On-Policy Reinforcement Fine-Tuning with Offline Reward for Multi-Step Embodied Planning

Di Wu[1], Jiaxin Fan[1], Chloe Gu[2], Guanbo Wang[3], Wei Yin[4], Wenhao Li[1,†], Bo Jin[1,†]
[1]Tongji University [2]Duke University [3]Tsinghua University [4]Bank of Communications
{wu2002, 2253538, whli, bjin}@tongji.edu.cn
chloe.qu@duke.edu, wanggb23@mails.tsinghua.edu.cn
yinw_8@bankcomm.com

Abstract

Embodied planning requires agents to make coherent multi-step decisions based on dynamic visual observations and verbal goals. While recent vision-language models (VLMs) excel at static perception tasks, they struggle in interactive environments. Reinforcement learning (RL) offers a natural way to address this limitation, yet online RL approaches suffer from costly interaction and sparse rewards in embodied settings. This paper introduces **ORBIT**, an **O**n-policy **R**einforcement fine-tuning (RFT) framework with offline rewards for Em**B**od**I**ed **T**ask Planning, that preserves the generalization benefits of RFT while addressing the challenges of costly interaction and sparse rewards, supported by solid theoretical guarantees. Our approach is evaluated on EmbodiedBench, a recent benchmark for interactive embodied tasks, covering both in-domain and out-of-domain scenarios. Experimental results show that ORBIT achieves SOTA performance on EB-ALFRED, outperforming all closed-source and online-RL-based methods, while being substantially more efficient in training speed and computational cost, remaining robust to sub-optimal expert trajectories, and exhibiting strong generalization to unseen environments. We released all code and data at https://github.com/mail-taii/Reinforced-Reasoning-for-Embodied-Planning.

1 Introduction

Embodied task planning serves as a cornerstone in hierarchical embodied AI systems (Shi et al., 2025b; Zhang et al., 2024a), where intelligent agents must not only perceive their environment but also reason and act within it to accomplish complex, real-world tasks (Duan et al., 2022). Unlike low-level controllers that govern precise trajectory execution (Zawalski et al., 2024; Kim et al., 2025), high-level planning is responsible for formulating coherent action sequences that translate complex instructions into manageable sub-tasks (Wu et al., 2023). While conventional language-based reasoning is confined to static, text-driven contexts (Lightman et al., 2023; Ye et al., 2025; Shao et al., 2024), 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 (Zhang et al., 2024b), they exhibit substantial limitations when applied to **multi-step interactive** embodied planning. Empirical analyses reveal that even SOTA VLMs, which excel in image captioning or visual question answering, struggle to maintain coherent decision sequences in dynamic environments (Yang et al., 2025). These observations highlight a critical gap: strong static VLM performance does not ensure robust multi-step planning under sparse feedback and the generalization demands of dynamic embodied settings.

Recent studies indicate that RL, when applied as a post-training paradigm for foundation models (i.e., reinforcement fine-tuning, RFT), can effectively enhance model performance and generalization, with distinct advantages over supervised fine-tuning (SFT) (Guo et al., 2025). Building on these advances, RFT has emerged as a key paradigm for embodied AI, with a growing body of work (Chen et al., 2025a; Zang et al., 2025; Li et al., 2025) demonstrating its importance.

While there is limited prior work applying RFT to embodied task planning, a straightforward methodology is online RL, where the model interacts with the environment during training, receives rewards from the environment or simulator, and uses the collected experience to optimize a foundation model. However, this approach faces two fundamental challenges. **First, acquiring reward signals through online interaction is prohibitively**

†Corresponding authors.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 39277–39307

expensive. During online RL training, the model must interact with the environment to obtain reward signals while undergoing large-scale fine-tuning, which incurs heavy computational overhead in embodied simulators and becomes impractical when extending to real-world scenarios. **Second, online RL-based RFT becomes challenging in multi-step embodied decision-making.** In embodied environments, reward signals are typically sparse and delayed, as simulators often provide only a terminal success or failure signal after many interaction steps. This severely weakens credit assignment and makes optimization over long horizons difficult (Wang et al., 2025; Zeng et al., 2025). As a result, training with online RL becomes unstable and inefficient for embodied task planning, due to its inherently multi-step nature.

To address the costly interaction and sparse reward problems, we propose *ORBIT*, an *on-policy reinforcement fine-tuning framework with offline rewards* for multi-step embodied planning. Instead of acquiring rewards through costly environment interaction, ORBIT computes rewards by prefix-matching on-policy rollouts against offline expert trajectories and optimizes the policy via GRPO(Shao et al., 2024), mitigating expensive interaction and sparse rewards while preserving the generalization and reasoning benefits of reinforcement fine-tuning. We further provide a theoretical analysis showing that our offline reward formulation differs fundamentally from SFT yet retains the essential properties of on-policy RFT; in particular, the induced policy is bounded relative to that learned with true online interaction, demonstrating that lightweight expert-based rewards can effectively substitute for environment feedback. Empirically, evaluations on EmbodiedBench (Yang et al., 2025) show that our approach achieves **state-of-the-art planning performance** with an average success rate of **72.4%** on EB-ALFRED, outperforming the strongest closed-source baseline Claude-3.5-Sonnet (65.2%) and the best online-RL method (67.2%)(Chen et al., 2025a), while requiring only approximately **25%** of the training time of online RL baselines and remaining robust to sub-optimal expert trajectories. Together, these results demonstrate that offline reward-driven RFT provides an efficient, robust, and scalable alternative to online RL for embodied task planning.

Our contributions are as follows: 1) We design ORBIT, an on-policy RFT framework with offline rewards for embodied task planning; this framework preserves the generalization benefits of RFT while mitigating the costly interaction and sparse-reward issues of online RL. 2) We propose a simple yet effective offline reward design. We theoretically show that the policy induced by offline rewards admits a bounded gap to that learned with true online interaction, and demonstrate that this reward is robust to sub-optimal expert trajectories and outperforms more complex reward formulations. 3) We conduct extensive evaluations on EmbodiedBench, showing that ORBIT achieves the best overall performance on EB-ALFRED, surpassing all closed-source models (e.g., Claude-3.5-Sonnet and GPT-4o) and online-RL-based methods, while requiring only approximately 25% of the training cost of online RL and exhibiting strong generalization to unseen domains.

2 Methodology

2.1 Problem Formulation

We formulate embodied task planning as a multi-step, partially observable decision-making process. At each time step t, the agent receives a visual observation $o_t \in \mathcal{O}$ and executes an action $a_t \in \mathcal{A}$, forming a history $h_t = \{o_0, a_0, \ldots, o_t\}$.

Given a task instruction $g \in \mathcal{G}$ specified by a language command L, task success is determined by a set of binary goal conditions $\mathcal{C}(g)$. The agent produces a trajectory $e = (g, o_0, a_0, \ldots, o_n, a_n)$ and receives a sparse task-level reward $r(e) = \mathbb{I}[\mathcal{C}(g) \subseteq \{\text{True}\}]$.

We parameterize the policy π_θ with a vision-language model (VLM), which samples actions as $a_{t+1} \sim \pi_\theta(\cdot \mid o_t, h_t, L, P)$, where P denotes a fixed prompt. The objective is to maximize the expected task success rate, i.e., $\max_\theta \mathbb{E}_{e\sim\pi_\theta}[r(e)]$.

2.2 On-policy RFT for Embodied Planning with Offline Reward

While RFT has been shown to retain prior knowledge and elicit reasoning compared to SFT, its application to embodied planning poses unique challenges. First, acquiring rewards via online interaction is prohibitively expensive, which scale poorly in LLM post-training and impractical for real-world scenarios. Second, embodied simulators expose only an episodic success or failure signal, which poses the sparse reward problem for multi-step embodied planning.

