visible on the floor near the TV stand. The next step is to pick up the ball and then place it on the sofa.", "language_plan": "The plan is to first navigate to the ball, pick it up, and then place it on the sofa.", "executable_plan": [{"action_id": 8, "action_name": "navigate to the ball"}, {"action_id": 16, "action_name": "pick up the ball"}, {"action_id": 12, "action_name": "navigate to the sofa"}, {"action_id": 56, "action_name": "place at the sofa"}]}"

Action 1

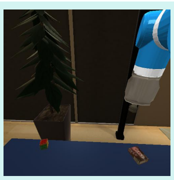
"env_step": 1
"env_feedback": "Last action executed successfully."
"action_id": 8
"action_description": "navigate to the TV stand"

Action 2

"env_step": 2
"env_feedback": "Last action is invalid. Robot cannot pick any object that is not near the robot. Navigate to other place to find the object."
"action_id": 16
"action_description": "pick up the ball"

Replanning

"reasoning": "{"visual_state_description": "The image shows a room with a plant in a pot on the floor, a sofa, and a table with some items on it. The ball is located on the floor next to the plant.", "reasoning_and_reflection": "The previous action was to navigate to the TV stand, but the ball was not found there. The ball is now visible on the floor next to the plant. The robot needs to pick up the ball and move it to the sofa.", "language_plan": "The robot should navigate to the ball, pick it up, and then move to the sofa to place the ball there.", "executable_plan": [{"action_id": 8, "action_name": "navigate to the TV stand"}, {"action_id": 16, "action_name": "pick up the ball"}, {"action_id": 12, "action_name": "navigate to the sofa"}, {"action_id": 56, "action_name": "place at the sofa"}]}"



Action 3

"env_step": 3
"env_feedback": "Last action executed successfully."
"action_id": 8
"action_description": "navigate to the TV stand"

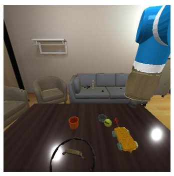
Action 4

"env_step": 4
"env_feedback": "Last action is invalid. Robot cannot pick any object that is not near the robot. Navigate to other place to find the object."
"action_id": 16
"action_description": "pick up the ball"

Replanning

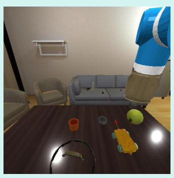
"reasoning": "{"visual_state_description": "The image shows a room with a plant in a pot, a blue and white object, a book, and a red and green object on the floor. The ball is not visible in the image.", "reasoning_and_reflection": "The previous attempts to pick up the ball were unsuccessful due to the robot being too far away from the ball. The ball is likely located on the floor, but the robot has not yet navigated to the correct position to pick it up. The current plan should involve navigating to the floor to find and pick up the ball.", "language_plan": "The robot needs to navigate to the floor to find the ball and then pick it up.", "executable_plan": [{"action_id": 6, "action_name": "navigate to the floor"}, {"action_id": 16, "action_name": "pick up the ball"}]}"

Figure 11: Our model's Full Trajectory execution example 2 (part 1) in EB-Habitat



Action 5

"env_step": 5
"env_feedback": "Last action executed successfully."
"action_id": 6
"action_description": "navigate to the table 1"

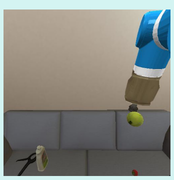


Action 6

"env_step": 6
"env_feedback": "Last action executed successfully and you are holding ball."
"action_id": 16
"action_description": "pick up the ball"

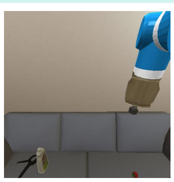
Replanning

"reasoning": "{"visual_state_description": "The robot is currently standing in front of the table 1, holding a ball. The ball is located on the table 1, along with a toy airplane, a cup, a bowl, and a lemon. The sofa is visible in the background.", "reasoning_and_reflection": "The robot has already picked up the ball from the table 1. The next step is to navigate to the sofa and place the ball there. The previous attempts to pick up the ball were successful, but the robot needs to move to the sofa to complete the task.", "language_plan": "The robot needs to navigate to the sofa and place the ball there.", "executable_plan": [{"action_id": 12, "action_name": "navigate to the sofa"}, {"action_id": 56, "action_name": "place at the sofa"}]}"



Action 7

"env_step": 7
"env_feedback": "Last action executed successfully."
"action_id": 12
"action_description": "navigate to the sofa"



