JTEmbodiedAgent: A Domain-Adaptive Framework for Bridging Symbolic Reasoning and Physical Grounding

Benhui Zhuang Qu Tang Wenjia Zhang Bo Yuan*

JIUTIAN Research, Beijing, China
{zhuangbenhui, tangqu, zhangwenjia, yuanbo}@cmjt.chinamobile.com

Abstract

Translating natural language instructions into executable plans requires balancing symbolic rigidity with physical adaptability. In this technical report for the Embodied Agent Interface (EAI) Challenge, we present a domain-adaptive methodology that aligns optimization strategies with environmental constraints. We demonstrate that while Supervised Fine-Tuning (SFT) is sufficient for the deterministic, logic-driven nature of *VirtualHome* environment, the high-variance physics of the *Behavior* environment necessitates a Reinforcement Learning (RL) paradigm to prevent out-of-distribution forgetting and hallucination. To this end,We propose a hybrid framework that integrates Gemini 2.5-pro for high-level reasoning and sequencing with a Qwen3-14B backbone, optimized via SFT and GRPO/GMPO, to handle precise symbolic tasks and state grounding. Our results demonstrate a distinct trade-off: learned policies excel at transition modeling and goal interpretation, whereas prompt engineering outperforms in long-horizon sequencing for sparse-reward domains.

1 Introduction

The integration of Large Language Models (LLMs) into embodied systems has sparked a paradigm shift in how agents understand instructions. However, a major hurdle remains: translating abstract reasoning into actual physical movement. This 'grounding gap' often leads to hallucinations, where a plan looks correct on paper but is impossible for the agent to execute in a dynamic environment.

Embodied Agent Interface (EAI) [8] underscores this challenge by presenting two contrasting simulation environments, each requiring distinct reasoning modalities. *VirtualHome* [10] operates under strict symbolic determinism governed by Linear Temporal Logic (LTL), requiring precise adherence to syntactic and logical constraints. In contrast, *Behavior* [13] introduces a high-variance, physics-rich interaction space where combinatorial complexity and stochasticity make pure rigid symbolic planning ineffective. In practice, a monolithic approach often fails to generalize across these domains; supervised methods degrade under the distribution shifts of physics-based simulations, while prompt-based strategies lack the precise state tracking required for long-horizon consistency.

To address these distinct environmental constraints, we propose a **Domain-Adaptive Methodology** (Figure 1) that aligns policy optimization with the specific complexity of each environment. We posit that the optimization strategy must mirror the characteristics of the domain. Consequently, we employ SFT as a "semantic compiler" for the deterministic constraints of *VirtualHome* environment, and leverage RL—specifically Group Relative Policy Optimization (GRPO) and Geometric Mean Policy Optimization (GMPO)—to enforce state-grounding in the stochastic *Behavior* environment. We combine these specialized solvers with Gemini 2.5-pro, which handles high-level reasoning and sub-goal decomposition.

