World Modeling Makes a Better Planner: DUAL PREFERENCE OPTIMIZATION FOR EMBODIED TASK PLANNING

Siyin Wang[1,2] **Zhaoye Fei**[1] **Qinyuan Cheng**[1] **Shiduo Zhang**[1]
Panpan Cai[2,4] **Jinlan Fu**[3] * **Xipeng Qiu**[1,2] *

[1]Fudan University [2]Shanghai Innovation Institute
[3]National University of Singapore [4]Shanghai Jiao Tong University

ABSTRACT

Recent advances in large vision-language models (LVLMs) have shown promise for embodied task planning, yet they struggle with fundamental challenges like dependency constraints and efficiency. Existing approaches either solely optimize action selection or leverage world models during inference, overlooking the benefits of learning to model the world as a way to enhance planning capabilities. We propose Dual Preference Optimization (D²PO), a new learning framework that jointly optimizes state prediction and action selection through preference learning, enabling LVLMs to understand environment dynamics for better planning. To automatically collect trajectories and stepwise preference data without human annotation, we introduce a tree search mechanism for extensive exploration via trial-and-error. Extensive experiments on VoTa-Bench demonstrate that our D²PO-based method significantly outperforms existing methods and GPT-4o when applied to Qwen2-VL (7B), LLaVA-1.6 (7B), and LLaMA-3.2 (11B), achieving superior task success rates with more efficient execution paths.

1 INTRODUCTION

Embodied task planning Singh et al. (2022); Inoue & Ohashi (2022); Mai et al. (2023), which enables AI systems to perform real-world tasks through physical interaction, demands both correctness and efficiency. Incorrect or inefficient task planning not only wastes computational resources but may also lead to unsafe operations, compromising system usability and reliability in dynamic environments. Previous LLM-based approaches rely heavily on environment metadata Yao et al. (2022); Sun et al. (2023) or external object detection models Singh et al. (2022); Song et al. (2022), limiting their ability to operate end-to-end in real-world scenarios. Recent advances in Large Vision-Language Models (LVLMs) OpenAI (2024) have opened new possibilities for embodied intelligence, yet state-of-the-art LVLMs still struggle with fundamental issues such as dependency constraints (placing objects before picking them up) and inefficient planning (repeating unnecessary steps). These limitations stem from a critical gap: LVLMs operate on static snapshots of the environment, lacking the ability to model the dynamic nature of physical interactions.

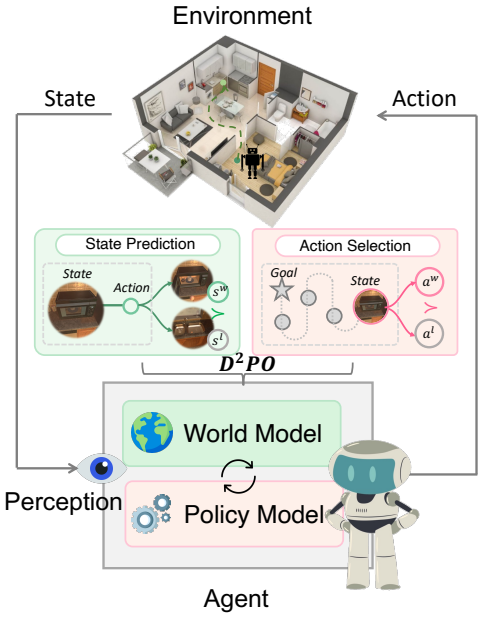


Figure 1: Overview of D²PO: World modeling enables better embodied task planning through joint preference optimization of state prediction and action selection.

*Corresponding authors.

Existing approaches leverage language models for embodied task planning, including prompt-based methods Song et al. (2022); Shin et al. (2024); Liang et al. (2022), supervised fine-tuning (SFT) from expert demonstrations Wu et al. (2023); Chen et al. (2024b); Jin et al. (2023), and RL-based optimization Carta et al. (2023); Yang et al. (2023); Szot et al. (2023). However, these methods primarily focus on learning direct mappings from state to action, optimizing for *what to do* without considering the consequences of these actions. To model environment dynamics, some recent methods leverage LLMs directly as world models through prompting Hao et al. (2023); Zhou et al. (2024) to guide the search path. However, these approaches introduce additional computational overhead while fail to develop world modeling capabilities during training.

Inspired by human cognition, where we naturally develop an *internal world model* through continuous interaction with the environment Johnson-Laird (1983); Tolman (1948); LeCun (2022), we propose to enhance planning capabilities by incorporating world modeling during the training stage. Accordingly, we propose a novel learning framework called **Dual Preference Optimization (D^2PO)**, which jointly optimizes state prediction and action selection through preference learning: (1) State Prediction, where the model predicts the next state given the current state and action, learning the consequences of actions over time; (2) Action Selection, which improves the model's policy ability to choose appropriate actions with reasoning based on the goal and interaction history. By representing world dynamics in natural language, we leverage the prior knowledge of large language models. More importantly, rather than treating world modeling as a separate component, our framework uses world modeling objectives to enhance the policy's planning capabilities. Through this dual optimization, the policy model naturally develops an understanding of world dynamics, leading to more informed action selection without requiring explicit world model guidance during inference.

To automatically collect correct trajectories and stepwise preference data for training, we introduce a tree search mechanism that systematically explores action sequences within a simulated environment. By combining model evaluations and environmental feedback, this scalable method can automatically generate extensive trajectories and construct preference pairs for both action selection and state prediction. This approach eliminates the need for expert demonstrations and preference annotations, while efficiently gathering diverse embodied interaction experiences.

Extensive experiments on VoTa-Bench, our vision-enhanced extension of the text-only LoTa-Bench (designed for LLMs) Choi et al. (2024), demonstrate that our method outperforms existing training approaches across multiple evaluation settings. Our evaluation shows significant improvements in both success rate and planning efficiency, with our 7B-parameter model surpassing GPT-4o's performance on multiple test types, highlighting the efficacy and potential of our approach.

Our main contributions are as follows:

- We propose to learn world modeling to enhance model's planning abilities through our novel Dual Preference Optimization (D^2PO) framework, which jointly optimizes state prediction and action selection through preference learning, enabling the model to learn action consequences while improving planning.
- We introduce a tree search algorithm that automatically collects trajectories and constructs multimodal stepwise preference data for embodied task planning via trial-and-error, eliminating the need for human annotation.
- We demonstrate that auxiliary world modeling objectives significantly improve embodied task planning with extensive experiments on VoTa-Bench. Our 7B-parameter model achieves a relative improvement of 31.4% and 33.0% in success rate and planning efficiency respectively compared to SFT baselines.

2 Relate Work

2.1 Embodied Task Planning

Embodied task planning is a key component of Embodied AI, enabling agents to perform complex tasks within dynamic and physical environments. Early LLM-based methods Yao et al. (2022); Sun et al. (2023); Zhao et al. (2023) rely purely on textual metadata from the environment, making them struggle to adapt to the unpredictable and dynamic nature of real-world settings. Later approaches Singh et al. (2022); Song et al. (2022); Shin et al. (2024); Yang et al. (2024); Zhao et al. (2024);

Shirai et al. (2023) introduce cascaded visual processing through external models. However, these multi-stage pipelines increase system complexity and potential error propagation. Notably, existing methods Pashevich et al. (2021); Inoue & Ohashi (2022); Lu et al. (2023); Chen et al. (2024b); Zhao et al. (2024) also heavily rely on manual step-by-step instructions. In contrast, we propose an end-to-end approach using a single VLM for both direct visual processing and autonomous planning, despite the increased modeling challenges.

Methodologically, several recent works have explored diverse prompting strategies Song et al. (2022); Shin et al. (2024); Liang et al. (2022) and multi-agent frameworks with specialized roles Zhang et al. (2023); Mai et al. (2023); Wang et al. (2024b). SFT-based approaches learn from expert demonstrations using human or language model annotated data Wu et al. (2023); Chen et al. (2024b); Jin et al. (2023), or collect training data through actor-critic simulation Li et al. (2024). Recent works explore PPO-based optimization using designed reward templates Carta et al. (2023) or optimizing through environment interaction feasibility Yang et al. (2023); Szot et al. (2023) These RL-based methods require designed reward or training separate reward models. Direct preference optimization (DPO) Rafailov et al. (2023), as an implicit reward modeling approach, has shown promise in LLM planning Song et al. (2024); Zhao et al. (2024). Different from existing approaches focusing on optimizing action selection alone, we propose to leverage DPO for joint optimization of state prediction and action selection in LVLMs.

2.2 World Model

World model is a computational framework that predicts future states based on current states and actions, enabling decision-making through simulated outcomes Sutton (1990). Traditional approaches based on recurrent state space models (RSSM) for low-level control, focus on learning state transitions in a latent space rather than language modeling and rely on handcrafted reward functions Hafner et al. (2019; 2020); Wu et al. (2022); Hafner et al. (2023). Recent advancements have explored integrating LLMs to leverage prior knowledge, with some using LLMs to generate symbolic plans or code to modeling world Guan et al. (2023); Dainese et al. (2024), and others using text prompting Hao et al. (2023); Zhou et al. (2024). However, these methods mainly utilize world modeling during inference, without incorporating it into the training process. In contrast, our approach jointly optimizes state prediction and action selection with DPO during training stage, learning world modeling capabilities that enhance the model's planning abilities.