Action 8

"env_step": 8
"env_feedback": "Last action executed successfully and you are holding nothing."
"action_id": 56
"action_description": "place at the sofa"

Task Success

Figure 12: Our model's Full Trajectory execution example 2 (part 2) in EB-Habitat

EB-ALFRED prompt

```
"<image>## You are a robot operating in a home. Given a task, you must
   accomplish the task using a defined set of actions to achieve the
   desired outcome.

## Action Descriptions and Validity Rules* Find: Parameterized by the
   name of the receptacle to navigate to. So long as the object is
   present in the scene, this skill is always valid * Pick up:
   Parameterized by the name of the object to pick. Only valid if the
   robot is close to the object, not holding another object, and the
   object is not inside a closed receptacle.* Put down: Parameterized by
    the name of the object to put down to a nearby receptacle. Only
   valid if the robot is holding an object. * Drop: Parameterized by the
    name of the object to put down. It is different from Put down action
   , as this does not guarantee the held object will be put into a
   specified receptacle. * Open: Parameterized by the name of the
   receptacle to open. Only valid if the receptacle is closed and the
   robot is close to the receptacle. * Close: Parameterized by the name
   of the receptacle to close. Only valid if the receptacle is open and
   the robot is close to the receptacle. * Turn on: Parameterized by the
    name of the object to turn on. Only valid if the object is turned
   off and the robot is close to the object. * Turn off: Parameterized
   by the name of the object to turn off. Only valid if the object is
   turned on and the robot is close to the object. * Slice:
   Parameterized by the name of the object to slice. Only valid if the
   object is sliceable and the robot is close to the object.

## The available action id (0 ~ 207) and action names are: {ALFRED
   ACTION LIST}

## Task Execution Example:{IN-CONTEXT TASK EXAMPLE}

## Guidelines 1. **Output Plan**: Avoid generating empty plan. Each plan
    should include no more than 20 actions. 2. **Visibility**: Always
   locate a visible object by the 'find' action before interacting with
   it. 3. **Action Guidelines**: Make sure match the action name and its
    corresponding action id in the output.newline Avoid performing
   actions that do not meet the defined validity criteria. For instance,
    if you want to put object in a receptacle, use 'put down' rather
   than 'drop' actions.  4. **Prevent Repeating Action Sequences**: Do
   not repeatedly execute the same action or sequence of actions.  Try
   to modify the action sequence because previous actions do not lead to
    success. 5. **Multiple Instances**: There may be multiple instances
   of the same object, distinguished by an index following their names,
   e.g., Cabinet_2, Cabinet_3. You can explore these instances if you do
    not find the desired object in the current receptacle. 6. **
   Reflection on History and Feedback**: Use interaction history and
   feedback from the environment to refine and improve your current plan
   . If the last action is invalid, reflect on the reason, such as not
   adhering to action rules or missing preliminary actions, and adjust
   your plan accordingly.

## Now the human instruction is: Rinse off a ladle and move it to the
   table. You are supposed to output in json. You need to describe
   current visual state from the image, output your reasoning steps and
   plan. At the end, output the action id (0 ~ 207) from the available
   actions to excute."
```

EB-Habitat prompt

<image>##You are a robot operating in a home. Given a task, you must accomplish the task using a defined set of actions to achieve the desired outcome.

Action Descriptions and Validity Rules: * Navigation: Parameterized by the name of the receptacle to navigate to. So long as the receptacle is present in the scene, this skill is always valid. * Pick: Parameterized by the name of the object to pick. Only valid if the robot is close to the object, not holding another object, and the object is not inside a closed receptacle. * Place: Parameterized by the name of the receptacle to place the object on. Only valid if the robot is close to the receptacle and is holding an object. * Open: Parameterized by the name of the receptacle to open. Only valid if the receptacle is closed and the robot is close to the receptacle. * Close: Parameterized by the name of the receptacle to close. Only valid if the receptacle is open and the robot is close to the receptacle.