To address these limitations, we adopt ORBIT, an *offline reward* approach that avoids online exe-

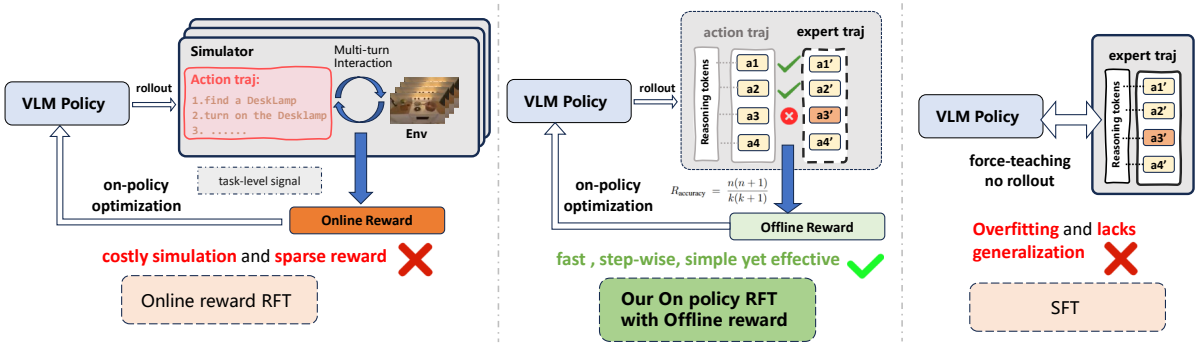


Figure 1: Comparison of *ORBIT*, our on-policy RFT framework with offline rewards, against other methods. Our approach avoids costly simulation overhead and sparse reward issues, while achieving stronger generalization through reinforcement fine-tuning.

cution. Instead of collecting interactive feedback, we compute reward by comparing on-policy rollouts to expert trajectories. This design not only circumvents costly simulator interaction but also alleviates reward sparsity while retaining the generalization advantages over SFT. We will present the methodological overview in this section and provide deeper analysis in Section 3.

MDP Formulation Conventional online RL for embodied planning is typically formulated as an episodic MDP, where at each step the agent samples an action $a_{t+1} \sim \pi_\theta(\cdot \mid o_t, h_t, L, P)$, receives a new observation $o_{t+1} \sim \mathcal{P}(\cdot \mid o_t, a_{t+1})$ through environment transition, and repeats this process until termination, forming a trajectory $e = (g, o_0, a_0, \ldots, o_n, a_n)$ with a sparse task-level reward. This formulation requires tight coupling with a simulator to provide state transitions and visual observations. Notably, our evaluation protocol strictly follows this same MDP.

In contrast, our offline reward training induces a fundamentally different decision process. Given an expert trajectory, each training state is defined as $s_n = (g, o_n, a_{0:n-1})$, where observations and histories are fixed from data. Conditioned on s_n, the policy predicts the entire remaining action sequence as a single decision, $\hat{a}_{n:} \sim \pi_\theta(\cdot \mid s_n, L, P)$. The reward is then computed offline by comparing $\hat{a}_{n:}$ with the expert reference, without invoking any environment transition. As a result, the training MDP reduces to a single-step decision process (i.e., a contextual bandit), while remaining on-policy since rollouts are sampled from the current policy.

Offline Expert Trajectory Construction. We construct our offline expert dataset from ALFRED (Shridhar et al., 2020), which provides complete ground-truth household trajectories in simulation. It intentionally differs from the evaluation setup so that RFT must generalize beyond a matched format (Appendix F). Each expert trajectory $e = (g, o_0, a_0, ..., o_k, a_k)$ 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 $43,898$ RFT training samples.

Reward Design via Expert Trajectory. We define an *offline* reward from expert trajectories: given the model's rollout and its expert action sequence to score the reward for policy optimization.

Let $\hat{a} = \{a_1, \ldots, a_m\}$ be the predicted sequence and $a^* = \{a_1^*, \ldots, a_k^*\}$ the expert reference. Define $n = \max\{i : a_j = a_j^* \; \forall j \le i\}$ as the length of the longest correct prefix. We use a smooth, long-horizon–aware shaping term:

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

which emphasizes longer consecutive matches and yields stable gradients. Despite its simplicity, the quadratic prefix-based reward proves highly effective, outperforming more complex alternatives. Further analysis is provided in Section 4.3.

Complementary to correctness reward, we incorporate a format reward as an auxiliary signal. Prior

studies have shown that structural regularization improves stability and prevents degenerate outputs. Following prior RFT practices (Meng et al., 2025; Tan et al., 2025), we set the maximum format reward to 0.5 and the accuracy reward to 1.0, yielding the final composite reward. We provide the detailed format reward components in the Appendix F.2.

$$R(\text{response}, \text{answer}) = R_{\text{accuracy}}(\text{response}, \text{answer}) + R_{\text{format}}(\text{response}). \tag{2}$$

Training Pipeline. For the overall pipeline, we first perform supervised fine-tuning to establish structured planning priors, and then apply reinforcement fine-tuning as the main part, with the offline reward to strengthen multi-step reasoning. Concretely, the model generates candidate rollouts, which are scored against expert trajectories by our reward function; policy parameters are then updated using GRPO with an additional data filtering strategy to ensure stable optimization.

Step 1: SFT. We initialize the VLM by distilling expert-style trajectories, training via maximum likelihood to align with commonsense patterns and structured conventions. This provides a strong initialization for downstream reinforcement learning. More details are provided in Appendix E.

Step 2: On-Policy RFT with GRPO. Building on the offline reward signals, we adopt Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to optimize the policy. For each prompt, the model samples multiple candidate responses, each scored by the reward function. Relative advantages are then computed within the group, and gradients encourage high-reward responses while regularizing against deviation from a reference policy.

Step 3: Data Filtering. To ensure informative and stable gradients, we incorporate an online filtering strategy during RFT. Prompt groups with too few (< 0.1) or too many (> 0.9) perfect-reward responses are discarded, maintaining a balanced learning signal and stabilizing training.

Conclusion. In this section, we introduced our RFT framework for multi-step embodied planning. The core idea is to compute **offline** rewards by prefix-based comparison with expert trajectories, and to optimize the policy through **on-policy** GRPO. This direction also aligns with emerging trends in the broader LLM community toward scalable policy optimization with offline signals (Deng et al., 2024, 2025). In the next section, we provide a deeper theoretical analysis of these properties, clarifying the effectiveness of our approach and its distinctions from existing methods.

3 Theoretical Analysis

We conduct a deeper analysis from two perspectives: (1) the distinction and advantages of offline reward RFT over SFT, given that both rely on offline ground-truth datasets for supervision (Section 3.1); and (2) the validity of using offline trajectories as a proxy for real environmental feedback, with an upper bound on the theoretical gap between the two methods (Section 3.2)

3.1 Formal Comparison of SFT and RFT

We present the formulations of SFT, offline reward RFT, and online reward RFT as follows.

Supervised Fine-Tuning (SFT). Here every output token (reasoning or action) is directly supervised, enforcing strict imitation:

$$\mathcal{L}_{\text{SFT}}(\pi) = -\,\mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\beta(\cdot|x)}\left[\sum_{t=1}^{k}\Big(\underbrace{\log\pi(y_r \mid x)}_{\text{reasoning tokens}} + \underbrace{\log\pi(y_a \mid x)}_{\text{action tokens}}\Big)\right]. \tag{3}$$

While effective, SFT limits reasoning diversity and generalization, whereas RFT alleviates these limitations. The following two formulations present the details of RFT, whose main difference lies in the reward source (online vs. offline).

Online reward RFT

$$\mathcal{J}^{\text{on}}_{\text{RFT}}(\pi) = \mathbb{E}_{y\sim\pi(\cdot|x)}\Big[\underbrace{A_{\text{env}}(x, y_a)}_{\text{from env. reward}}\ \log\pi(y \mid x)\Big] - \beta\, D_{\text{KL}}(\pi \,\|\, \pi_{\text{ref}})\,. \tag{4}$$

Offline reward RFT (ours).

$$\mathcal{J}^{\text{off}}_{\text{RFT}}(\pi) = \mathbb{E}_{y\sim\pi(\cdot|x)}\Big[\underbrace{A_{\text{off}}(x, y_a;\, y_E)}_{\text{from expert-comparison}}\ \log\pi(y \mid x)\Big] - \beta\, D_{\text{KL}}(\pi \,\|\, \pi_{\text{ref}})\,. \tag{5}$$

where y_E is the expert trajectory and A_{off} is computed from an offline *reward function* $r_{\text{off}}(x, y; y_E)$, which in our setting is instantiated via prefix-based matching between the rollout and the expert trajectory, followed by group-wise standardization to form advantages.

Although both SFT and our offline reward RFT rely on offline datasets for supervision, the formulations reveal two key differences:

First, SFT provides token-level supervision over all output tokens (including intermediate reasoning), enforcing strict imitation but limiting reasoning diversity and increasing the risk of path overfitting. In contrast, both online and offline RFT optimize a sequence-level scalar advantage that depends only on action tokens, shaping reasoning tokens indirectly and thus preserving flexibility. **Second**, unlike SFT's fixed data distribution, RFT performs *on-policy* optimization under the current policy and explicitly leverages negative examples through negative advantages ($A < 0$), which has been shown to improve generalization and mitigate forgetting (Shenfeld et al., 2025). Despite using an expert-derived offline reward, our approach retains these core RFT properties, aligning its training dynamics with online reward RFT and explaining its stronger empirical generalization (Section 4.2).