3 Method

3.1 Task Formulation

We model the embodied task planning problem as a Partially Observable Markov Decision Process (POMDP), where the agent operates in a partially observable environment and generates actions based on multimodal feedback. The POMDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{M}, \mathcal{R}, \gamma)$, where S is the state space, $\mathcal{A}$ is the action space, O is the observation space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the transition function ($s_t = \mathcal{T}(s_{t-1}, a_t)$), $\mathcal{M} : \mathcal{S} \rightarrow \mathcal{O}$ is the observation function provided by the simulation environment, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ is the reward function, and γ is the constant discount factor. Due to partial observability, the agent cannot directly access the complete state $s_t \in \mathcal{S}$, but instead receives first-person visual observations $o_t = \mathcal{M}(s_t) \in \mathcal{O}$ from the environment.

Given a task goal $g \in \mathcal{G}$, where $\mathcal{G}$ is the space of natural language task instructions, the agent interacts with the environment through a sequential planing process. At each time step t, the agent receives an observation $o_t \in \mathcal{O}$ from the simulation environment and maintains a history of past observations and actions $h_t = (o_0, a_1, o_1, ..., a_t, o_t)$. Based on this history and the task goal, the agent's policy π_θ generates an action $a_{t+1} \sim \pi_\theta(\cdot|g, h_t)$, where the policy $\pi_\theta : \mathcal{G} \times \mathcal{H} \rightarrow \mathcal{A}$ maps the current history h_t and goal g to a distribution over the action space $\mathcal{A}$.

Through this interaction process, a trajectory is formed as $e = (g, o_0, a_1, o_1, ..., o_{n-1}, a_n, o_n)$, where n is the length of the trajectory, and each observation o_t is provided by the environment after executing action a_t. The task is considered successfully completed if the final state satisfies the goal condition, with the reward defined as $r(e) = 1$ if the goal condition is satisfied and 0 otherwise.

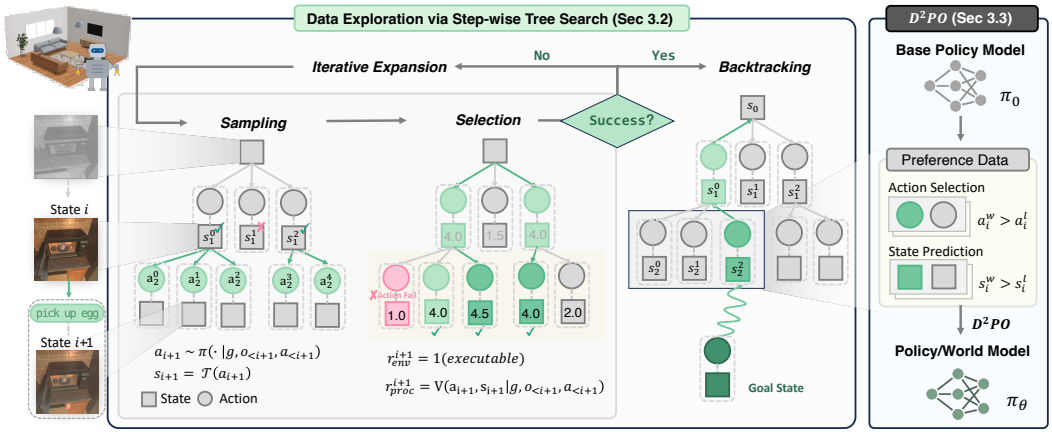


Figure 2: Our method consists of two dimensions: (1) Data Exploration via Step-wise Tree Search (Sec 3.2), which collects preference data through sampling and selecting potential actions, iterative tree expansion, and trajectory backtracking; (2) Dual Preference Optimization (D^2PO) framework (Sec 3.3) that leverages the collected preference pairs to jointly optimize action selection and state prediction.

3.2 Data Exploration via Step-wise Tree Search

Previous training methods often rely on costly human expert annotations or GPT-4o-generated labels, which can be both time-consuming and limited in diversity. To address these challenges, we introduce a novel tree search framework for embodied task planning that explores the action space step-by-step with environment interaction, eliminating the need for human expert annotation.

Our framework consists of three components: action sampling and evaluation, iterative tree expansion, and trajectory validation and backtracking. First, we sample and evaluate potential actions at each state using a hybrid scoring mechanism. Then, we iteratively expand the search tree by selecting and exploring promising nodes at each level, following a breadth-first strategy. Once a goal state is reached, we backtrack through the trajectory to create preference pairs for dual optimization of action selection and state prediction. More detailed implementation is provided in the appendix D.

Action Sampling and Evaluation At each selected state node s_t, we sample multiple potential actions $a_t^{(i)}{}_{i=1\ldots K}$ using a base policy model. Actions are evaluated through a hybrid scoring mechanism combining two components: a process reward score $r_{\text{proc}}^{(i)}$ from GPT-4o, which evaluates how actions contribute to goal completion based on the history according to a score-based prompt, and a binary environmental feasibility score $r_{\text{env}}^{(i)}$ indicating action executability (1 if executable, 0 if not). These scores are normalized and combined with equal weights into $r_{\text{total}}^{(i)} = \alpha r_{\text{proc}}^{(i)} + (1-\alpha) r_{\text{env}}^{(i)}$ where $\alpha = 0.5$, guiding exploration towards both goal-oriented and executable trajectories.

Iterative Tree Expansion Following a breadth-first strategy, actions with high scores $r_{\text{total}}^{(i)} \geq \tau$ (where τ is a predefined threshold) are selected for expansion. The states after selected actions execution in the environment form the next level of exploration. This step-by-step expansion ensures extensive exploration of promising solution paths at each depth while maintaining physical feasibility.

Trajectory Validation and Backtracking Upon reaching a goal state, we extract the trajectory by backtracking and constructing preference pairs for both action selection and state prediction. At each step $s_{t-1} \rightarrow a_t$ in a successful trajectory, where visual observations $o_{t-1} = \mathcal{M}(s_{t-1})$ represent the agent's first-person view of states as input, we generate two types of preference pairs. For action selection, we obtain $(g, a_{<t}, o_{<t}, r_t^w, a_t^w, r_t^j, a_{t\,j\in\mathcal{N}(t)}^j)$, where (r_t^w, a_t^w) represents the chosen reasoning-action pair and $r_t^j, a_{t\,j\in\mathcal{N}(t)}^j$ are alternatives from sibling nodes. For state prediction, we

extract $(s_{t-1}, a_t, s_t^w, s_{t\,j\in\mathcal{N}(t)}^j)$, where s_t^w represents the state description that would result from executing action a_t^w, and $s_{t\,j\in\mathcal{N}(t)}^j$ are the corresponding state descriptions from alternative actions.

3.3 Dual Preference Optimization (D^2PO) Framework

We propose the Dual Preference Optimization (D^2PO) framework, building upon Direct Preference Optimization (DPO) Rafailov et al. (2023). The core idea of DPO is to directly optimize the model using preference pairs $\{y^w, y^l\}$, where the optimization objective encourages the model to assign a higher probability to preferred responses $p(y^w \succ y^l)$ while maintaining proximity to a reference model, without additional reward model.

We extend this preference learning framework to embodied task planning by simultaneously optimizing two critical aspects: **action selection** and **state prediction**. The action selection optimization focuses on enhancing the policy model, enabling the agent to choose the most appropriate action based on the current state, history, and task instruction. Meanwhile, the state prediction optimization targets the world modeling, which learns to predict the next state resulting from the current state and action. This dual optimization approach enhances the agent's understanding of environment dynamics, leading to better planning performance.

Action Selection Given context $(g, a_{<t}, o_{<t})$, we optimize the probability of selecting preferred reasoning-action pairs (r_t^w, a_t^w) over rejected pairs (r_t^l, a_t^l):

$$L_{\text{action}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(g,a_{<t},o_{<t},r_t^w,a_t^w,r_t^l,a_t^l)\sim\mathcal{D}}\Big[\log\sigma\Big(\beta\log\frac{\pi_\theta(r_t^w,a_t^w|g,a_{<t},o_{<t})}{\pi_{\text{ref}}(r_t^w,a_t^w|g,a_{<t},o_{<t})} - \beta\log\frac{\pi_\theta(r_t^l,a_t^l|g,a_{<t},o_{<t})}{\pi_{\text{ref}}(r_t^l,a_t^l|g,a_{<t},o_{<t})}\Big)\Big]. \tag{1}$$

State Prediction Given state-action pairs (s_{t-1}, a_t), we optimize the prediction of preferred outcome states s_t^w after executing action a_t over rejected states s_t^l. The states are represented as descriptions that capture key object properties, spatial relationships, and agent status (e.g., "the plate is on the table, and the agent is holding the cup"). This optimization enables the model to learn the dynamic state changes induced by actions. Formally, the state prediction objective is:

$$L_{\text{state}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(a_t,s_{t-1},s_t^w,s_t^l)\sim\mathcal{D}}\Big[\log\sigma\Big(\beta\log\frac{\pi_\theta(s_t^w|s_{t-1},a_t)}{\pi_{\text{ref}}(s_t^w|s_{t-1},a_t)} - \beta\log\frac{\pi_\theta(s_t^l|s_{t-1},a_t)}{\pi_{\text{ref}}(s_t^l|s_{t-1},a_t)}\Big)\Big]. \tag{2}$$

Finally, we combine both objectives in a joint optimization problem. The total loss is a weighted sum of the action selection and state prediction losses, with the objective function defined as:

$$\mathcal{L}_{\text{total}} = L_{\text{action}}(\pi_\theta; \pi_{\text{ref}}) + \lambda L_{\text{state}}(\pi_\theta; \pi_{\text{ref}}),$$

where λ is a hyperparameter controlling the balance between the two optimization objectives.