*Corresponding author: yuanbo@cmjt.chinamobile.com

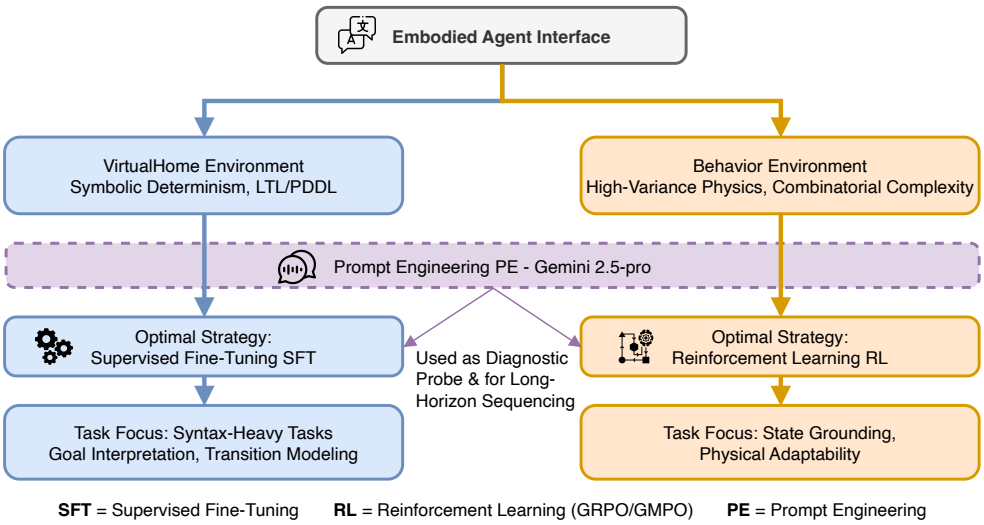

Figure 1: Overview of the Domain-Adaptive Methodology. The framework decouples reasoning based on environmental constraints: the deterministic nature of *VirtualHome* is addressed via SFT for precise symbolic adherence, while the high-variance dynamics of *Behavior* are managed using RL. The Gemini 2.5-pro serves as a central reasoning engine, providing long-horizon sequencing and diagnostic probing across both domains.

Our primary contributions are summarized as follows:

- **Domain-Adaptive Architecture:** We introduce a hybrid architecture that effectively decouples symbolic logic from physical interaction, demonstrating that assigning SFT to symbolic tasks and RL to physics-grounded tasks significantly outperforms monolithic baselines.
- **Targeted Policy Optimization:** We introduce specialized learning strategies tailored to distinct environmental challenges: a diff-based semantic extractor to ensure precise LTL adherence in *VirtualHome*, and a GMPO-based RL formulation to mitigate hallucinations and prevent policy collapse in *Behavior*.
- **Diagnostic Separation of Concerns:** Our results demonstrate a critical trade-off between local grounding and global planning, establishing that while learned policies (RL/SFT) achieve superior state transition accuracy, the prompt engineering approach remains the optimal strategy for long-horizon sequencing in sparse-reward settings.

2 Related Work

Early embodied planning relied on symbolic PDDL solvers [4], which provided guarantees of correctness but faced scalability issues due to the need for manual rule encoding. While the advent of Large Language Models (LLMs) introduced zero-shot semantic planning [5], these end-to-end approaches often suffer from grounding errors and hallucinations [1]. To enhance reasoning capabilities, prompt engineering techniques such as Chain-of-Thought (CoT) [14] and ReAct [16] have been widely adopted to elicit intermediate logic and interleaved execution. However, these in-context learning methods often lack precise state-grounding, necessitating more robust integration with environmental constraints. To bridge this gap, recent neuro-symbolic architectures have emerged, employing LLMs to translate natural language into formal logic (e.g., LTL) for verification [9, 3] or to induce symbolic world models from interaction traces [12]. Our work adopts this hybrid strategy for the EAI Challenge, leveraging SFT and RL to align LLM outputs with strict formal grammars.

The choice of optimization paradigm is equally critical for robust performance. While Supervised Fine-Tuning (SFT) effectively learns rigid instruction-following formats, it is prone to "out-of-distribution forgetting" in dynamic, physics-rich environments [6]. Conversely, Reinforcement Learning (RL) has proven superior for complex reasoning tasks. Specifically, GRPO [11] eliminates

the need for memory-intensive value networks by normalizing rewards across sampled groups. We further leverage its stabilized variant, GMPO [17], for the *Behavior* track, as it mitigates the high variance of sparse simulation rewards and prevents policy collapse in long-horizon planning.