3.2 Optimal-Policy Distance Upper Bound.

In our approach, offline rewards replace real environmental feedback, raising the natural question of whether such a substitution is valid. In this section, we provide a theoretical upper bound on the distance between the policies induced by the two reward formulations:

For a fixed prompt x, consider the KL-regularized objective

$$\begin{aligned}\mathcal{J}_r(\pi \mid x) &= \mathbb{E}_{y\sim\pi(\cdot|x)}\big[\, r(y) \,\big] \\ &\quad - \ \beta_{\text{eff}}\, D_{\text{KL}}\big(\pi(\cdot \mid x) \,\|\, \pi_{\text{ref}}(\cdot \mid x)\big).\end{aligned} \tag{6}$$

where $\beta_{\text{eff}} > 0$ and $\pi_{\text{ref}}(\cdot \mid x)$ has full support. It is standard that the maximizer has the Gibbs form

$$\pi_r^\star(y \mid x) \ = \ \frac{\pi_{\text{ref}}(y \mid x)\ \exp\big(r(y)/\beta_{\text{eff}}\big)}{\sum_{y'} \pi_{\text{ref}}(y' \mid x)\ \exp\big(r(y')/\beta_{\text{eff}}\big)}. \tag{7}$$

Let y_E be a fixed expert trajectory of length k. Define the *expert-comparison* reward r_{exp} via the longest correct prefix, While the *environment* success reward is not tied to a single trajectory but to the whole success set $\mathcal{Y}_{\text{succ}}(x)$.

$$\begin{aligned} r_{\text{exp}}(y) &= \frac{n(y)\big(n(y)+1\big)}{k(k+1)}, \\ r_{\text{env}}(y) &= \mathbb{I}\{\, y \in \mathcal{Y}_{\text{succ}}(x) \,\}. \end{aligned} \tag{8}$$

Let the *minimal expert-prefix among all successful trajectories* be $n_{\min}(x) = \min_{y\in\mathcal{Y}_{\text{succ}}(x)} n(y)$. Then the smallest expert-comparison reward among successful trajectories is

$$\min_{y\in\mathcal{Y}_{\text{succ}}(x)} r_{\text{exp}}(y) = \frac{n_{\min}(x)\big(n_{\min}(x)+1\big)}{k(k+1)}. \tag{9}$$

We upper bound the pointwise gap:

$$\begin{aligned}\delta_\star(x) &= \big\| r_{\text{exp}} - r_{\text{env}} \big\|_\infty \\ &\le \max\left\{ 1 - \frac{n_{\min}(x)\big(n_{\min}(x)+1\big)}{k(k+1)},\ \frac{k-1}{k+1} \right\}.\end{aligned} \tag{10}$$

The first term controls the max discrepancy *among successes* ($y \in \mathcal{Y}_{\text{succ}}$), while the second term controls the max discrepancy *among failures* ($y \notin \mathcal{Y}_{\text{succ}}$), using $\max_{n\le k-1} \frac{n(n+1)}{k(k+1)} = \frac{k-1}{k+1}$.

Lemma 1 (Softmax stability). Let $r_1, r_2 : \mathcal{Y} \to \mathbb{R}$ be bounded rewards with $\|r_1 - r_2\|_\infty \le \delta$. Let $\pi_i^\star$ be the maximizer of (6) with $r = r_i$ (thus of the form (7)). Then

$$D_{\text{KL}}\big(\pi_1^\star \,\|\, \pi_2^\star\big) \ \vee \ D_{\text{KL}}\big(\pi_2^\star \,\|\, \pi_1^\star\big) \ \le \ \frac{2\delta}{\beta_{\text{eff}}}. \tag{11}$$

Detailed proof will be given in Appendix D

Proposition 1 (Bounded Optimality Gap). Let $\pi_{\text{exp}}^\star$ and $\pi_{\text{env}}^\star$ be the optimal policies induced by the expert-derived reward r_{exp} and the environment reward r_{env} under the regularized objective in (6), respectively. Then, with $\delta_\star(x)$ in (10), we get the policy distance by Pinsker's inequality,

$$\begin{aligned} D_{\text{KL}}\big(\pi_{\text{exp}}^\star \,\|\, \pi_{\text{env}}^\star\big) &\le \frac{2\,\delta_\star(x)}{\beta_{\text{eff}}}, \\ \big\|\pi_{\text{exp}}^\star - \pi_{\text{env}}^\star\big\|_{\text{TV}} &\le \sqrt{\tfrac{1}{2}\, D_{\text{KL}}\big(\pi_{\text{exp}}^\star \,\|\, \pi_{\text{env}}^\star\big)}, \\ &\le \sqrt{\frac{\delta_\star(x)}{\beta_{\text{eff}}}}. \end{aligned} \tag{12}$$

Inequalities (12) establish that the optimal policy under the offline *expert-comparison* reward remains provably close to the environment-optimal policy. Importantly, this does not imply that the sparse environment reward is always superior: when a trajectory ultimately fails, the environment reward collapses to zero, whereas our expert-based reward can still assign a positive signal proportional to prefix similarity. This alleviates **reward sparsity problem** in such multi-step settings while the bounded discrepancy guarantees that optimization under the offline reward remains directionally aligned with the online environment objective.

Model	EB-ALFRED (Seen)						EB-Habitat (Unseen)					
	Avg	Base	Common	Complex	Visual	Spatial	Avg	Base	Common	Complex	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
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
Training-based Embodied MLLMs												
WAP-7B	61.2	66	**62**	70	56	52	9.6	22	6	8	4	8
ERA-7B	67.2	74	54	70	70	**68**	18	40	10	14	8	18
VAGEN-3B	52.8	70	38	70	58	28	11.2	28	6	6	8	8
Our Method												
ORBIT-7B (base)	2.0	4	2	2	2	0	14	38	4	12	4	12
ORBIT-7B (SFT)	57.3	62.7	48	62	62	52	13.6	34	2	10	10	12
ORBIT-7B (SFT+RFT)	**72.4**	**86**	**62**	**82**	**72**	60	**22.4**	56	8	18	16	14

Table 1: Side-by-side comparison: left EB-ALFRED (seen during SFT, with only the Base subset seen during RFT) vs. right EB-Habitat (fully unseen during both SFT and RFT).

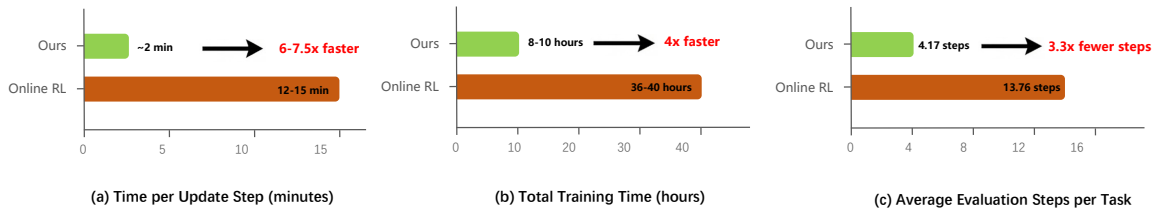


Figure 2: Efficiency of *ORBIT*. By avoiding costly online environment interaction during training, *ORBIT* achieves approximately 6× faster average update time per step and about 4× shorter overall training time than online RL.

4 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 for multi-step embodied task planning, in terms of both effectiveness and efficiency?* (Section 4.1)

(Q2) *Is reinforcement fine-tuning necessary and uniquely beneficial, especially compared to supervised fine-tuning?* (Section 4.2)

(Q3) *Is our prefix-based offline reward effective and robust compared to more complex reward formulations and under suboptimal expert trajectories?* (Section 4.3)

4.1 Evaluation on EmbodiedBench (Q1)

4.1.1 Experimental Settings and Baselines

We evaluate ORBIT on **EmbodiedBench** (Yang et al., 2025), a benchmark designed for interactive, multi-step embodied planning. EmbodiedBench includes two main environments: *EB-ALFRED*, built on ALFRED and AI2-THOR, and *EB-Habitat*, based on Habitat 2.0 rearrangement tasks. Tasks are further organized into six subsets, including a *Base* set and several augmented variants (e.g., Common Sense, Complex Instruction, Spatial Awareness, Visual Appearance) that increase reasoning or perception difficulty.