4 Experiment

4.1 Experimental Settings

4.1.1 VoTa-Bench

Dataset Our evaluation is based on the LoTa-Bench Choi et al. (2024), which leverages the AI2-THOR Kolve et al. (2017) simulation environment and repurposes data from ALFRED Shridhar et al. (2019). Unlike ALFRED, which provides both task- and step-level instructions for translating detailed step-by-step guidance into robot actions, LoTa-Bench focuses on high-level task planning using only task-level instructions.

In this work, we extend LoTa-Bench to create a new multimodal benchmark, VoTa-Bench, to better support LVLMs. (1) Unlike the original LoTa-Bench, which relies on textual descriptions, VoTa-Bench incorporates egocentric visual information as both the initial state and the observation after

each operation, requiring the model to effectively process visual inputs. (2) For evaluation, we do not rely on executable skills and logits computation; instead, we adopt an open-domain generation approach, which may result in the model generating non-executable skills. (3) The original dataset's environments were same to the training environment (seen scene). We expanded the dataset by adding new unseen environments to test the model's generalization, resulting in 549 seen test samples and 646 unseen test samples, covering 108 objects and 120 scenes. More details about VoTa-Bench are in Appendix C.

4.1.2 BASELINES

Our evaluation includes the zero-shot performance of several leading LVLMs, such as GPT-4o, GPT-4o-mini OpenAI (2024), Gemini-1.5-Pro Team (2024), Qwen2-VL-72B Wang et al. (2024a) and LLaVA-1.6-34B Liu et al. (2024).

Additionally, we validate our approach on Qwen2-VL-7B Wang et al. (2024a), LLaVA-1.6-7B Liu et al. (2024), and Llama-3.2-Vision-11B Meta (2024). The compared methods are as follows: **(1) In-Context Learning:** We provide 5-shot examples to prompt the model for generation. **(2) SFT:** We fine-tune the models using our collected dataset. **(3) DPO:** We optimize the models using our collected action selection data. Notably, the DPO data is collected by us and focuses solely on action selection optimization, serving as an ablation of our D^2PO method. **(4) D^2PO (Ours):** We propose a dual preference optimization approach, leveraging both action selection and state prediction data for enhanced performance.

4.1.3 EVALUATION METRICS

Success Rate (SR) The Success Rate (SR) measures task completion by verifying if the final state of the environment, including object states and positions, satisfies the task's goal conditions. For example, in the task "Place a cold apple on the dinner table," success is achieved only if the apple is chilled and located on the dinner table.

Path-Length Weighted Success Rate (PL) We introduce the Path-Length Weighted Success Rate (PL) Shridhar et al. (2019) to evaluate efficiency, which adjusts SR by comparing the model's step sequence length to the expert demonstration. The PL score is calculated as:

$$\text{PL} = \text{SR} \times \frac{L^*}{\max(L^*, \hat{L})},$$

where L^* is the expert's trajectory length, and $\hat{L}$ is the model's trajectory length. This penalizes models that take longer than the expert, ensuring both task success and efficiency are considered. For instance, a model takes twice as long as the expert receives half the credit.

4.1.4 IMPLEMENTATION DETAILS

For the models Qwen2-VL-7B, LLaVA-1.6-7B, and Llama-3.2-Vision-11B, we adopt the same training protocol. We use full-parameter tuning, first performing SFT for 3 epochs, using a learning rate of $3e{-}5$ and a batch size of 32. Following SFT, we conduct D^2PO for 1 epoch, with a learning rate of $5e{-}7$ and a batch size of 32. In the D^2PO loss function, we set the balancing parameter $\lambda = 1$ to equally weigh the contributions of action selection and state prediction. The DPO implementation is kept identical to the D^2PO setup. Our training data consists of 4.5k SFT samples and 15k DPO samples. Due to the inherent properties of VLMs, we use images as state inputs and text descriptions as outputs for state prediction. The maximum number of steps is set to 25 and the temperature is set to 0 during evaluation.

4.2 MAIN RESULTS

Our experimental results highlight the substantial advantages of the Dual Preference Optimization (D^2PO) framework over existing baselines. Results are shown in Tab. 1, and we summarize the key findings as follows:

Table 1: Performance of D²PO and baselines on VoTa-Bench (Seen). **Bold** values indicate the highest performance within the same model, and our method (D²PO), including its ablation (DPO), are highlighted in green.

	Examine&Light		Pick&Place		Stack&Place		Clean&Place		Heat&Place		Cool&Place		Overall	
	SR	PL	SR	PL	SR	PL	SR	PL	SR	PL	SR	PL	SR	PL
GPT-4o	33.33	23.37	51.19	36.27	0.00	0.00	0.00	0.00	8.41	6.55	2.38	2.02	14.39	10.37
+ ICL	41.67	30.60	64.29	45.95	4.17	1.31	1.79	1.79	24.30	23.81	11.90	11.39	23.50	18.78
GPT-4o-mini	22.22	10.88	14.29	8.16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.10	2.68
Gemini-1.5-pro	34.72	29.38	27.38	12.07	0.00	0.00	0.00	0.00	7.48	7.37	3.17	1.72	10.93	6.81
Qwen2-VL (72B)	34.72	21.62	39.29	21.81	0.00	0.00	0.00	0.00	3.97	3.47	0.79	0.56	11.66	7.10
LLaVA-1.6 (34B)	12.50	2.09	7.14	2.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.73	0.68
Qwen2-VL (7B)	26.39	8.55	14.29	8.22	2.08	0.60	0.00	0.00	0.00	0.00	0.00	0.00	5.83	2.46
+ ICL	25.00	9.25	21.43	12.29	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.56	3.14
+ SFT	70.83	55.24	69.05	57.74	6.25	5.38	26.79	26.04	58.88	58.34	31.75	31.11	44.63	40.33
+ DPO	72.22	56.67	80.95	66.30	10.42	8.47	44.64	44.64	60.75	60.75	**44.44**	44.04	53.92	49.37
+ D²PO	**84.72**	**66.67**	**84.52**	**71.27**	**12.50**	**10.23**	**48.21**	**48.21**	**66.36**	**66.36**	**44.44**	**44.33**	**58.11**	**53.33**
LLaVA-1.6 (7B)	4.17	0.67	7.14	1.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.64	0.26
+ ICL	1.39	0.22	4.76	0.76	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.91	0.15
+ SFT	56.94	45.37	63.10	51.65	12.50	9.81	31.25	31.18	50.47	50.08	30.16	29.34	41.35	37.56
+ DPO	66.67	45.77	72.62	59.17	20.83	18.20	44.64	44.64	44.86	44.86	43.65	43.07	49.54	44.38
+ D²PO	**69.44**	**52.60**	**78.57**	**65.48**	**22.92**	**19.60**	**47.32**	**47.32**	**60.75**	**60.41**	**44.44**	**44.33**	**54.83**	**50.23**
LLaMA-3.2 (11B)	12.50	2.00	4.76	0.86	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.37	0.39
+ ICL	8.33	1.33	3.57	0.57	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.64	0.26
+ SFT	58.33	44.13	72.62	47.04	8.33	6.69	30.36	26.03	46.73	46.73	35.71	31.98	42.99	35.33
+ DPO	**76.39**	59.31	78.57	62.61	12.50	9.97	29.46	25.47	43.93	43.35	36.51	34.24	46.08	39.73
+ D²PO	**76.39**	**59.63**	**88.10**	**71.32**	**14.58**	**12.19**	**38.39**	**32.97**	**48.60**	**48.26**	**39.68**	**38.80**	**51.18**	**44.84**

World Modeling Enhances Planning Performance: The consistent superiority of D²PO over standard DPO (average +9.84% SR across models) validates our core hypothesis - incorporating world modeling objectives significantly enhances the model's planning capabilities.

Learning from Mistakes: The performance gains of DPO and D²PO over SFT (average relative improvements of 15.95% and 27.29% in SR across models) underscore the value of learning from both successful and unsuccessful exploration. While SFT relies solely on successful trajectories, DPO and D²PO additionally utilize suboptimal or failed attempts, enabling the model to learn not just what to do but also what not to do. This mirrors human learning, where mistakes often provide critical insights into task dynamics and constraints.