The available action id (0 ~ 69) and action names are:{HABITAT ACTION LIST}

Task Execution Example:{IN-CONTEXT TASK EXAMPLE}

Guidelines 1. **Output Plan**: Avoid generating empty plan. Each plan should include no more than 20 actions. 2. **Visibility**: If an object is not currently visible, use the \"Navigation\" action to locate it or its receptacle before attempting other operations. 3. **Action Validity**: Make sure match the action name and its corresponding action id in the output. Avoid performing actions that do not meet the defined validity criteria. 4. **Prevent Repeating Action Sequences**: Do not repeatedly execute the same action or sequence of actions. Try to modify the action sequence because previous actions do not lead to success. 5. **Multiple Instances**: There may be multiple instances of the same object, distinguished by an index following their names, e.g., cabinet 2, cabinet 3. You can explore these instances if you do not find the desired object in the current receptacle. 6. **Reflection on History and Feedback**: Use interaction history and feedback from the environment to refine and enhance your current strategies and actions. If the last action is invalid, reflect on the reason, such as not adhering to action rules or missing preliminary actions, and adjust your plan accordingly.

Now the human instruction is: Move one of the pear items to the indicated sofa. You are supposed to output in json. You need to describe current visual state from the image, output your reasoning steps and plan. At the end, output the action id (0 ~ 69) from the available actions to excute."

Our RFT prompt

```
You are a robot operating in a home. Given a task, you must accomplish
   the task using a defined set of actions to achieve the desired
   outcome.

## Action Descriptions and Validity Rules * GotoLocation: Parameterized
   by the name of the target location or receptacle to navigate to.
   Always valid so long as the target exists in the scene. *
   PickupObject: Parameterized by the name of the object to pick up.
   Valid only if the robot is close to the object, is not holding
   anything, and the object is accessible. * PutObject: Parameterized by
    the name of the receptacle or surface where the held object will be
   placed. Valid only if the robot is holding an object. * ToggleObject:
    Parameterized by the name of the object whose state can be toggled (
   e.g., lamp, faucet). Valid only if the robot is close to the object.
   * CoolObject: Parameterized by the name of the object to cool.
   Requires the robot to be holding the object and near a cooling
   appliance such as a fridge. * SliceObject: Parameterized by the name
   of the object to slice. Requires that the object is slice-able and
   the robot holds an appropriate cutting tool. * CleanObject:
   Parameterized by the name of the object to clean. Requires the robot
   to be near a water source and the object supports cleaning. *
   HeatObject: Parameterized by the name of the object to heat. Requires
    the robot to be holding the object and near a heating appliance such
    as a microwave or stove.

## The available action id (0 ~ 224) and action names are:{OUR RFT
   ACTION LIST}

## Guidelines 1. **Output Plan**: Avoid generating empty plan. Each plan
    should include no more than 20 actions. 2. **Visibility**: Always
   locate a visible object by the 'goto' action before interacting with
   it. 3. **Action Guidelines**: Make sure the action name and its
   corresponding action id match in the output. Avoid performing actions
    that do not meet the defined validity criteria. 4. **Prevent
   Repeating Action Sequences**: Do not repeatedly execute the same
   action or sequence of actions. 5. **Multiple Instances**: There may
   be multiple instances of the same object, distinguished by an index
   following their names, e.g., Cabinet_2. 6. **Reflection on History
   and Feedback**: Use interaction history and feedback from the
   environment to refine and improve your current plan.

## Expected JSON output format```json {\"reasoning_and_reflection\": \"<
   string>\",  \"visual_state_description\": \"<string>\",  \"language_
   plan\": \"<string>\", \"executable_plan\": [ {\"action_id\": <int>,
   \"action_name\": \"<string>\"} ]}```

## Now the human instruction is: put a towel into a garbage can The
   history actions are: [{HISTORY LIST}] newlineConsidering the above
   interaction history and the current image state, to achieve the human
    instruction.newlineYou are supposed to output in json. You need to
   describe current visual state from the image, output your reasoning
   steps and plan. You shuold think carefully and output the
   comprehensive thought process in 'reasoning_and_reflection' part. At
   the end, output the action id (0 ~ 224) from the available actions to
    execute."
```