3 Method

To comprehensively tackle the given tasks by EAI challenge, we adopt a domain-adaptive methodology that distinguishes between the strict symbolic determinism of the *VirtualHome* environment and the high-variance, combinatorial complexity of the *Behavior* environment. Our approach investigates three distinct learning paradigms: Prompt Engineering, SFT, and RL, aligning each with the specific constraints of the domain. We posit our strategy on the observation that while Prompt Engineering serves as an essential probe for identifying failure modes and affordance violations, iit lacks the granular control required for robust state-grounding. therefore, we employ SFT as our primary solver for syntax-heavy tasks in the *VirtualHome*, leveraging its ability to precisely clone rigid formal logic and syntax. However, o mitigate the distribution shift often seen in learned policies, we retain a high-level prompt-based planner for long-horizon sequencing. Conversely, for the *Behavior* domain, where the solution space is vast and physics-rich, we utilize RL (specifically GRPO and GMPO) to ensures active state-grounding and prevent the policy collapse and distribution shift that supervised methods frequently suffer in stochastic environments.

3.1 Diagnostic Analysis via Prompt Engineering

We employ the Prompt Engineering method not merely as a baseline, but as a diagnostic probe to characterize LLM failure modes in embodied settings. By analyzing execution logs from the Gemini 2.5-pro model [2] across both domains, we identified that the predominant error modes stem from a misalignment between the model's generation and the simulator's strict state-dependent constraints. Specifically, our analysis revealed a distinct lack of internal modeling regarding action preconditions—such as attempting *SWITCHON* without *PLUGIN* in *VirtualHome*, or executing *CLEAN_STAIN* without tool preparation in *Behavior*. Furthermore, affordance-related failures and hallucinations indicated that generic language models struggle to maintain environment consistency, frequently referencing objects absent from the scene graph.

These findings directly informed the design of our downstream learning pipelines. To mitigate precondition errors, we found that explicitly encoding dependency structures for high-risk actions (e.g., *GRAB*, *OPEN*) within the context window significantly improved logical sequencing. Similarly, to address hallucinations, we integrated strict object-state validity checks and environment-consistency enforcement directly into the reasoning prompt. Crucially, these constraints—originally tested via prompt context—were subsequently formalized into the loss functions and data construction pipelines of our SFT and RL approaches.

3.2 Symbolic Determinism and Supervised Fine-Tuning

The *VirtualHome* environment demands precise adherence to symbolic states and long-horizon logic. To map natural language to the requisite Linear Temporal Logic (LTL) and PDDL constraints, we employ a specialized SFT approach using the Qwen3-14B backbone [15]. While we utilize Gemini 2.5-pro for complex Subgoal Decomposition (see Section 3.1), SFT provides the granular control necessary for valid syntax generation. Our training corpus consists of approximately 1,500 trajectories curated from open-source *VirtualHome* data, excluding EAI challenge test IDs to ensure integrity.

Goal Interpretation via Semantic Extraction. For Goal Interpretation task, we prioritize the *semantic outcome* of an instruction over procedural steps. Our analysis of the develop-phase data revealed that ground-truth LTL formulas are predominantly state-centric (e.g., "the light is ON"). To replicate this, we implemented a Semantic-Aware Relation Extractor that computes the "diff" between initial and final scene graphs. This diff encapsulates the essential state changes required by

the task. Recognizing that some tasks are inherently procedural, we integrated an Action Supplement Mechanism; if the scene-graph diff yields insufficient state evidence, the system falls back to extracting key action predicates (e.g., `WASH`) from the ground-truth scripts, ensuring the generated LTL remains both valid and executable.

Transition Modeling with Diff-Based Augmentation. For Transition Modeling, the objective is to generate PDDL problem files that accurately reflect preconditions and effects. To mitigate simulator noise, we propose a Diff-Based Goal Extraction strategy. We define O_{diff} as objects undergoing state changes not explicitly mentioned in the execution script. The final goal set G is constructed by unifying explicit execution goals with a minimal subset of diff-based goals:

$$G = G_{\text{exec}} \cup \{g \in G_{\text{diff}} \mid \text{arg}(g) \cap O_{\text{diff}} \neq \emptyset\}$$

To prevent overfitting to specific object instances, we implement a Combinatorial Variant Generation pipeline, which generates powerset variants of O_{diff} and permutes the goal sets, forcing the model to distinguish relevant targets from distractors, thereby ensuring robustness against under-specified goal conditions.