Surpassing Process Reward Model through Environment Exploration: Our D²PO framework, with a 7B model, Qwen2-VL-7B outperforms GPT-4o (only 14.39% SR) by 43.72 points in SR, despite GPT-4o serving as the process reward model. This reveals how our framework effectively combines process guidance from larger models with environmental feedback to develop superior planning capabilities, even when the process reward model's direct performance on the task is limited.

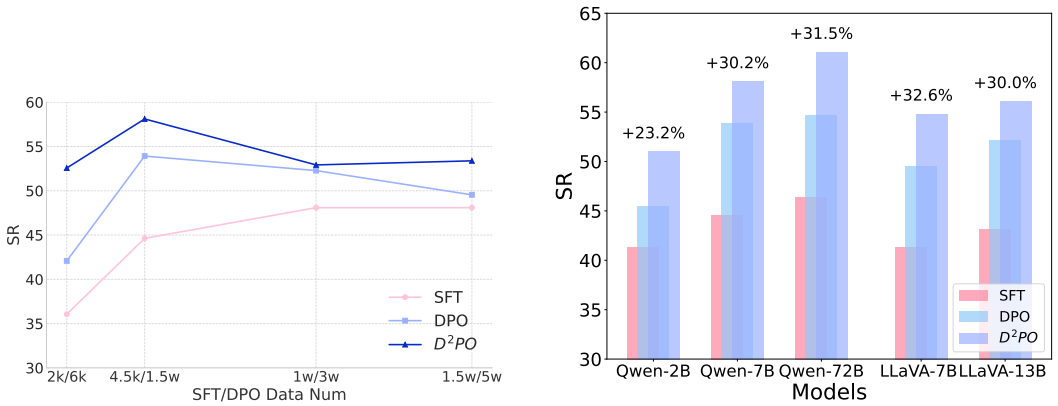
Efficiency Gains from World Model Understanding: The improved path-length weighted success rate (PL) metrics across all tasks (average +11.35% compared to DPO) indicate that our model develops physics-aware planning capabilities. Even more, in some tasks, while DPO and D²PO achieve similar SR, D²PO increases the PL, showing more efficient action sequencing through anticipated state transitions.

4.3 Generalization: Unseen Scene

We further evaluated the generalization capabilities of our model by testing it on unseen scenes that were not part of the training environment. As shown in Tab. 2, we observe that our method consistently outperforms baseline methods in both success rate (SR) and path-length weighted success rate (PL), with average relative improvements of 7.17% and 8.58% respectively across different models compared to DPO. These results demonstrate that incorporating world modeling objectives enhances the model's planning capabilities and generalization to novel environments.

Table 2: Generalization performance on VoTa-Bench (Unseen). **Bold** values indicate the highest performance within the same model, and our method (D²PO), including its ablation (DPO), are highlighted in green.

	Examine&Light		Pick&Place		Stack&Place		Clean&Place		Heat&Place		Cool&Place		Overall	
	SR	PL	SR	PL	SR	PL	SR	PL	SR	PL	SR	PL	SR	PL
Qwen2-VL (7B)	25.53	9.34	15.79	9.58	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.43	3.18
+ ICL	26.95	12.20	3.95	1.69	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.35	2.86
+ SFT	68.79	56.93	52.63	44.46	4.29	2.61	43.36	43.37	62.50	62.29	49.54	47.38	50.77	46.70
+ DPO	73.76	60.17	53.95	46.95	7.14	5.15	52.21	52.21	66.18	66.18	66.97	66.97	57.59	53.65
+ D²PO	**77.30**	**62.67**	**56.58**	**49.56**	**11.43**	**8.66**	**55.75**	**55.75**	**72.79**	**72.79**	**68.81**	**68.51**	**61.46**	**57.16**
LLaVA-1.6 (7B)	4.26	0.77	6.58	1.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.70	0.30
+ ICL	2.84	0.45	2.63	1.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.93	0.23
+ SFT	64.54	52.41	57.89	51.39	4.29	3.00	42.48	41.61	56.62	56.16	44.04	43.51	48.14	44.33
+ DPO	75.89	51.53	**60.53**	45.25	7.14	4.62	56.64	56.21	65.44	64.61	63.30	63.12	58.82	51.23
+ D²PO	**77.30**	**58.98**	**60.53**	**49.30**	**14.29**	**10.38**	**60.18**	**60.18**	**69.12**	**68.90**	**65.14**	**64.46**	**61.61**	**55.78**
LLaMA-3.2 (11B)	12.06	2.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.63	0.46
+ ICL	9.22	1.48	5.26	0.83	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.63	0.42
+ SFT	70.92	58.75	53.95	46.25	**7.14**	4.61	51.33	50.02	47.06	46.85	52.29	50.81	50.31	46.02
+ DPO	74.47	61.40	**64.47**	54.16	**7.14**	5.63	45.13	43.76	51.47	50.33	53.21	51.41	52.32	47.39
+ D²PO	**82.27**	**66.47**	**64.47**	**55.34**	**7.14**	**5.69**	**53.10**	**51.52**	**58.09**	**57.59**	**57.80**	**55.79**	**57.59**	**52.27**



(a) Impact of data scale on performance (SR).

(b) Impact of model scale on performance (SR).

Figure 3: Analysis of data scale and model scale.

5 Further Analysis

5.1 Data Scale

To investigate the impact of the data scale on performance, we varied the SFT data from 2K to 15K samples (with corresponding DPO data from 6K to 50K). Using Qwen2-VL-7B as the backbone model, our results in Fig. 3a show that D²PO consistently outperforms baselines across all data scales, achieving an average improvement of 5-15% in success rate (SR) over SFT.

As the data size increases, we observed a non-monotonic trend in D²PO's performance: initial improvements followed by plateauing or slight decline at larger scales. This phenomenon likely stems from the shared source with SFT data, where simply increasing DPO data may lead to overfitting. This highlights the importance of data quality and diversity for model generalization.

5.2 Model Scale

We further examined the effect of model scale on performance by conducting experiments with models of varying sizes, ranging from 2B to 72B parameters. As shown in Fig. 3b, performance improves as the model scale increases. Notably, D²PO consistently outperforms SFT across all model

sizes, with both methods benefiting from larger model capacities. On the largest models (Qwen 72B and LLaVA 13B), D^{2}PO achieves approximately 30% improvement in SR over baselines.

5.3 ACTION-CONDITIONED *v.s.* GOAL-DIRECTED WORLD MODELING

Inspired by recent advances in video prediction Ren et al. (2025) that demonstrate the potential of learning world dynamics without explicit actions, we investigate two distinct approaches to world modeling. The conventional action-conditioned world model learns to predict the next state based on the current state and action ($\pi(s_t|s_{t-1}, a_t)$), while the goal-directed world model directly imagines future states from history h_{t-1} and goal conditions ($\pi(s_t|g, h_{t-1})$).

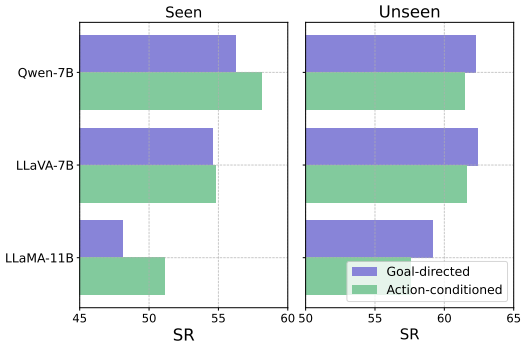


Figure 4: Success rates (SR) of action-conditioned and goal-directed world models across seen and unseen scenarios.

Our empirical analysis in Fig. 4 reveals that while the action-conditioned model achieves a higher success rate on seen scenarios, the goal-directed model demonstrates superior generalization to unseen scenarios. This suggests a fundamental trade-off: explicit action supervision helps anchor predictions in familiar contexts, but removing such constraints enhances the model's imaginative capacity, leading to more flexible dynamics learning that better generalizes to novel situations.

5.4 ERROR ANALYSIS

We classify error types by comparing standard trajectories with erroneous ones, noting that a single trajectory may contain multiple types of errors simultaneously. Through analyzing error cases of Qwen2-VL-7B in seen scenarios, Tab. 5 shows that our method significantly reduced dependency error (212 $\rightarrow$ 141), affordance error (144 $\rightarrow$ 128), and inefficient Error (141 $\rightarrow$ 78). Details are provided in Appendix E.

Table 3: Distribution of Error Types Across Different Methods

	SFT	DPO	D^{2}PO
Dependency Error	212	157	141
Affordance Error	144	141	128
Inefficient Error	141	93	78
Others	20	16	17

5.5 CASE STUDY

To better understand our approach's advantages in handling dependency constraints and efficiency, we present a detailed analysis of representative cases in Appendix F. Our case studies demonstrate how D²PO consistently produces more coherent action sequences by properly respecting dependencies between actions and generating more efficient plans compared to SFT baselines.