Our model is built upon **Qwen2.5-VL-7B**. During supervised fine-tuning, the model is exposed to task instructions from all EB-ALFRED subsets. In contrast, reinforcement fine-tuning is conducted *exclusively* on the EB-ALFRED *Base* set. As a result, EB-Habitat constitutes a fully unseen environment, while EB-ALFRED tasks beyond the Base set remain unseen during RFT and differ in

action space details and execution dynamics, enabling a controlled evaluation of generalization.

We adopt **task success rate** as the primary evaluation metric. For each episode, the model generates a sequence of actions conditioned on the current egocentric observation, which are executed iteratively in the environment until all goal conditions are satisfied or a predefined step limit is reached.

We compare ORBIT against a diverse set of baselines, including proprietary models such as Claude-3.5-sonnet and open-source general VLMs like LLaMA-3.2-Vision-90B. We further compare our method with several recent training-based approaches for embodied tasks, including WAP-7B (Wang et al.), which is trained with data augmentation under a pure SFT paradigm, as well as VAGEN-3B (Wang et al., 2025) and ERA-7B (Chen et al., 2025a), both optimized using online reinforcement learning.

4.1.2 Main Results on EmbodiedBench

In-Domain Results (EB-ALFRED). We first evaluate ORBIT on EB-ALFRED. As shown in Table 1, our full model (SFT+RFT) achieves an average success rate of **72.4%**, establishing a new state of the art on this benchmark. It outperforms all closed-source models, including Claude-3.5-Sonnet (65.2%) and Gemini-2.0-Flash (50.8%), as well as all open-source and training-based baselines such as ERA (67.2%), which relies on more resource-intensive online reinforcement learning.

Notably, although our RL training data only includes the *Base* subset, the model consistently improves performance across all other task categories, indicating strong generalization within the same environment distribution.

Out-of-Domain Results (EB-Habitat). We further evaluate generalization on EB-Habitat, which is fully unseen during both SFT and RFT stages and differs substantially from ALFRED in scenes, objects, action space, and task structure. Despite this strict setting, our method demonstrates stable and competitive performance, outperforming all baseline models of comparable 7B scale, suggesting that the learned policy transfers beyond the training environment without overfitting.

4.1.3 Efficiency of Offline Reward

We evaluate whether offline reward–based reinforcement fine-tuning provides tangible advantages over online RL in terms of training efficiency and execution speed. The result is in fig2.

Variant	EB-ALFRED (Seen)		EB-Habitat (Unseen)	
	Avg	Base	Avg	Base
Base	2	4	14	38
SFT only	57.3	62.7	13.6	30
RFT only	14.4	22	17.6	40
RFT → SFT	59.2	72	11.4	30
SFT → SFT	66.8	71	11.6	22
SFT → RFT (ours)	**72.4**	**86**	**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.

By decoupling policy optimization from interactive environments, our offline reward formulation reduces training to standard VLM optimization with precomputed rewards. This avoids simulator stepping, environment resets, and synchronization overheads inherent to online RL. On EB-ALFRED, each update step takes approximately $\sim$2 minutes, compared to 12–15 minutes for online RL, yielding a 6–7.5$\times$ per-step speedup. As a result, our method converges within 8–10 hours, whereas online RL requires 36–40 hours, corresponding to an overall $\sim$4$\times$ reduction in wall-clock training time.

Also, offline training enables stable generation of multi-step action chunks, which is difficult to achieve in online RL due to rollout instability and compounding distributional shift. This capability directly translates to faster execution: our model completes tasks in an average of 4.17 evaluation steps, compared to 13.76 steps for single-step online policies, reducing interaction steps by 3.3$\times$.

4.2 RFT Generalizes While SFT Overfits (Q2)

Is Reinforcement Fine-Tuning Necessary? A key question is whether the gains from GRPO-based reinforcement fine-tuning arise from the optimization process itself or simply from additional trajectory exposure. To disentangle these factors, we compare five training strategies: **Base** (no tuning), **SFT only**, **RFT only**, **RFT→SFT**, **SFT→SFT**, and our proposed **SFT→RFT** pipeline. The SFT→SFT means conducting SFT , followed by additional SFT using the same trajectories during RFT, which isolates the effect of data exposure from optimization.

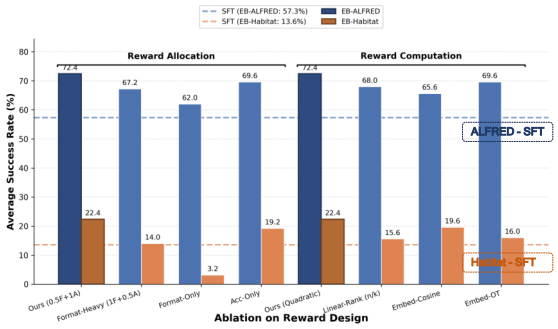


Figure 3: Ablation of our reward design. Comparison of different reward allocation and computation strategies.

RFT Data Quality	Avg	Base	Com	Cplx	Visual	Spatial
0% Perturbation (Standard)	72.4	86	62	82	72	60
10% Perturbation	69.6 ↓**3.87%**	82	62	76	68	60
20% Perturbation	64.0 ↓**11.60%**	82	52	76	62	48

Table 3: Impact of RFT data perturbation on EmbodiedBench performance.

RFT Data Scale	Avg	Base	Com	Cplx	Visual	Spatial
100% (Standard)	72.4	86	62	82	72	60
50% Data	68.4 ↓**5.52%**	78	60	80	70	54
20% Data	65.2 ↓**9.95%**	80	54	78	60	54

Table 4: Impact of offline RFT data scale on EmbodiedBench performance.

As shown in Table 2a, SFT→RFT achieves the best performance on both seen and unseen environments. While SFT→SFT yields moderate improvements over SFT only on seen tasks, it degrades generalization to unseen domains, highlighting the limitations of purely supervised optimization. Notably, RFT only performs poorly, indicating that RFT alone is insufficient without strong prior knowledge from supervised pre-alignment.

Does RFT Overfit to Embodied Benchmarks? To further evaluate the generalization capability of RFT, we assess whether fine-tuning on EmbodiedBench harms the model's performance on its original training domains besides embodied task planning. Specifically, we evaluate on **SpatialEval** (Wang et al., 2024), 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.

4.3 Ablation and Robustness Study on Reward Design (Q3)

We investigate both the *effectiveness* and *robustness* of our prefix-based offline reward through targeted ablations and controlled perturbation studies.

4.3.1 Reward Design Ablation

Reward allocation. We first study how different allocations between accuracy and format rewards affect performance, comparing our design (`1Accuracy+0.5Format`) with `Format-Heavy` (`1Format+0.5Accuracy`), `Format-Only`, and `Accuracy-Only`. As shown in Figure 3, our allocation consistently achieves the best results. The `Format-Only` fails to yield meaningful gains, and `Accuracy-Only` underperforms our approach, indicating that explicit format guidance is a necessary component for effective policy learning.

Reward computation. We further compare different reward formulations, including our quadratic prefix curve $n(n+1)/k(k+1)$, a linear variant n/k, and two embedding-based rewards: `Embed-Cosine` and `Embed-OT`. The embedding-based rewards compute similarity between rollout and expert trajectories using Qwen3-Embedding-0.8B, with cosine similarity or OT distance (via Sinkhorn (Cuturi, 2013)), respectively. As shown in Figure 3, our simple prefix-based reward consistently outperforms these alternatives. We attribute this to its structured yet lightweight design, which avoids spurious similarity matching and encourages gradual long-horizon alignment with expert behaviors.

4.3.2 Robustness to Sub-Optimal Expert Trajectories

In embodied planning, expert trajectories are rarely uniquely optimal, as tasks often admit multiple valid solutions. Our theoretical analysis (Section 2.3.2) shows that policy optimization remains bounded and directionally correct under non-unique expert trajectories. To empirically validate this robustness, we deliberately perturb the RFT training data by injecting sub-optimal noise, including irrelevant action insertions and redundant navigation loops. As shown in Table 3, the model remains robust under moderate noise: with 10% perturbed trajectories, performance drops only slightly to **69.6%**, while even at 20% perturbation it degrades gracefully to **64.0%**. These results indicate that our method does not rely on perfectly optimal expert data and can reliably learn effective planning

policies from partially sub-optimal trajectories. We further conduct a dataset-reduction study by randomly subsampling the offline RFT expert data to 50% and 20%. As shown in Table 4, performance degrades gracefully under reduced data, with 20% data still remaining competitive, suggesting that our method is robust to reduced data scale.