3.3 Combinatorial Complexity and Reinforcement Learning

The *Behavior* environment presents a combinatorial interaction space governed by stochastic physics, making SFT prone to out-of-distribution forgetting. To address this, we formulate policy learning as a reinforcement learning problem, optimizing a policy π_θ to generate action plans a conditioned on context s. We employ GRPO [11] to eliminate value network overhead by estimating advantages $\hat{A}_i$ via group-normalized rewards.

For Subgoal Decomposition, direct simulation rewards are computationally expensive and sparse. To accelerate learning, we bootstrap the agent using human demonstrations rather than real-time execution. We generate supervision by replaying annotated traces, applying retry heuristics (e.g., auto-releasing objects) to ensure precondition satisfaction. By analyzing semantic graph differences between post-action snapshots, we extract dense subgoal chains: added predicates (e.g., *cooked*) serve as positive subgoals, while removed ones are negated. This anchors fine-tuning to executable sequences, preventing high-entropy policy drift.

To steer the model toward executable outputs and bridge the syntax-semantics gap, we design specialized, dense reward functions $R(s, a) \in [0, 1]$ for each subtask. A goal-interpretation reward R_{GI} that scores the predicted goal graph against the annotated configuration while penalizing hallucinated entities and invalid formats; a subgoal reward R_{SG} that favors complete, well-ordered, and concise decompositions; an action-sequencing reward R_{AS} that combines terminal task success with predicate- and execution-level feedback to reward executable, grammatically valid action traces while penalizing affordance violations; and a transition-modeling reward R_{TM} that evaluates predicted preconditions and effects to improve local dynamics modeling. Collectively, these rewards couple high-level success with fine-grained diagnostic penalties to suppress hallucination and enforce temporal consistency; their formal definitions and hyperparameters are provided in Appendix A.4.

4 Evaluation Results

We evaluate our proposed approach on the official EAI Challenge benchmark suite. Our experimental framework compares three distinct policy learning paradigms: PE, SFT, and RL. The results presented below reflect the optimal strategy selected for each specific task and domain combination.

4.1 Summary of Best Results

Table 1 provides a comprehensive summary of our top-performing configurations. Our analysis reveals a domain-specific dichotomy: the symbolic determinism of *VirtualHome* favors a hybrid of

Supervised Fine-Tuning for syntactic adherence and Prompt Engineering for planning, while the high-variance physics of *Behavior* necessitates leveraging Reinforcement Learning for state-grounding tasks alongside robust Prompting for long-horizon sequencing.

Table 1: Summary of best results selected among Prompting, SFT, and RL methods (Eval Phase). The "Best Method" column indicates the approach that yielded the highest performance for that specific task/domain pair.

Domain	Task	Best Method	Primary Metric	Score
Behavior	Goal Interpretation	RL (GRPO)	F1 Score	86.80
	Subgoal Decomposition	Prompting	Task SR	80.00
	Action Sequencing	Prompting	Task SR	83.00
	Transition Modeling	RL (GRPO)	F1 Score	78.80
VirtualHome	Goal Interpretation	SFT (Qwen-14B)	F1 Score	53.20
	Subgoal Decomposition	Prompting	Task SR	71.10
	Action Sequencing	Prompting	Task SR	70.20
	Transition Modeling	SFT (Qwen-14B)	Success SR	99.80

4.2 Prompt Engineering Performance

Table 2 details the performance of the Prompt Engineering baseline. We utilized Gemini 2.5-pro as a diagnostic probe and a primary solver. Notably, prompt-based methods demonstrated superior zero-shot reasoning capabilities in the **Behavior** domain for sequential planning tasks (Subgoal Decomposition and Action Sequencing), outperforming learned policies which struggled with the high dimensionality of the physics-rich action space. A similar trend was observed in the **VirtualHome** environment, where prompt-driven models demonstrated greater robustness in generating coherent multi-step plans and handling object–state dependencies. These results collectively underscore the advantages of prompt engineering as a lightweight yet effective strategy for enhancing planning quality in embodied AI tasks without requiring additional training.