6 CONCLUSION

Embodied task planning requires AI systems to understand environment dynamics for effective physical interactions, yet existing approaches primarily focus on direct state-to-action mapping without considering action consequences. In this paper, we propose to learn world modeling to enhance the model's planning capability through presented Dual Preference Optimization (D^{2}PO), a new framework that jointly optimizes state prediction and action selection through preference learning. To automatically construct stepwise preference data for training, we also introduced a tree search mechanism, enabling systematic exploration and embodied experience accumulation in simulated environments. Extensive experiments on our proposed VoTa-Bench demonstrate that our 7B parameter model significantly outperforms existing approaches, including GPT-4o, across various evaluation metrics. These results validate that incorporating world modeling helps the model better understand environment dynamics, leading to improved planning capabilities.

REFERENCES

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. *ArXiv*, abs/2302.02662, 2023. URL https://api.semanticscholar.org/CorpusID:256615643.

Dongping Chen, Ruoxi Chen, Shilin Zhang, Yinuo Liu, Yaochen Wang, Huichi Zhou, Qihui Zhang, Pan Zhou, Yao Wan, and Lichao Sun. Mllm-as-a-judge: Assessing multimodal llm-as-a-judge with vision-language benchmark. In *International Conference on Machine Learning*, 2024a. URL https://api.semanticscholar.org/CorpusID:267523079.

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, 2024b. URL https://arxiv.org/abs/2311.15649.

Jae-Woo Choi, Youngwoo Yoon, Hyobin Ong, Jaehong Kim, and Minsu Jang. Lota-bench: Benchmarking language-oriented task planners for embodied agents. *ArXiv*, abs/2402.08178, 2024. URL https://api.semanticscholar.org/CorpusID:267636765.

Nicola Dainese, Matteo Merler, Minttu Alakuijala, and Pekka Marttinen. Generating code world models with large language models guided by monte carlo tree search. *ArXiv*, abs/2405.15383, 2024. URL https://api.semanticscholar.org/CorpusID:270045176.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. 2025. URL https://api.semanticscholar.org/CorpusID:275789950.

L. Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging pretrained large language models to construct and utilize world models for model-based task planning. *ArXiv*, abs/2305.14909, 2023. URL https://api.semanticscholar.org/CorpusID:258865907.

Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *ArXiv*, abs/1912.01603, 2019. URL https://api.semanticscholar.org/CorpusID:208547755.

Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *ArXiv*, abs/2010.02193, 2020. URL https://api.semanticscholar.org/CorpusID:222133157.

Danijar Hafner, J. Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. Mastering diverse domains through world models. *ArXiv*, abs/2301.04104, 2023. URL https://api.semanticscholar.org/CorpusID:255569874.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *ArXiv*, abs/2305.14992, 2023. URL https://api.semanticscholar.org/CorpusID:258865812.

Yuki Inoue and Hiroki Ohashi. Prompter: Utilizing large language model prompting for a data efficient embodied instruction following. *ArXiv*, abs/2211.03267, 2022. URL https://api.semanticscholar.org/CorpusID:253383940.

Chuhao Jin, Wenhui Tan, Jiange Yang, Bei Liu, Ruihua Song, Limin Wang, and Jianlong Fu. Alphablock: Embodied finetuning for vision-language reasoning in robot manipulation. *ArXiv*, abs/2305.18898, 2023. URL https://api.semanticscholar.org/CorpusID:258967880.

Philip Nicholas Johnson-Laird. *Mental models: Towards a cognitive science of language, inference, and consciousness*. Number 6. Harvard University Press, 1983.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Aniruddha Kembhavi, Abhinav Kumar Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *ArXiv*, abs/1712.05474, 2017. URL https://api.semanticscholar.org/CorpusID:28328610.

Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Boyu Li, Haobin Jiang, Ziluo Ding, Xinrun Xu, Haoran Li, Dongbin Zhao, and Zongqing Lu. Selu: Self-learning embodied mllms in unknown environments. *ArXiv*, abs/2410.03303, 2024. URL https://api.semanticscholar.org/CorpusID:273162831.

Jacky Liang, Wenlong Huang, F. Xia, Peng Xu, Karol Hausman, Brian Ichter, Peter R. Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500, 2022. URL https://api.semanticscholar.org/CorpusID:252355542.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024. URL https://llava-vl.github.io/blog/2024-01-30-llava-next/.

Guanxing Lu, Ziwei Wang, Changliu Liu, Jiwen Lu, and Yansong Tang. Thinkbot: Embodied instruction following with thought chain reasoning. *ArXiv*, abs/2312.07062, 2023. URL https://api.semanticscholar.org/CorpusID:266174229.

Jinjie Mai, Jun Chen, Bing chuan Li, Guocheng Qian, Mohamed Elhoseiny, and Bernard Ghanem. Llm as a robotic brain: Unifying egocentric memory and control. *ArXiv*, abs/2304.09349, 2023. URL https://api.semanticscholar.org/CorpusID:258212642.

AI Meta. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models. *Meta AI Blog. Retrieved December*, 20:2024, 2024.

OpenAI. Gpt-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

Alexander Pashevich, Cordelia Schmid, and Chen Sun. Episodic transformer for vision-and-language navigation. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15922–15932, 2021. URL https://api.semanticscholar.org/CorpusID:234482879.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *ArXiv*, abs/2305.18290, 2023. URL https://api.semanticscholar.org/CorpusID:258959321.

Zhongwei Ren, Yunchao Wei, Xun Guo, Yao Zhao, Bingyi Kang, Jiashi Feng, and Xiaojie Jin. Videoworld: Exploring knowledge learning from unlabeled videos, 2025. URL https://arxiv.org/abs/2501.09781.

Suyeon Shin, Sujin Jeon, Junghyun Kim, Gi-Cheon Kang, and Byoung-Tak Zhang. Socratic planner: Inquiry-based zero-shot planning for embodied instruction following. *ArXiv*, abs/2404.15190, 2024. URL https://api.semanticscholar.org/CorpusID:269302975.

Keisuke Shirai, Cristian Camilo Beltran-Hernandez, Masashi Hamaya, Atsushi Hashimoto, Shohei Tanaka, Kento Kawaharazuka, Kazutoshi Tanaka, Yoshitaka Ushiku, and Shinsuke Mori. Vision-language interpreter for robot task planning. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2051–2058, 2023. URL https://api.semanticscholar.org/CorpusID:264935138.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10737–10746, 2019. URL https://api.semanticscholar.org/CorpusID:208617407.

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. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530, 2022. URL https://api.semanticscholar.org/CorpusID:252519594.

Chan Hee Song, Jiaman Wu, Clay Washington, Brian M. Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2986–2997, 2022. URL https://api.semanticscholar.org/CorpusID:254408960.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization for llm agents. *ArXiv*, abs/2403.02502, 2024. URL https://api.semanticscholar.org/CorpusID:268249221.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. *ArXiv*, abs/2305.16653, 2023. URL https://api.semanticscholar.org/CorpusID:258947337.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2:160–163, 1990. URL https://api.semanticscholar.org/CorpusID:207162288.

Andrew Szot, Max Schwarzer, Harsh Agrawal, Bogdan Mazoure, Walter Talbott, Katherine Metcalf, Natalie Mackraz, Devon Hjelm, and Alexander Toshev. Large language models as generalizable policies for embodied tasks. *ArXiv*, abs/2310.17722, 2023. URL https://api.semanticscholar.org/CorpusID:264555578.

Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *ArXiv*, abs/2403.05530, 2024. URL https://api.semanticscholar.org/CorpusID:268297180.

Edward C Tolman. Cognitive maps in rats and men. *Psychological review*, 55(4):189, 1948.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Ke-Yang Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *ArXiv*, abs/2409.12191, 2024a. URL https://api.semanticscholar.org/CorpusID:272704132.

Zidan Wang, Rui Shen, and Bradly C. Stadie. Wonderful team: Zero-shot physical task planning with visual llms. 2024b. URL https://api.semanticscholar.org/CorpusID:271533474.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and P. Abbeel. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning*, 2022. URL https://api.semanticscholar.org/CorpusID:250088882.

Zhenyu Wu, Ziwei Wang, Xiuwei Xu, Jiwen Lu, and Haibin Yan. Embodied task planning with large language models. *ArXiv*, abs/2307.01848, 2023. URL https://api.semanticscholar.org/CorpusID:259342896.

Jingkang Yang, Yuhao Dong, Shuai Liu, Bo Li, Ziyue Wang, Chencheng Jiang, Haoran Tan, Jiamu Kang, Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. Octopus: Embodied vision-language programmer from environmental feedback. In *European Conference on Computer Vision*, 2023. URL https://api.semanticscholar.org/CorpusID:263909250.

Yuxiao Yang, Shenao Zhang, Zhihan Liu, Huaxiu Yao, and Zhaoran Wang. Hindsight planner: A closed-loop few-shot planner for embodied instruction following. *ArXiv*, abs/2412.19562, 2024. URL https://api.semanticscholar.org/CorpusID:275119585.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *ArXiv*, abs/2210.03629, 2022. URL https://api.semanticscholar.org/CorpusID:252762395.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models. *ArXiv*, abs/2307.02485, 2023. URL https://api.semanticscholar.org/CorpusID:259342833.