5 Conclusion

In this work, we introduce *ORBIT*, an on-policy RFT framework with offline rewards for multi-step embodied planning, which avoids costly simulator interaction and sparse rewards while preserving the generalization benefits of RFT. Supported by theoretical analysis and experiments on EmbodiedBench, our results highlight offline reward–driven reinforcement tuning as a scalable paradigm for embodied AI and other multi-step agentic RL settings, paving the way toward practical deployment in real-world applications.

Limitations

A natural next step is to extend our offline reward paradigm beyond embodied planning, applying it to broader scenarios such as vision-language-action (VLA) models and other multi-step reasoning tasks. In these settings, a hybrid approach that combines offline trajectory-based rewards with selective online feedback may further enhance adaptability while retaining stability.

In addition, our current focus lies on high-level embodied planning in simulation, producing structured action sequences that guide downstream controllers. While our method already demonstrates strong performance and generalization in benchmarks, deploying it on physical robotic platforms remains an important avenue for validation and integration with low-level control systems.

Our method also has limitations on ultra long-horizon tasks, where errors accumulate over extended action sequences and make planning increasingly difficult. Future work will explore methods such as dynamic action chunking and memory mechanisms to improve longer-horizon planning and robustness.

Acknowledgments

This work was supported by the NSFC (62406270) and the STCSM Shanghai Rising-Star Program (24YF2748800).

References

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, and 26 others. Do as i can, not as i say: Grounding language in robotic affordances. *Preprint*, arxiv:2204.01691 [cs].

Anthropic. Introducing claude 3.5 sonnet \ anthropic.

Hanyang Chen, Mark Zhao, Rui Yang, Qinwei Ma, Ke Yang, Jiarui Yao, Kangrui Wang, Hao Bai, Zhenhailong Wang, Rui Pan, and 1 others. 2025a. Era: Transforming vlms into embodied agents via embodied prior learning and online reinforcement learning. *arXiv preprint arXiv:2510.12693*.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. 2025b. Retaining by doing: The role of on-policy data in mitigating forgetting. *arXiv preprint arXiv:2510.18874*.

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. *Preprint*, arxiv:2311.15649 [cs].

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, and 1 others. 2024. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*.

Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26.

Google DeepMind. Introducing gemini 2.0: our new AI model for the agentic era.

Yihe Deng, I Hsu, Jun Yan, Zifeng Wang, Rujun Han, Gufeng Zhang, Yanfei Chen, Wei Wang, Tomas Pfister, Chen-Yu Lee, and 1 others. 2025. Supervised reinforcement learning: From expert trajectories to step-wise reasoning. *arXiv preprint arXiv:2510.25992*.

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

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

Xian Fu, Min Zhang, Peilong Han, Hao Zhang, Lei Shi, Hongyao Tang, and 1 others. 2024. What can vlms do for zero-shot embodied task planning? In *ICML 2024 Workshop on LLMs and Cognition*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jian Hu, Xibin Wu, Zilin Zhu, Weixun Wang, Dehao Zhang, Yu Cao, and 1 others. 2024. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*.

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. *Preprint*, arxiv:2311.17842 [cs].

Byeonghwi Kim, Jinyeon Kim, Yuyeong Kim, Cheolhong Min, and Jonghyun Choi. Context-aware planning and environment-aware memory for instruction following embodied agents. *Preprint*, arxiv:2308.07241 [cs].

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, and 1 others. 2025. Openvla: An open-source vision-language-action model. In *8th Annual Conference on Robot Learning*.

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

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, and 1 others. 2025. Simplevla-rl: Scaling vla training via reinforcement learning. *arXiv preprint arXiv:2509.09674*.

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

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

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, and 1 others. 2025. Mm-eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2503.07365*.

Meta. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models.

OpenAI. a. GPT-4o mini: advancing cost-efficient intelligence.

OpenAI. b. Hello GPT-4o | OpenAI.

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

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.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, and 1 others. 2019. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, and 1 others. 2025. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615*.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. 2025. Rl's razor: Why online reinforcement learning forgets less. *arXiv preprint arXiv:2509.04259*.

Junhao Shi, Zhaoye Fei, Siyin Wang, Qipeng Guo, Jingjing Gong, and Xipeng QIu. 2025a. World-aware planning narratives enhance large vision-language model planner. *arXiv preprint arXiv:2506.21230*.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, and 1 others. 2025b. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*.

Suyeon Shin, Sujin jeon, Junghyun Kim, Gi-Cheon Kang, and Byoung-Tak Zhang. Socratic planner: Inquiry-based zero-shot planning for embodied instruction following. *Preprint*, arxiv:2404.15190 [cs].

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. 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.

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.

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. *Preprint*, arxiv:2209.11302 [cs].

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

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

Qwen Team. 2025. Qwen2. 5-vl technical report.

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

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Chi Wan, Yiping Lu, Zhengyuan Yang, and 1 others. 2025. Vagen: Reinforcing world model reasoning for multi-turn vlm agents. *arXiv preprint arXiv:2510.16907*.

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. *Preprint*, arxiv:2503.10480 [cs].

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

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2025. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101.

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

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, and 1 others. 2025. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *arXiv preprint arXiv:2502.09560*.

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

Hongzhi Zang, Mingjie Wei, Si Xu, Yongji Wu, Zhen Guo, Yuanqing Wang, Hao Lin, Liangzhi Shi, Yuqing Xie, Zhexuan Xu, and 1 others. 2025. Rlinf-vla: A unified and efficient framework for vla+ rl training. *arXiv preprint arXiv:2510.06710*.

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

Siliang Zeng, Quan Wei, William Brown, Oana Frunza, Yuriy Nevmyvaka, Yang Katie Zhao, and Mingyi Hong. 2025. Reinforcing multi-turn reasoning in llm agents via turn-level credit assignment. In *ICML 2025 Workshop on Computer Use Agents*.

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

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

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

Wenqi Zhang, Mengna Wang, Gangao Liu, Xu Huixin, Yiwei Jiang, Yongliang Shen, Guiyang Hou, Zhe Zheng, Hang Zhang, Xin Li, and 1 others. 2025b. Embodied-reasoner: Synergizing visual search, reasoning, and action for embodied interactive tasks. *arXiv preprint arXiv:2503.21696*.

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

Hanqing Zhu, Zhenyu Zhang, Hanxian Huang, DiJia Su, Zechun Liu, Jiawei Zhao, Igor Fedorov, Hamed Pirsiavash, Zhizhou Sha, Jinwon Lee, and 1 others. 2025. The path not taken: Rlvr provably learns off the principals. *arXiv preprint arXiv:2511.08567*.

A Appendix Contents

- Section B: Use of LLMs
- Section C: Related Work
- Section D: Proof of Lemma 1 (Softmax stability)
- Section E: Additional Details of SFT Training Stage
- Section F: Additional Details of RFT Training Stage
- Section G: Additional Details for Evaluation
- Section H: Use and Intended Scope of Artifacts
- Section I: Case study and Visualization

B Use of LLMs

Large language models (LLMs) were employed to provide assistance with language refinement during the preparation of this paper. We emphasize that the LLMs were used only as writing aids; all conceptual contributions—including research problem formulation, methodological design, experimental analysis, and interpretation of findings—are the sole work of the authors.

C Related Work

C.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 (Xi et al., 2025; Xu et al., 2024), researchers have explored using pretrained LLMs or VLMs to generate plans from textual and visual observations, typically relying on carefully crafted prompts (Shin et al.; Rana et al.; Hu et al.; Kim et al.; Singh et al.; Fu et al., 2024) or auxiliary tools (Rana et al.; Ahn et al.; Silver et al.) to provide necessary planning cues. While simple and data-efficient, such methods often struggle with spatial grounding and temporal coherence in visually rich environments.

More advanced methods aim to improve planning performance through model fine-tuning.Several works have employed supervised fine-tuning pipelines (Wu et al., 2023; Chen et al.; Shi et al., 2025a), while others adopt preference optimization methods (Wang et al.; Song et al., 2024) such as Direct Preference Optimization (DPO) (Rafailov et al., 2023) to better align model behavior with expert planning preferences.

Reinforcement learning has recently emerged as an important post-training paradigm for foundation models, with a few works exploring its application to embodied task planning via online simulation, such as ERA (Chen et al., 2025a). However, online RL in embodied environments is hindered by high interaction cost and sparse rewards. In contrast, our work adopts an on-policy reinforcement fine-tuning framework with offline rewards, which avoids expensive environment interaction while retaining the benefits of reinforcement-based optimization.