Part of EB-ALFRED Action list

```
action id 1: find a Potato, action id 2: find a Faucet, action id 3:
   find a Ottoman, action id 4: find a CoffeeMachine, action id 5: find
   a Candle, action id 6: find a CD, action id 7: find a Pan, action id
   8: find a Watch, action id 9: find a HandTowel, action id 10: find a
   SprayBottle, action id 11: find a BaseballBat, action id 12: find a
   CellPhone, action id 13: find a Kettle, action id 14: find a Mug,
   action id 15: find a StoveBurner, action id 16: find a Bowl, action
   id 17: find a Toilet, action id 18: find a DiningTable, action id 19:
    find a Spoon, action id 20: find a TissueBox, action id 21: find a
   Shelf, action id 22: find a Apple, action id 23: find a TennisRacket,
    action id 24: find a SoapBar, action id 25: find a Cloth, action id
   26: find a Plunger, action id 27: find a FloorLamp, action id 28:
   find a ToiletPaperHanger, action id 29: find a CoffeeTable, action id
    30: find a Spatula, action id 31: find a Plate, action id 32: find a
    Bed, action id 33: find a Glassbottle, action id 34: find a Knife,
   action id 35: find a Tomato, action id 36: find a ButterKnife, action
    id 37: find a Dresser, action id 38: find a Microwave, action id 39:
    find a CounterTop, action id 40: find a GarbageCan, action id 41:
   find a WateringCan, action id 42: find a Vase, action id 43: find a
   ArmChair, action id 44: find a Safe, action id 45: find a KeyChain,
   action id 46: find a Pot, action id 47: find a Pen, action id 48:
   find a Cabinet, action id 49: find a Desk, action id 50: find a
   Newspaper, action id 51: find a Drawer, action id 52: find a Sofa,
   action id 53: find a Bread, action id 54: find a Book, action id 55:
   find a Lettuce, action id 56: find a CreditCard, action id 57: find a
    AlarmClock, action id 58: find a ToiletPaper, action id 59: find a
   SideTable, action id 60: find a Fork, action id 61: find a Box,
   action id 62: find a Egg, action id 63: find a DeskLamp, action id
   64: find a Ladle, action id 65: find a WineBottle, action id 66: find
    a Pencil, action id 67: find a Laptop, action id 68: find a
   RemoteControl, action id 69: find a BasketBall, action id 70: find a
   DishSponge, action id 71: find a Cup, action id 72: find a SaltShaker
   , action id 73: find a PepperShaker, action id 74: find a Pillow,
   action id 75: find a Bathtub, action id 76: find a SoapBottle, action
    id 77: find a Statue, action id 78: find a Fridge, action id 79:
   find a Sink, action id 80: pick up the KeyChain, action id 81: pick
   up the Potato, action id 82: pick up the Pot, action id 83: pick up
   the Pen, action id 84: pick up the Candle, action id 85: pick up the
   CD, action id 86: pick up the Pan, action id 87: pick up the Watch,
   action id 88: pick up the Newspaper, action id 89: pick up the
   HandTowel, action id 90: pick up the SprayBottle, action id 91: pick
   up the BaseballBat, action id 92: pick up the Bread, action id 93:
   pick up the CellPhone, action id 94: pick up the Book, action id 95:
   pick up the Lettuce, action id 96: pick up the CreditCard, action id
   97: pick up the Mug, action id 98: pick up the AlarmClock, action id
   99: pick up the Kettle, action id 100: pick up the ToiletPaper
```

EB-Habitat Action list

```
action id 0: navigate to the cabinet 7, action id 1: navigate to the
   cabinet 6, action id 2: navigate to the cabinet 5, action id 3:
   navigate to the cabinet 4, action id 4: navigate to the refrigerator
   push point, action id 5: navigate to the chair 1, action id 6:
   navigate to the table 1, action id 7: navigate to the table 2, action
    id 8: navigate to the TV stand, action id 9: navigate to the sink in
    the kitchen, action id 10: navigate to the right counter in the
   kitchen, action id 11: navigate to the left counter in the kitchen,
   action id 12: navigate to the sofa, action id 13: navigate to the
   refrigerator, action id 14: navigate to the left drawer of the
   kitchen counter, action id 15: navigate to the right drawer of the
   kitchen counter, action id 16: pick up the ball, action id 17: pick
   up the clamp, action id 18: pick up the hammer, action id 19: pick up
    the screwdriver, action id 20: pick up the padlock, action id 21:
   pick up the scissors, action id 22: pick up the block, action id 23:
   pick up the drill, action id 24: pick up the spatula, action id 25:
   pick up the knife, action id 26: pick up the spoon, action id 27:
   pick up the plate, action id 28: pick up the sponge, action id 29:
   pick up the cleanser, action id 30: pick up the plum, action id 31:
   pick up the pear, action id 32: pick up the peach, action id 33: pick
    up the apple, action id 34: pick up the lemon, action id 35: pick up
    the can, action id 36: pick up the box, action id 37: pick up the
   banana, action id 38: pick up the strawberry, action id 39: pick up
   the lego, action id 40: pick up the rubriks cube, action id 41: pick
   up the book, action id 42: pick up the bowl, action id 43: pick up
   the cup, action id 44: pick up the mug, action id 45: pick up the
   orange, action id 46: pick up the lid, action id 47: pick up the toy
   airplane, action id 48: pick up the wrench, action id 49: place at
   the chair 1, action id 50: place at the table 1, action id 51: place
   at the table 2, action id 52: place at the TV stand, action id 53:
   place at the sink in the kitchen, action id 54: place at the right
   counter in the kitchen, action id 55: place at the left counter in
   the kitchen, action id 56: place at the sofa, action id 57: place at
   the refrigerator, action id 58: place at the left drawer of the
   kitchen counter, action id 59: place at the right drawer of the
   kitchen counter, action id 60: open the refrigerator, action id 61:
   close the refrigerator, action id 62: open the cabinet 7, action id
   63: open the cabinet 6, action id 64: open the cabinet 5, action id
   65: open the cabinet 4, action id 66: close the cabinet 7, action id
   67: close the cabinet 6, action id 68: close the cabinet 5, action id
    69: close the cabinet 4
```