Qi Zhao, Haotian Fu, Chen Sun, and George Dimitri Konidaris. Epo: Hierarchical llm agents with environment preference optimization. *ArXiv*, abs/2408.16090, 2024. URL https://api.semanticscholar.org/CorpusID:272146208.

Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning. *ArXiv*, abs/2305.14078, 2023. URL https://api.semanticscholar.org/CorpusID:258841057.

Siyu Zhou, Tianyi Zhou, Yijun Yang, Guodong Long, Deheng Ye, Jing Jiang, and Chengqi Zhang. Wall-e: World alignment by rule learning improves world model-based llm agents. *ArXiv*, abs/2410.07484, 2024. URL https://api.semanticscholar.org/CorpusID:273233468.

A LIMITATION

Sim-to-Real Gap Similar to others in embodied task planning, our current training and evaluation are conducted in the AI2-THOR simulation environment, which may not fully capture the complexity and uncertainty of real-world scenarios, and may lead to the sim-to-real gap. Nevertheless, our learning algorithm is designed to be environment-agnostic and independent of simulation metadata, enabling potential deployment and optimization in real-world settings. Additionally, existing research efforts are actively exploring methods to bridge this gap, which could further facilitate real-world applications.

Data Collection Efficiency Given the current limitations in multimodal language models' critique capabilities Chen et al. (2024a), our data collection pipeline utilizes GPT-4o as the judge model for process rewarding, which requires additional computational resources. As vision-language models continue to advance rapidly, and with future exploration of embodied self-rewarding mechanisms, we believe these computational costs will be significantly reduced, making the framework more scalable for practical applications.

B ETHICS STATEMENT

Our research aims to develop robots that serve as assistive tools to augment human capabilities in daily tasks rather than replacing human workers, creating new opportunities for human-AI collaboration in household scenarios. To ensure responsible development and prioritize user safety, we advocate for implementing comprehensive safety protocols and monitoring mechanisms before deploying similar systems in real-world environments, particularly when handling potentially hazardous appliances.

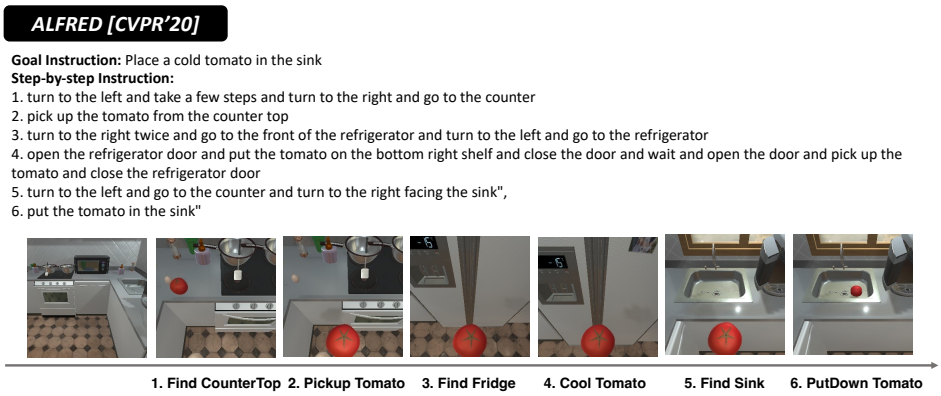
C VOTA-BENCH

C.1 TASK FORMULATION AND COMPARISON

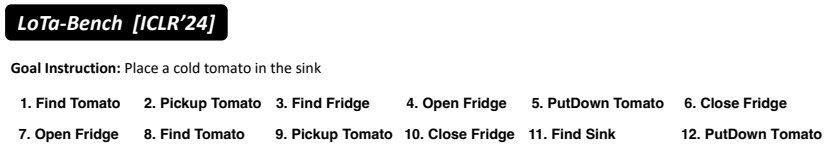
Task Formulation VoTa-Bench is designed as a closed-loop task planning framework. For each task sample, the framework consists of a natural language goal, an initial environment state detailing object locations and states (which are used to initialize the simulator), and a goal condition specifying the criteria for task completion.

The task execution follows an interactive closed-loop process. Initially, the model receives a goal instruction along with an egocentric view of the environment state. Based on these inputs, the model begins its planning process. At each step, the model plans only the next action, which is then executed in the simulation environment. The environment provides feedback including both the action execution status (success or failure) and an updated egocentric view of the new state. The model incorporates this feedback to plan its next step. This interactive process continues until either the model signals completion by outputting a "done" action or reaches the maximum allowed steps (25).

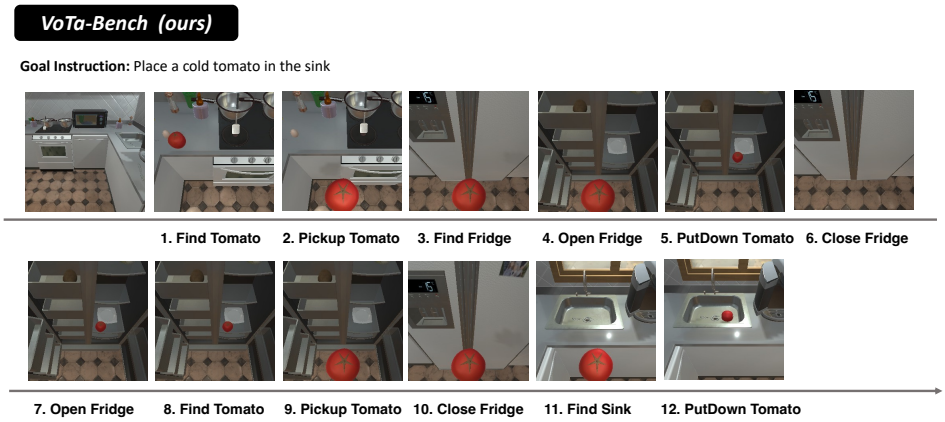
LoTa-Bench vs. ALFRED Our VoTa-Bench is based on Lota-bench. Although both LoTa-Bench and ALFRED are based on the AI2Thor simulation environment, they represent different approaches to embodied task evaluation. LoTa-Bench focuses specifically on assessing LLM's planning capabilities, providing a low-level controller to handle the execution of language actions in the simulation environment. In contrast, ALFRED evaluates models' overall performance, including low-level action execution, without decoupling task success metrics. This distinction is particularly relevant in modern hierarchical systems where LLMs serve as the embodied brain for task planning, while separate action models handle low-level execution. LoTa-Bench effectively isolates and measures the model's planning ability specifically. Furthermore, LoTa-Bench implements more fine-grained step decomposition, breaking tasks into simple, executable actions, compared to ALFRED's higher-level planning approach (Fig. 5). Another key difference lies in the instruction format: while ALFRED provides human-written step-by-step instructions to guide task planning, LoTa-Bench presents a greater challenge by providing only goal instructions.



(a) ALFRED (high-level planning) Shridhar et al. (2019)



(b) LoTa-Bench Choi et al. (2024)



(c) VoTa-Bench (ours)

Figure 5: Comparison of ALFRED, LoTa-Bench, and VoTa-Bench in the task "Place a cold tomato in the sink". (a) ALFRED emphasizes high-level task planning with human-written step-by-step instructions, breaking the task into subgoals like "Cool Tomato" (step 4). (b) LoTa-Bench provides only goal instructions and decomposes tasks into fine-grained low-level actions (e.g., "Open Fridge", "PutDown Tomato", etc.; steps 4–10) but lacks guidance from visual input, relying on predefined executable actions, choosing actions based on maximum logits to ensure they are valid in the simulation. (c) VoTa-Bench extends LoTa-Bench by incorporating egocentric visual observations, requiring models to generate open-domain actions based on visual information to handle both seen and unseen environments.

C.2 Data Statics

C.2.1 Tasks

Following the design of LoTa-Bench, VoTa-Bench incorporates 6 task types: Examine & Light, Pick & Place, Stack & Place, Clean & Place, Heat & Place, and Cool & Place. Compared to LoTa-Bench's 208 samples, we expanded the dataset to 549 samples in seen environments and further added 646 samples in unseen environments. The average action sequence length varies across different task types, ranging from 4.00 steps for simple examination tasks to 18.35 steps for more complex

Task Type	Seen		Unseen		Sample Instruction
	Num	Avg Length	Num	Avg Length	
Examine & Light	72	4.00	141	4.34	Examine a vase under a tall lamp
Pick & Place	84	4.46	77	5.70	Put pencil on bureau top
Stack & Place	48	10.60	70	8.49	Put a pot with a sponge in it in the sink.
Clean & Place	112	12.66	113	12.88	Put a cleaned washcloth away in a cabinet.
Heat & Place	107	18.35	136	17.38	To heat a potato slice and put it on the table by the spoon.
Cool & Place	126	15.48	109	14.48	Chill a knife and place a chilled slice of lettuce in a sink.
Total	549	11.85	646	10.90	

Table 4: Distribution of task types in VoTa-Bench. The dataset is divided into seen and unseen environments, with statistics showing the number of samples (Num) and average action sequence length (Avg Length) for each task type. Example instructions are provided to illustrate typical tasks.