C.2 RL for Foundation Models

Recent work has shown that reinforcement learning is an effective post-training paradigm for foundation models, with policy-based methods such as PPO and GRPO substantially improving reasoning and decision-making beyond supervised fine-tuning (Guo et al., 2025; Shao et al., 2024). This paradigm has been extended from text reasoning to multimodal domains (Meng et al., 2025) like visual question answering (Liu et al., 2025; Shen et al., 2025; Zhang et al., 2025a) and structured decision-making tasks (Zhao et al., 2025; Zhang et al., 2025b; Tan et al., 2025).

Beyond empirical gains, prior studies (Shenfeld et al., 2025; Chen et al., 2025b; Zhu et al., 2025) suggest that on-policy optimization is a key factor underlying these improvements, as it enables learning from self-generated mistakes and negative examples, leading to stronger generalization than imitation-based objectives. Related work has further explored the idea of on-policy distillation (Deng et al., 2024, 2025), drawing connections to DAgger-style learning (Ross et al., 2011), where expert trajectories provide supervision while policy optimization is performed through reinforcement learning rather than token-level imitation.

Building on these insights, we study on-policy reinforcement fine-tuning with expert guidance for embodied planning. We propose a simple prefix-based reward that compares model rollouts with expert trajectories and optimize the policy using GRPO (Shao et al., 2024).

D Proof of Lemma 1 (Softmax stability)

Lemma 1 (Softmax stability). Let $r_1, r_2 : \mathcal{Y} \to \mathbb{R}$ be bounded rewards on a finite (or countable) outcome space $\mathcal{Y}$, and assume a reference policy $\pi_{\text{ref}}(\cdot \mid x)$ with full support. Fix $\beta_{\text{eff}} > 0$ and define, for $i \in \{1, 2\}$,

$$\pi_i^\star(y \mid x) = \frac{\pi_{\text{ref}}(y \mid x) \exp\big(r_i(y)/\beta_{\text{eff}}\big)}{\sum_{y'} \pi_{\text{ref}}(y' \mid x) \exp\big(r_i(y')/\beta_{\text{eff}}\big)}.$$

If $\|r_1 - r_2\|_\infty \le \delta$, then

$$D_{\text{KL}}\big(\pi_1^\star \,\|\, \pi_2^\star\big) \le \frac{2\delta}{\beta_{\text{eff}}}, D_{\text{KL}}\big(\pi_2^\star \,\|\, \pi_1^\star\big) \le \frac{2\delta}{\beta_{\text{eff}}}.$$

Proof. Fix x and omit it from notation. Introduce the "potentials"

$$\begin{aligned} f(y) &\coloneqq \log \pi_{\text{ref}}(y) + \frac{r_1(y)}{\beta_{\text{eff}}}, \\ g(y) &\coloneqq \log \pi_{\text{ref}}(y) + \frac{r_2(y)}{\beta_{\text{eff}}}. \end{aligned} \tag{13}$$

and the partition functions $Z_f = \sum_y \exp(f(y))$, $Z_g = \sum_y \exp(g(y))$. Then $\pi_1^\star(y) = \exp(f(y))/Z_f$ and $\pi_2^\star(y) = \exp(g(y))/Z_g$. We will bound $D_{\text{KL}}(\pi_1^\star \| \pi_2^\star)$; the reverse bound follows by symmetry.

Step 1: KL in terms of potentials. By definition,

$$\begin{aligned} D_{\text{KL}}\big(\pi_1^\star \| \pi_2^\star\big) &= \sum_y \pi_1^\star(y) \, \log \frac{\pi_1^\star(y)}{\pi_2^\star(y)} \\ &= \sum_y \pi_1^\star(y) \, \big(f(y) - g(y)\big) + \log \frac{Z_g}{Z_f} \\ &= \mathbb{E}_{y \sim \pi_1^\star}\big[f(y) - g(y)\big] + \log \frac{Z_g}{Z_f}. \end{aligned}$$

Step 2: Bounding the expectation term. Since $\|r_1 - r_2\|_\infty \le \delta$, we have

$$\|f - g\|_\infty = \left\| \tfrac{r_1 - r_2}{\beta_{\text{eff}}} \right\|_\infty \le \frac{\delta}{\beta_{\text{eff}}}.$$

Thus

$$\mathbb{E}_{\pi_1^\star}[f - g] \;\le\; \|f - g\|_\infty \;\le\; \frac{\delta}{\beta_{\text{eff}}}.$$

Step 3: Lipschitzness of log-sum-exp. For any vectors a, b on $\mathcal{Y}$,

$$\log \sum_y e^{b(y)} - \log \sum_y e^{a(y)} \;\le\; \max_y \big(b(y) - a(y)\big),$$

which follows from $\sum_y e^{b(y)} = \sum_y e^{a(y)} e^{b(y)-a(y)} \le e^{\max(b-a)} \sum_y e^{a(y)}$. By swapping a, b one also has $\log \sum_y e^{a(y)} - \log \sum_y e^{b(y)} \le \max_y \big(a(y) - b(y)\big)$. Hence

$$\big|\log Z_g - \log Z_f\big| \;\le\; \|g - f\|_\infty \;\le\; \frac{\delta}{\beta_{\text{eff}}}.$$

Step 4: Combine the bounds. Putting Steps 2–3 into Step 1 yields

$$D_{\text{KL}}\big(\pi_1^\star \| \pi_2^\star\big) \;\le\; \frac{\delta}{\beta_{\text{eff}}} + \frac{\delta}{\beta_{\text{eff}}} = \frac{2\delta}{\beta_{\text{eff}}}.$$

Exchanging the roles of (r_1, f, Z_f) and (r_2, g, Z_g) gives $D_{\text{KL}}(\pi_2^\star \| \pi_1^\star) \le 2\delta/\beta_{\text{eff}}$ as well. □

E Additional Details of SFT Training Stage

E.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 teacher model. We collect response traces on a subset of EB-ALFRED tasks from the EmbodiedBench benchmark and retain only trajectories that successfully complete the task. For each task, we record the full model output, including both intermediate reasoning and the executable action plan.

Following the embodied planning setup, each training instance conditions on the current state and requires the model to predict all remaining actions. Concretely, for an expert trajectory of length k, we decompose it into k training samples, where each sample uses the current observation and action history as input and targets the remaining action sequence. The final dataset consists of 8839 training samples, each containing a natural language instruction, egocentric visual observations, and a structured model response. An example training instance is shown in Figure E.1.

E.2 Training Hyperparameters

We perform full-parameter supervised fine-tuning on the Qwen2.5-VL-7B model. The training is conducted on 4 NVIDIA H200 GPUs for approximately 1~2 hours. All hyperparameters are summarized in Table 5.

SFT dataset example


```
{
 "messages": [
   {
     "role": "user",
     "content": "{EB-ALFRED prompt}"
   },
   {
     "role": "assistant",
     "content": [
       {
         "from": "gpt",
         "value": "<|think_start|>visual_description: The ladle is in the sink along
    with other utensils. The faucet is visible above the sink. reasoning_and_reflection:
     The previous actions successfully placed the ladle in the sink. The next step is to
     turn on the faucet to rinse the ladle. The faucet is visible, so we can proceed
    with turning it on. After rinsing, we will need to turn off the faucet, pick up the
    ladle, and move it to the dining table. language_plan: 1. Turn on the faucet to
    rinse the ladlenewline2. Turn off the faucet after rinsingnewline3. Pick up the
    ladle from the sinknewline4. Find the dining tablenewline5. Place the clean ladle on
     the dining table<|think_end|><|action_start|>[[2, 'find a Faucet'], [155, 'turn on
    the Faucet'], [156, 'turn off the Faucet'], [64, 'find a Ladle'], [109, 'pick up the
     Ladle'], [18, 'find a DiningTable'], [133, 'put down the object in hand']]<|action_
    end|>"
       }
 ],
 "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}\"}]}]"
 }
},
```


E.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.

F Additional Details of RFT

F.1 Training Dataset Details

We construct our reinforcement fine-tuning (RFT) dataset based on the original ALFRED benchmark, following the decomposition and formatting strategy described in Section 2. Notably, we do not reuse the SFT-distilled dataset for RFT. The SFT dataset adopts an instruction format that exactly matches the evaluation prompts in EB-ALFRED, whereas the RFT dataset uses a different instruction formulation to reduce prompt-level coupling and encourage policy generalization.

In addition, the two datasets originate from different environments: the SFT data is distilled from EB-ALFRED, while the RFT data is constructed

Component	Setting	Component	Setting
	Model Configuration		
model	Qwen2.5-VL-7B-Instruct	model_max_length	8192
freeze_visual_encoder	true	freeze_language_model	false
freeze_multimodal_projector	true	precision	bf16
deepspeed config	ZeRO-3	gradient_checkpointing	true
	Training Configuration		
finetuning_type	full	num_train_epochs	1
learning_rate	1e-5	per_device_batch_size	8
grad_accum_steps	1	lr_scheduler	cosine
warmup_ratio	0.05	tf32	true
num_gpus	4		

Table 5: Detailed hyperparameters used in supervised fine-tuning.