Part of Our RFT Action list

```
action id 1: goto apple, action id 2: goto armchair, action id 3: goto
   baseballbat, action id 4: goto basketball, action id 5: goto
   bathtubbasin, action id 6: goto bed, action id 7: goto bowl, action
   id 8: goto box, action id 9: goto bread, action id 10: goto
   butterknife, action id 11: goto cabinet, action id 12: goto candle,
   action id 13: goto cart, action id 14: goto cellphone, action id 15:
   goto cloth, action id 16: goto coffeemachine, action id 17: goto
   coffeetable, action id 18: goto countertop, action id 19: goto
   creditcard, action id 20: goto cup, action id 21: goto desk, action
   id 22: goto desklamp, action id 23: goto diningtable, action id 24:
   goto dishsponge, action id 25: goto drawer, action id 26: goto
   dresser, action id 27: goto egg, action id 28: goto floorlamp, action
    id 29: goto fork, action id 30: goto fridge, action id 31: goto
   garbagecan, action id 32: goto handtowelholder, action id 33: goto
   keychain, action id 34: goto knife, action id 35: goto laptop, action
    id 36: goto lettuce, action id 37: goto microwave, action id 38:
   goto mug, action id 39: goto newspaper, action id 40: goto ottoman,
   action id 41: goto pan, action id 42: goto pen, action id 43: goto
   pencil, action id 44: goto plate, action id 45: goto plunger, action
   id 46: goto pot, action id 47: goto potato, action id 48: goto
   remotecontrol, action id 49: goto safe, action id 50: goto shelf,
   action id 51: goto sidetable, action id 52: goto sinkbasin, action id
    53: goto soapbar, action id 54: goto soapbottle, action id 55: goto
   sofa, action id 56: goto spatula, action id 57: goto spoon, action id
    58: goto statue, action id 59: goto stoveburner, action id 60: goto
   tennisracket, action id 61: goto tissuebox, action id 62: goto toilet
   , action id 63: goto toiletpaper, action id 64: goto
   toiletpaperhanger, action id 65: goto tomato, action id 66: goto vase
   , action id 67: goto watch, action id 68: goto wateringcan, action id
    69: pickup alarmclock, action id 70: pickup apple, action id 71:
   pickup baseballbat, action id 72: pickup basketball, action id 73:
   pickup book, action id 74: pickup bowl, action id 75: pickup box,
   action id 76: pickup bread, action id 77: pickup butterknife, action
   id 78: pickup candle, action id 79: pickup cd, action id 80: pickup
   cellphone, action id 81: pickup cloth, action id 82: pickup
   creditcard, action id 83: pickup cup, action id 84: pickup dishsponge
   , action id 85: pickup egg, action id 86: pickup fork, action id 87:
   pickup glassbottle, action id 88: pickup handtowel, action id 89:
   pickup kettle, action id 90: pickup keychain, action id 91: pickup
   knife, action id 92: pickup ladle, action id 93: pickup laptop,
   action id 94: pickup lettuce, action id 95: pickup mug, action id 96:
    pickup newspaper, action id 97: pickup pan, action id 98: pickup pen
   , action id 99: pickup pencil, action id 100: pickup peppershaker,
```