(a) Seen Scene (b) Unseen Scene

Figure 6: Examples of seen and unseen scenes.

operations like Heat & Place, with an overall average of 11.85 steps in seen environments and 10.90 steps in unseen environments. More details is shown in Tab. 4.

C.2.2 Actions

Based on the AI2-THOR simulator, VoTa-Bench supports eight fundamental actions that can be combined to accomplish the above tasks:

- Find(<object>): A navigation action that enables the agent to locate and approach a specific object. The agent needs to identify and move to the target object's location before any interaction can occur.
- PickUp(<object>): Allows the agent to grasp and lift an object. The precondition is that the agent must be within the interaction range of the object and not currently holding anything. The effect is that the agent holds the specified object.
- PutDown(<object>): Places a held object onto the last visited receptacle. The agent must be holding the object and within range of the receptacle.
- Open(<object>): Opens containers such as cabinets, drawers, or appliances. The agent must be within the interaction range of the target object.
- Close(<object>): Closes previously opened containers. Similar to Open, requires the agent to be within the interaction range.
- TurnOn(<object>): Activates objects like lights or appliances. The agent must be within the interaction range of the target object.
- TurnOff(<object>): Deactivates previously turned on objects. Requires the agent to be within interaction range.

- Slice(<object>): Allows the agent to cut or slice certain objects. The agent must be holding an appropriate cutting tool and be within range of the target object.

Each action can only be executed when its preconditions are met, ensuring realistic interaction sequences. For example, interaction actions like "PickUp" can only be executed when the distance between the agent and the target object is within a predefined threshold. If the target object is not within visual range, the agent needs to use the "Find" action first to locate and approach the object before interaction.

C.2.3 SCENE

VoTa-Bench environments are based on the AI2-THOR simulation platform, covering four indoor scenes: Kitchen, Living Room, Bedroom, and Bathroom. We extend LoTa-Bench by introducing unseen scenes for testing generalization capability.

- Seen Scene: These household environments share identical layouts with the training set. Object positions are randomly initialized according to pre-defined commonsense distributions in AI2-THOR.
- Unseen Scene: These household environments feature different layouts from the training set. Object positions are randomly initialized according to pre-defined commonsense distributions in AI2-THOR.

Fig. 6 shows examples of layouts in our seen and unseen environments.

C.3 LICENSE STATEMENT

This work builds upon ALFRED (MIT License), AI2-THOR (Apache-2.0), and LoTa-Bench (CC BY 4.0). All modifications and derived work comply with their respective licenses.

D DETAILS OF PREFERENCE DATA

D.1 DATA CONSTRUCTION DETAILS

Our task instructions are sampled from the ALFRED dataset's training set. This process can be automated through defining formal goal conditions (including object relationships like <object> on <object> and object states like "heated"), which, combined with instruction generation capabilities of large language models, enables automated construction of large-scale instruction-goal paired datasets.

We use the Qwen2-VL-7B as the policy model for data collection with a temperature setting of 0.8, and GPT-4o (temperature = 0) is utilized as the process reward model to assess action quality (0-5). Environmental feasibility is determined through binary scoring (0/1), indicating whether an action can be physically executed in the environment. To ensure balanced consideration of both aspects, we normalize the environmental score to a 0-5 scale before averaging it with the semantic score.

Our tree search implementation employs several key parameters to maintain efficiency while ensuring thorough exploration. The selection threshold τ is set to 3.75, which creates a strict filtering mechanism: actions must be both environmentally feasible and semantically meaningful to be selected for expansion. This threshold effectively filters out non-executable actions (environmental score = 0) and executable actions with low semantic scores (< 2). To manage computational resources and maintain search efficiency, we sample 5 candidate actions for each state and set a maximum search depth of 25 steps. These parameters were determined through empirical testing to balance between exploration breadth and computational feasibility.

The hybrid scoring mechanism integrates both semantic and environmental feasibility assessments. For semantic evaluation, we employ GPT-4o to assess how effectively each step contributes to task completion, using a 5-point scale rating prompt (detailed in Appendix G).

D.2 DATA SAMPLE

For each step in the correct trajectory, we can construct preference pairs for both action selection and state prediction. For action selection, the action from the correct trajectory serves as the "chosen" action, while other actions at the same step serve as "reject" actions. Since there may be multiple incorrect actions when using tree-search methods, the number of DPO samples exceeds the number of SFT samples. The example of action selection is as follows:

Input:

<image>
Please generate the plan for the next step based on the given Goal, Previous Steps, and Images. The plan should select one action and one object from the provided list.

Goal: Place a cooked apple inside the fridge.
Previous Steps:
#Step 1: find apple (this action success)
#Step 2: pick up apple (this action success)
#Step 3: find microwave (this action success)
Action List: ['find', 'pick up', 'put down', 'open', 'close', 'slice', 'turn on', 'turn off', 'done']

Generate the next step in the format:
Reasoning:
Action:
Object:

Chosen Output:

Reasoning: To cook the apple, it needs to be placed inside the microwave. After that, it can be cooled in the fridge.
Action: open
Object: microwave

Rejected Output:

Reasoning: The user needs to put the cooked apple inside the fridge. The apple has already been found and picked up in the previous steps. The next step requires finding the fridge.
Action: find
Object: fridge

For state prediction, given the current state and correct action as input, we treat the state that the correct action transitions to as "chosen", and the states that incorrect actions lead to as "reject" states. We use GPT-4o to convert environment metadata (object visibility, spatial relationships, etc.) and visual information into natural language state descriptions. Since there might be a language style gap between GPT-4o and our model's generations, we have the model first reproduce the current state description as an anchor point before predicting the next state. The example of state prediction is as follows:

Input:

<image>
Given the current state and step, describe the current state and predict what the next state should look like.

Step: open microwave

First describe the current state, then predict the next state after the step is executed:
Current State:
Next State:

Chosen Output:

Current State: The microwave is on the counter, and the apple is held.
Next State: The microwave is open with a blue plate inside. The apple is held.

Rejected Output:

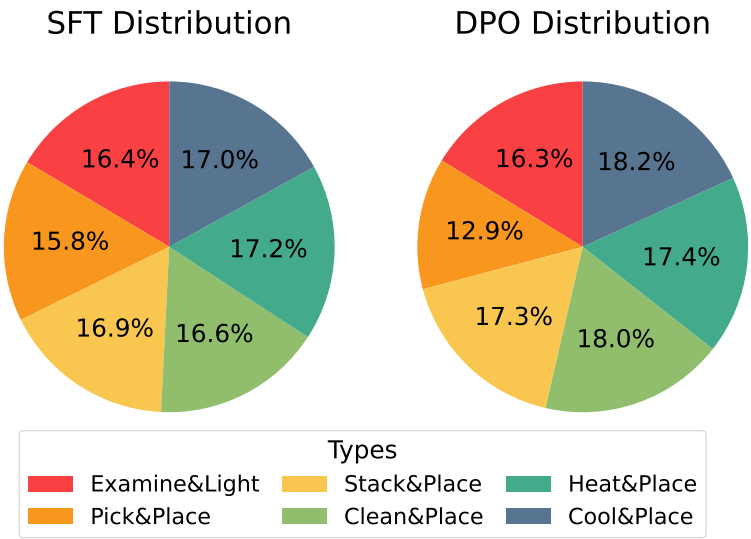


Figure 7: Distribution of the SFT and DPO dataset across different task types.

Current State: The microwave is on the counter, and the apple is held.
Next State: The fridge is visible with a temperature display showing -15, and the apple is held.

D.3 Data Distribution

To achieve a balanced dataset, we processed the collected data to ensure similar sample sizes across task types, with the detailed distribution presented in the Fig. 7.

E Error Analysis

	SFT	DPO	D^2PO
Dependency Error	212	157	141
Affordance Error	144	141	128
Inefficient Error	141	93	78
Others	20	16	17

Table 5: Distribution of Error Types Across Different Methods

To systematically analyze the error patterns, we employed Deepseek-R1 DeepSeek-AI (2025) to classify error types by comparing standard trajectories with erroneous ones. Note that a single trajectory may contain multiple types of errors simultaneously. We categorized the errors into three main types:

- **Dependency Error (DE)**: Occurs when actions are executed without meeting necessary prerequisites, violating the logical sequence of operations.
- **Affordance Error (AE)**: Manifests as incorrect object interaction sequences, indicating a misunderstanding of how to properly interact with objects in the environment. This includes both action affordance errors (using incorrect methods to interact with objects) and existence affordance errors (attempting to interact with non-existent objects).
- **Inefficient Error (IE)**: Involves redundant or unnecessary actions that do not contribute to achieving the task goal efficiently.