Figure 4: Summary of SFT training results.

Metric	Value
Epochs	1.0
Total steps	277
Total FLOPs	1.89e14
Training Loss	0.379
Runtime (s)	3618.9743
Samples/sec	2.442
Steps/sec	0.077

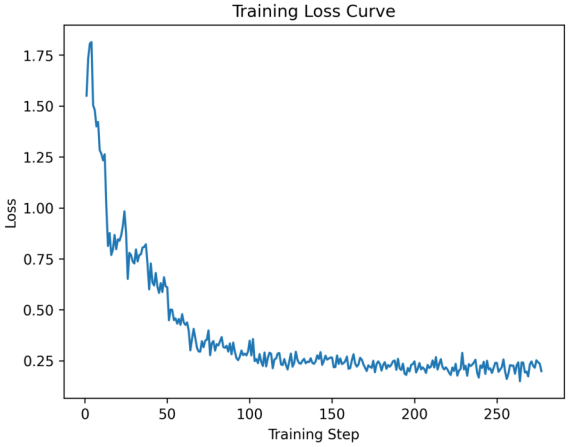


Figure 5: Training loss curve during SFT stage.

directly from ALFRED. Since EB-ALFRED is derived from ALFRED with additional processing, the two differ in action space definitions, action naming conventions, and instruction phrasing, although they share the same underlying environment distribution.

Consequently, EB-ALFRED evaluation spans multiple task variants, including *Base*, *Common Sense*, *Complex* , and others. where the latter variants are created via instruction-level augmentation over the Base tasks. In contrast, our RFT dataset contains only Base-type tasks, ensuring that improvements on augmented evaluation settings reflect generalization induced by reinforcement fine-tuning rather than exposure to augmented instructions during training.

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

F.2 Format reward

To encourage valid and interpretable plans, we design a structured format reward inspired by EmbodiedBench (Yang et al., 2025), 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:

$$R_{\text{format}} = R_{\text{structure}} + R_{\text{valid}} + R_{\text{match}}, \quad (14)$$

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.

F.3 Training Hyperparameters

We implement reinforcement fine-tuning using the `OpenRLHF` (Hu et al., 2024) framework, adopting the Generalized Reinforced Preference Optimization (GRPO) algorithm (Shao et al., 2024) to optimize policy learning from structured reward feedback. A full list of training hyperparameters is provided in Table 6.

F.4 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.

All experiments are conducted on 8 H200 GPUs. We select the checkpoint at 450 training steps as the final model, with the entire reinforcement fine-tuning process taking approximately 8–10 hours to complete.

G Additional Details for Evaluation

G.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 (Shridhar et al., 2020) and implemented on top of the AI2-THOR simulator (Kolve et al., 2017). It supports eight core skill types such as *pick up*, *put down*, *find*, *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 (Savva et al., 2019), 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, EmbodiedBench introduces six distinct task subsets. Due to space limitations, we omit the subset *Long Horizon* from the main table. Performance on long-horizon tasks remains challenging for our method, and we leave further improvements on this subset to future work.

- **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.

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 6: Hyperparameter configuration used during reinforcement fine-tuning.

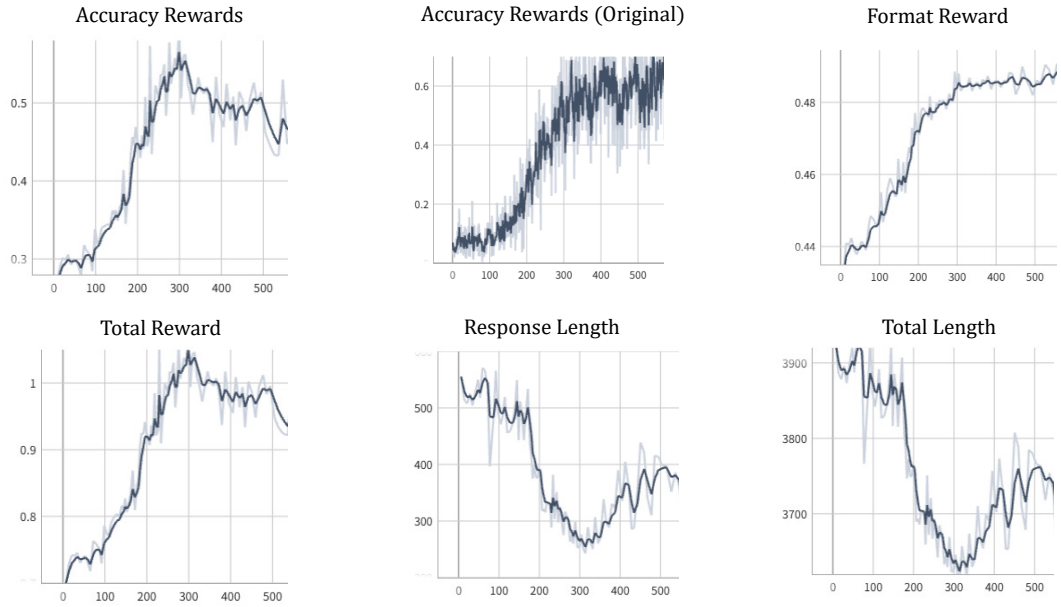


Figure 6: Training curve during reinforcement fine-tuning. The figure shows the progression of total reward, filtered and unfiltered accuracy reward, and generation length statistics.

- **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 7, these categories span a wide range of reasoning challenges. Notably, since our reinforcement fine-tuning 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, EmbodiedBench provides a rigorous, scalable, and diagnostic framework for benchmarking embodied agents across diverse real-world chal-

Table 7: 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.

lenges. 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.

G.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 (Anthropic), Gemini-2.0-flash (DeepMind), GPT-4o (OpenAI, b), and GPT-4o-mini (OpenAI, a).

(2) *Open-source general VLMs*: we evaluate widely adopted open-source VLMs trained for generic multimodal tasks, such as LLaMA-3.2-Vision-11B (Meta), Qwen2.5-VL-7B (Team, 2025) and InternVL2.5-8B (Chen et al., 2024).

(3) *Training-based embodied VLMs*. We further compare our method with several recent training-based approaches specifically designed for embodied task planning. This category includes WAP-7B (Wang et al.), which is trained via extensive data augmentation under a pure supervised fine-tuning (SFT) paradigm, as well as VAGEN-7B (Wang et al., 2025) and ERA-7B (Chen et al., 2025a), both of which adopt online reinforcement learning to optimize embodied decision-making policies through interactive simulator feedback. These models represent the current state of training-based embodied VLMs, spanning both SFT-only and online-RL-based optimization strategies, and serve as strong baselines for evaluating the effectiveness and efficiency of our offline reward–driven reinforcement fine-tuning approach.

Run ID	Base	Com	Cplx	Visual	Spatial	Avg
Run 1	86	62	82	72	60	72.4
Run 2	88	60	84	74	60	73.2
Run 3	88	64	82	72	62	73.6
Run 4	88	62	80	71	58	71.8
Run 5	86	64	82	72	64	73.6

Table 8: Our method's raw results (%) across 5 independent runs in EB-ALFRED with different random seeds.

Method	Avg	Base	Com	Cplx	Visual	Spatial
Ours	**72.92 ± 0.71**	87.2 ± 0.98	63.2 ± 1.60	82.0 ± 1.26	72.2 ± 0.98	60.8 ± 2.04

Table 9: Statistical summary over 5 independent runs (mean ± standard deviation, %) in EB-ALFRED.

G.3 Multi-Seed Variance Analysis

We additionally evaluate the statistical robustness of our final method (**SFT+RFT**) on the **EB-ALFRED** benchmark using **5 independent training runs** with different random seeds. Since EmbodiedBench is relatively deterministic compared with highly stochastic RL environments, we expect naturally low variance. Table 8 reports the raw results of all runs, and Table 9 summarizes the mean and standard deviation. The consistently low variance across all splits confirms that the performance gains of our method are stable across seeds.