Table 2: Detailed performance metrics for the Prompt Engineering approach across Behavior and VirtualHome environments.

Domain	Task	F1 Score	Task SR	Exec SR	Planner SR
Behavior	Goal Interpretation	85.10	–	–	–
	Subgoal Decomposition	–	80.00	87.00	–
	Action Sequencing	–	83.00	92.00	–
	Transition Modeling	59.40	–	–	97.00
VirtualHome	Goal Interpretation	45.60	–	–	–
	Subgoal Decomposition	–	71.10	89.20	–
	Action Sequencing	–	70.20	82.80	–
	Transition Modeling	48.30	–	–	91.20

4.3 SFT Performance

We applied Supervised Fine-Tuning (SFT) primarily to the *VirtualHome* scenario to address the strict Linear Temporal Logic (LTL) requirements. As shown in Table 3, SFT significantly outperformed prompt-based baselines on tasks requiring precise syntactic adherence, such as Goal Interpretation and Transition Modeling.

Specifically, fine-tuning Qwen3-14B allowed for a **25.7% absolute improvement** in Goal Interpretation F1 score compared to the Gemini baseline. Furthermore, data augmentation strategies in Transition Modeling pushed the Success Rate to near-perfect levels (97.60%).

Table 3: Performance comparison of Supervised Fine-Tuning (SFT) on the *VirtualHome* benchmark tasks (Dev Phase). **SR** denotes Success Rate. Bold values indicate the best performance per metric.

Task	Metric	Prompting (Qwen-14B)	Prompting (Gemini)	SFT (Ours)
Goal Interpretation	F1 Score	26.30	45.60	**71.30**
Subgoal Decomposition	Task SR	75.10	**88.50**	75.70
Subgoal Decomposition	Exec SR	83.10	**91.40**	84.90
Action Sequencing	Task SR	67.20	**76.10**	66.50
Action Sequencing	Exec SR	74.40	**80.00**	79.20
Transition Modeling	F1 Score	42.20	48.30	**96.80**
Transition Modeling	Success SR	11.50	91.20	**97.60**

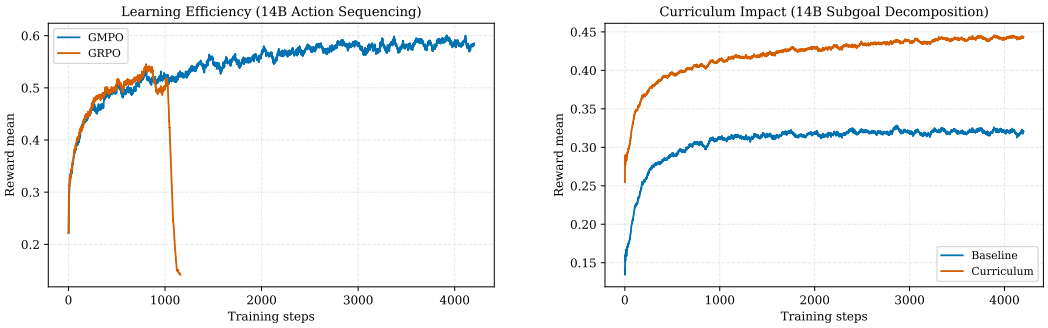

Figure 2: Learning Dynamics (RL). **Left:** GMPO stabilizes the 14B Action Sequencing policy, preventing the policy collapse observed in standard GRPO after $\approx 1.1k$ steps. **Right:** The curriculum-based approach for Subgoal Decomposition maintains reward growth in later training stages, resulting in a final reward ≈ 0.12 higher than the static baseline.