As shown in Tab. 5, our D^2PO method demonstrates significant improvements in reducing these error types compared to baseline methods. The analysis reveals that D^2PO particularly excels in

minimizing Dependency Errors (212 → 141), Affordance Errors (144 → 128), and Inefficient Errors (141 → 78).

However, we acknowledge certain limitations in our current approach. While we have made substantial progress in reducing these common error types, there remain opportunities for future work to further enhance the model's performance and address more complex error patterns that may emerge in different scenarios.

F CASE STUDY

We conduct case studies to demonstrate the advantages of our proposed D²PO method over SFT in terms of dependency and efficiency.

Dependency As shown in Fig. 8, our method exhibits superior dependency modeling compared to SFT in the task "put washed plate inside fridge". At step 2, SFT attempts to "pick up" without first locating an accessible plate, while our method correctly performs "find plate" before attempting any manipulation. Similarly, at step 4, SFT executes "put down plate" without having successfully picked up any plate, whereas our approach ensures proper prerequisites are met. These initial errors in SFT propagate throughout the sequence - despite multiple pick and place attempts, they remain invalid operations, ultimately resulting in task failure.

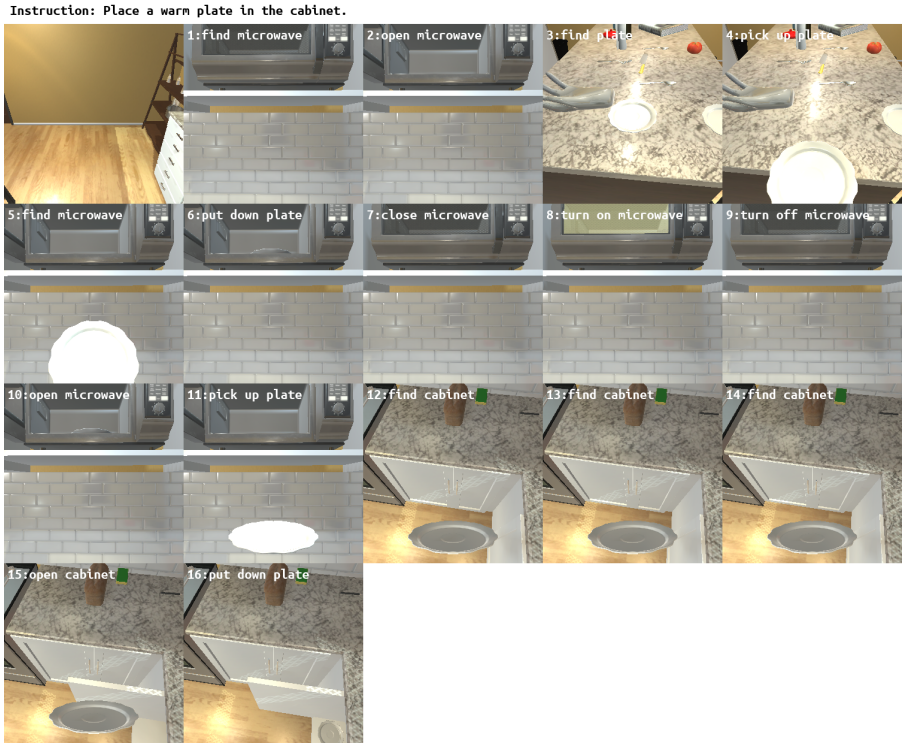
Efficiency Fig. 9 demonstrates our method's superior efficiency in the task "place a warm plate in the cabinet". Even when both approaches successfully complete the task, our method requires fewer steps through better action sequencing. D²PO first locates the plate before proceeding to operate the microwave, following a logical and efficient order. In contrast, SFT inefficiently operates the microwave before finding the plate, leading to redundant "find plate" actions in steps 1 and 5. Furthermore, SFT exhibits unnecessary repetition in steps 12-14, where it performs the same action multiple times. This comparison highlights our method's ability to generate more streamlined and efficient action sequences while maintaining task success.

G PROMPT TEMPLATE

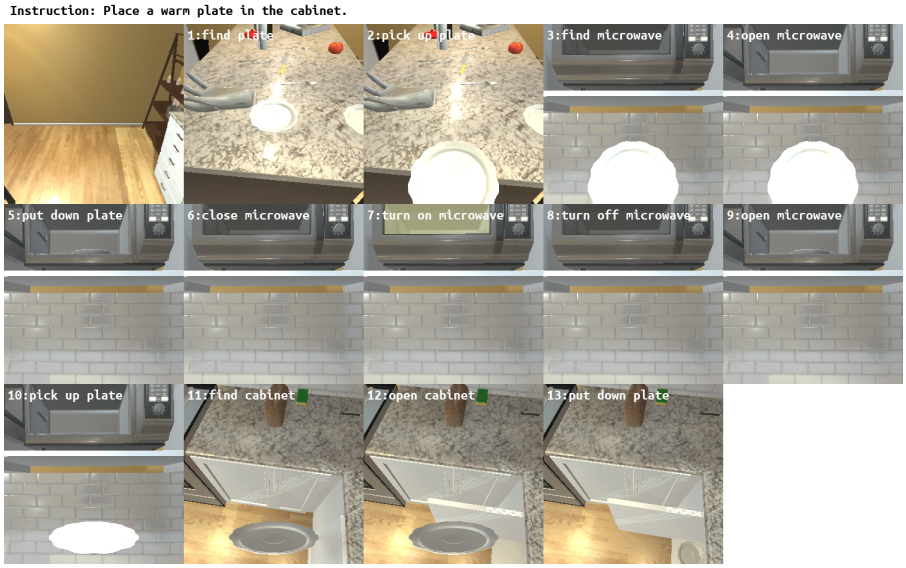
(a) SFT Trajectory (Fail)

(b) D^2PO Trajectory (Success)

Figure 8: Case Study about Dependency. This example demonstrates our method's superiority in dependency modeling compared to SFT. At step 2, SFT attempts "pick up" without locating an accessible plate, while our method first performs "find plate". Similarly, at step 4, SFT executes "put down plate" without having picked up any plate, whereas our approach ensures the plate is properly held before putting it down. These initial errors in SFT propagate throughout the sequence - despite multiple pick and place attempts, they remain invalid operations, ultimately resulting in task failure.



(a) SFT Trajectory (Success)



(b) D^2PO Trajectory (Success)

Figure 9: Case Study about Efficiency. Even when both SFT and D^2PO methods successfully complete the task, our approach requires fewer steps. Our method first locates the plate before proceeding to operate the microwave, while SFT operates the microwave before finding the plate, resulting in redundant "find plate" actions in steps 1 and 5. Additionally, SFT's repetitive execution of the same action in steps 12-14 further reduces efficiency. This comparison demonstrates our method's superior action sequencing and efficiency, even when both approaches ultimately achieve the goal.

GPT Evaluation Prompt

Please serve as an unbiased evaluator for the AI-generated next step in the task planning according to the goal progress. The task involves robotic actions that typically follow a logical sequence of steps to achieve a defined goal.
{example}
Input Data:
Goal: {goal}

Previous Steps: {previous_steps}
—
AI-generated Next Step to Evaluate:
Step: {step}
Execution Result: {action_ret}
After executing the step, you can see the following environment state: <image>

Evaluation Criteria:

Goal Progress (1-5 points):
Evaluate how effectively the step moves toward completing the task by considering:
1. **Action Sequence** - Does it follow a logical progression of actions based on the task requirements? (e.g., preparation → execution → refinement → goal completion)
2. **Previous Actions** - How does it build on prior steps? Does it avoid unnecessary repetition or conflicting actions?
3. **Goal State** - Does the step advance the task toward achieving the defined goal or final condition?
4. **Environment State** - Does the environment state after executing the step align with the expected progress toward the goal?

Scoring for Goal Progress:
- **[1]:** Step moves away from the goal or makes goal completion more difficult.
- **[2]:** Step is redundant or repeats the exact same action as the immediate previous step without progress.
- **[3]:** Step makes moderate progress toward the goal.
- **[4]:** Step makes significant progress toward the goal, aligning well with the task sequence.
- **[5]:** Step makes excellent progress, directly advancing toward goal completion.

Examples:
- A step that repeats an action unnecessarily (e.g., "find object" followed by "find object") = [2].
- A step that logically follows the sequence (e.g., "find object" before "pick up object") = [4].
- A step that conflicts with the goal (e.g., "pick up object" followed by "put down object" without correct location) = [1].

—
Output Format:
Evaluation:
Analysis: Briefly explain how the step compares to prior actions, whether it follows a logical sequence, and how it advances the goal.
Goal Progress Score: Use the following scale format: [1], [2], [3], [4], [5].

Figure 10: Prompt Template for GPT-Evaluation during the Data Collection.

Inference Prompt

<image>
Please generate the plan for the next step based on the given Goal, Previous Steps, and Images. The plan should select one action and one object from the provided list.

Goal: {query}
Previous Steps: {previous_step}

Action List: ['find', 'pick up', 'put down', 'open', 'close', 'slice', 'turn on', 'turn off', 'done']

Generate the next step in the format:
Reasoning:
Action:
Object:

Figure 11: Prompt Template for Inference.