G.4 Action Chunking Ablation

Our method predicts and optimizes an *action chunk* during training: the model outputs multiple future actions at once, and the offline reward is computed sequentially over the chunk. At evaluation time, we follow the same chunk-based setup: the model

Method	Training	Evaluation	Avg	Base	Com	Cplx	Visual	Spatial
SFT only	one-step	one-step	49.6	54	44	52	58	40
SFT only	multi-step	multi-step	57.3	62.7	48	62	62	52
SFT only	multi-step	one-step	52.0	58	46	58	56	42
SFT+RFT	one-step	one-step	69.68	76.4	64	80	72	56
SFT+RFT	multi-step	multi-step	72.4	86	62	82	72	60
SFT+RFT	multi-step	one-step	70.0	82	60	82	72	54

Table 10: One-step vs. multi-step training/evaluation. Multi-step prediction helps, but does not fully explain the gains from RFT.

Method	Training	Evaluation	Avg	Base	Com	Cplx	Visual	Spatial
RFT+SFT	one-step	one-step	55.6	64	46	66	56	46
RFT+SFT	multi-step	multi-step	59.2	72	42	68	66	48

Table 11: Reversed stage-order ablation (RFT→SFT).

predicts one chunk, executes it until task completion or abnormal termination, and then predicts the next chunk if needed.

To examine whether the performance gain mainly comes from chunk-based prediction rather than the proposed RFT objective, we conduct a controlled ablation on **one-step** versus **multi-step**. Here, **one-step** predicts only the next action at each turn, while **multi-step** predicts the remaining action sequence as an action chunk.

As shown in Table 10, multi-step consistently outperforms one-step. This benefit is more pronounced under SFT (49.6 → 57.3 Avg), suggesting that chunking helps maintain short-horizon action coherence and reduces forgetting of the immediately previous action. After RFT, the gain remains positive but smaller (69.68 → 72.4 Avg). We also evaluate a practical hybrid setting that trains with multi-step chunks but executes only the first action at test time; this remains better than pure one-step, though slightly below full multi-step execution.

However, chunking does not explain the full improvement. With SFT alone, multi-step reaches 57.3, while adding RFT further improves result to 72.4. Conversely, even in the strict one-step setting, RFT still yields a large gain over SFT (49.6 → 69.68), showing that the offline RFT objective contributes substantial benefits beyond chunking.

Chunking is also closely coupled with our offline reward formulation. Early exploration often produces highly variable chunks, yet the offline prefix reward remains tolerant in this regime, whereas online RL tends to become less stable when chunk-level rollouts drift substantially. We observe the same trend in the reversed stage-order ablation (RFT→SFT): multi-step still helps, but cannot replace the benefit of RFT, as shown in Table 11.

H Use and Intended Scope of Artifacts

All third-party datasets, benchmarks, and pre-trained models used in this work are employed in accordance with their original licenses and stated terms of use. Our usage is consistent with their intended purposes, namely academic research and benchmarking in embodied AI.

The artifacts introduced in this work, including derived training data and fine-tuned models, are intended solely for research use and are compatible with the access conditions of the original datasets. We plan to release the processed data, code, and model checkpoints after the review process, in compliance with the original licenses.

I Case study and Visualization

I.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.

I.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 EmbodiedBench. Our SFT stage also adopts this prompt format.

EB-Habitat Prompt. This prompt format is used in EmbodiedBench'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.

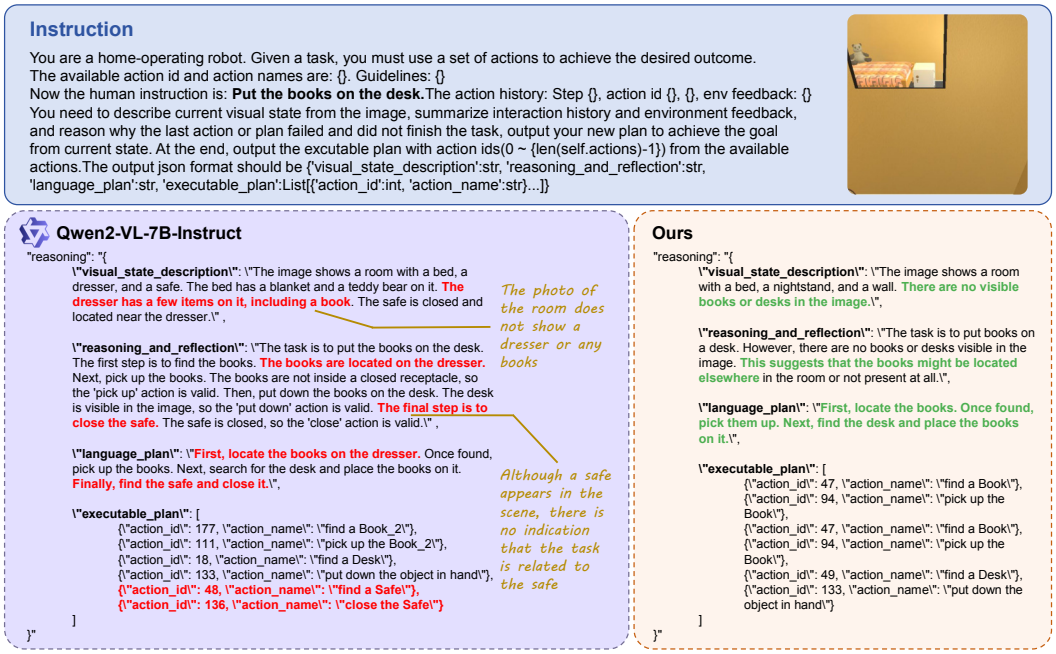


Figure 7: Case comparison (Our model vs Base model) in EB-ALFRED

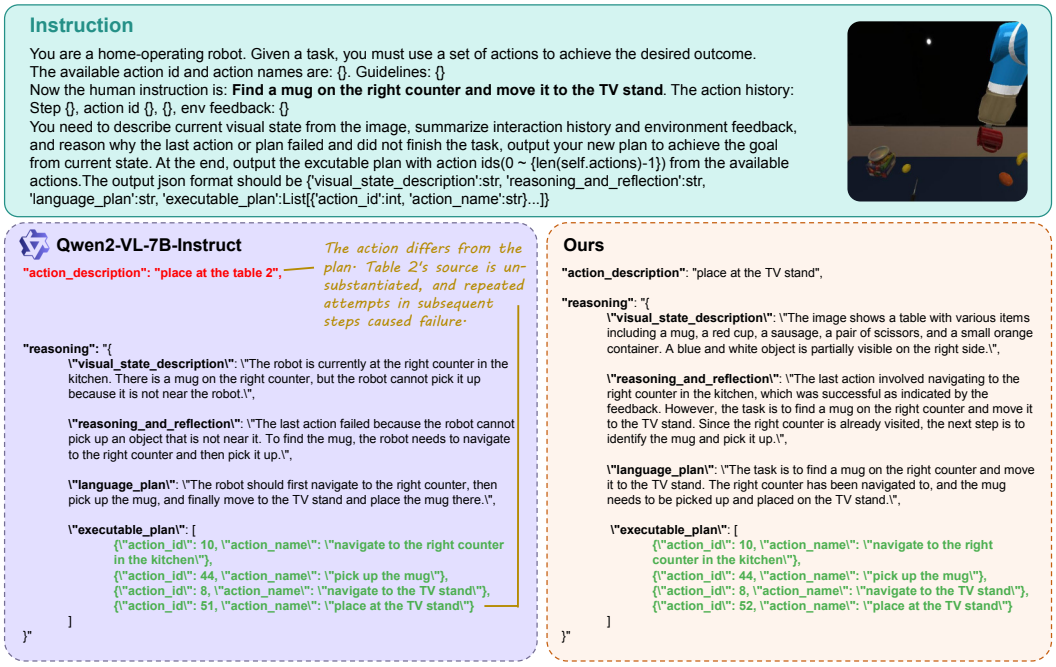


Figure 8: Case comparison (Our model vs Base model) in EB-Habitat

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.

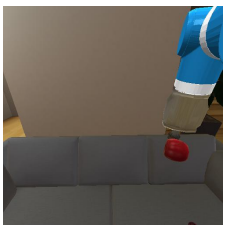
"reasoning": "{"visual_state_description": "The image shows a living room with a gray sofa, a coffee table, a staircase, and a blue object resembling a person. The apple is not visible in the image.", "reasoning_and_reflection": "The task is to retrieve an apple and place it on the sofa. However, the 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"

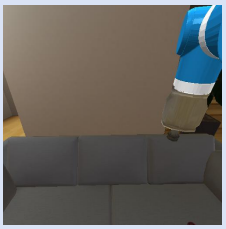
Action 2

"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"



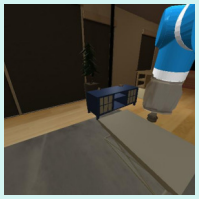
Action 4

"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,