4.4 Reinforcement Learning Performance

In the *Behavior* scenario, we deployed Reinforcement Learning to mitigate hallucination and enforce grounding. Table 4 summarizes the performance of our GRPO and GMPO policies. While RL did not exceed the Prompting baseline (Gemini-2.5-pro) for long-horizon sequencing tasks (likely due to the sparse reward landscape), it achieved state-of-the-art results in goal interpretation (F1 86.8%) and significantly improved the grounding of transition models compared to zero-shot approaches.

Impact of Optimization Strategy. The substitution of GRPO with GMPO for Action Sequencing proved critical. As illustrated in Figure 2 (Left), standard GRPO suffered from policy collapse due to high variance in the sparse reward signal. GMPO facilitated stable convergence to a reward of 0.59, translating to a 71% execution success rate with negligible syntax or hallucination errors.

Curriculum Efficacy. For Subgoal Decomposition, the dynamic weighting schedule significantly improved asymptotic performance. The curriculum raised the tail reward from 0.32 (static baseline) to 0.44 (Figure 2, Right). This training gain directly correlated with a $+2\%$ improvement in execution success on the evaluation set, validating the hypothesis that enforcing structural constraints prior to semantic optimization leads to more robust policy learning.

Table 4: Performance benchmarks across the four EAI tasks using the RL method. Training reward represents the mean of the final 200 optimization steps. All models utilize the 14B backbone.

Task (Method)	Train Reward	Primary Metric	Secondary Metric
Action Sequencing (GMPO)	0.59 ± 0.09	Task SR 55.00	Exec. SR 71.00
Subgoal Decomp. (GMPO + curr.)	0.44 ± 0.05	Task SR 37.00	Exec. SR 43.00
Goal Interpretation (GRPO)	0.78 ± 0.01	F1 86.80	Precision 88.40
Transition Modeling (GRPO)	0.70 ± 0.04	F1 78.80	Planner SR 95.00

5 Analysis and Discussion

5.1 Comparative Methodology and Domain Adaptation

Our empirical evaluation uncovers a fundamental dichotomy in embodied planning: the trade-off between symbolic adherence and physical adaptability. The divergence in performance across the three optimization paradigms—Prompt Engineering, Supervised Fine-Tuning (SFT), and Reinforcement Learning (RL)—is strictly correlated with the entropic profile of the target environment. In *VirtualHome*, where the state space is deterministic and governed by rigid Linear Temporal Logic, SFT functions effectively as a "semantic compiler," achieving near-perfect transition modeling by cloning the logical structures of human demonstrators. Conversely, the *Behavior* environment introduces stochastic physics and combinatorial complexity that render simple pattern matching insufficient. Here, SFT suffers from distribution shift, whereas RL (specifically GMPO) significantly improves local grounding by optimizing directly against environmental feedback. However, we observe that while learned policies excel at these local tasks, they often lack the global coherence required for long-horizon sequencing, a domain where the massive reasoning priors of Gemini 2.5-pro still hold a distinct competitive advantage.

5.2 Diagnostic Profiling via Prompt Engineering

To characterize failure modes in general-purpose language models, we analyzed execution logs from the Gemini 2.5-pro baseline. In the *Behavior* domain, an evaluation of 100 Action Sequencing and Subgoal Decomposition cases revealed that errors primarily stem from discrepancies between the model's world representation and the simulator's physical constraints. Missing preconditions caused 23% and 34% of failures in these respective tasks, typified by interactions with inaccessible objects or attempting *CLEAN_STAIN* without preparation (e.g., *SOAK*). Additionally, affordance and temporal causality violations were frequent, suggesting that while LLMs possess abstract semantic knowledge, they lack the internal simulation capabilities required to manage temporal state dependencies.

Conversely, the *VirtualHome* domain exhibited a distinct error distribution dominated by hallucination and logical redundancy. Analysis of 61 Action Sequencing errors highlighted the insertion of unnecessary primitives (e.g., *PLUGIN* for non-electric objects) and state-tracking deficiencies, such as issuing *WALK* commands while the agent remained in a *SIT* posture. This contrast underscores the limitations of zero-shot prompting in maintaining strict schema adherence within symbolic environments, highlighting the necessity of structural constraints provided by fine-tuning.

5.3 Efficacy of Supervised Fine-Tuning

Our investigation into Supervised Fine-Tuning on the *VirtualHome* benchmark focused on model scaling laws and data efficiency. We established the Qwen3-14B architecture as the Pareto-optimal choice; empirical comparisons demonstrated that while it significantly outperforms the 0.6B variant in reasoning capacity, the marginal gains provided by the 32B model did not justify the increased computational overhead. The effectiveness of SFT proved highly sensitive to the "solution entropy" of the subtask. For constrained tasks like Goal Interpretation and Transition Modeling—where the mapping from state to output is relatively deterministic—fine-tuning achieved substantial gains.

However, for open-ended tasks such as Subgoal Decomposition, where the trajectory manifold is high-variance, SFT yielded negligible improvements over baselines, suggesting that the limited scale of the *VirtualHome* dataset is insufficient to capture the full diversity of valid plans.

Furthermore, we identified data density as a critical bottleneck for robust transition modeling. Initial SFT experiments on the raw dataset yielded a Success Rate (SR) of 75% due to the sparsity of object interaction examples. By implementing a combinatorial variation-based augmentation pipeline, we expanded the training distribution, resulting in a dramatic performance uplift to 99% SR (Table 5). This finding underscores that for symbolic reasoning tasks, the diversity of syntactic permutations is as critical as the volume of unique semantic scenarios.

Table 5: Ablation study on data augmentation for Transition Modeling (*VirtualHome* test phase). Combinatorial expansion of the training set is critical for generalization.

Setting	**F1 Score**	**Success Rate**
Original Data	92.0%	75.0%
Augmented Data	**99.0%**	**99.0%**

5.4 Reinforcement Learning Dynamics and Grounding

In the *Behavior* environment, the application of Reinforcement Learning effectively resolved the syntax-semantics gap but exposed deeper theoretical limitations regarding spatial reasoning. Our reward shaping strategy successfully eliminated superficial syntax errors, reducing parsing failures to near zero. However, execution analysis reveals a persistent "grounding gap." In Goal Interpretation, while the agent achieved high recall for object state attributes (94.8%), it struggled with relational grounding (82.5%), indicating a deficiency in inferring spatial connectivity between entities without explicit Chain-of-Thought reasoning.

This spatial deficit is further corroborated by the Transition Modeling results, where the RL agent displayed a significant performance divergence between non-spatial relations (F1 91.6) and spatial relations (F1 64.8). The model appears to adopt a conservative policy regarding contact dynamics, frequently failing to predict geometric preconditions such as *inside* or *on_top*. Additionally, the sub-optimal performance in Subgoal Decomposition (37% task success) highlights a tension between inference latency and semantic coverage. By disabling intermediate reasoning tokens to optimize training throughput, we likely restricted the model's planning horizon, resulting in plans that are physically plausible (low affordance violation rate of 3%) but semantically truncated (40% missing intermediate steps). This suggests that future architectures must integrate explicit geometric priors or hierarchical reasoning steps to bridge the gap between valid syntax and complete physical execution.

6 Conclusion

Our work demonstrates that effective embodied planning relies on matching the learning algorithm to the domain's constraints. We validated that SFT is a superior a solver for the symbolic rigidity of*VirtualHome*, whereas the stochastic dynamics of *Behavior* demand the active exploration of RL. Although our GMPO-based approach successfully mitigated hallucination in short-horizon tasks, the continued dominance of prompt engineering for long-horizon sequencing highlights the challenge of maintaining semantic coherence in learned policies. Future work will focus on reconciling these paradigms, moving beyond modular hand-offs to fully integrated systems that maintain semantic consistency across long time scales.

References

[1] Trishna Chakraborty, Udita Ghosh, Xiaopan Zhang, et al. HEAL: An empirical study on hallucinations in embodied agents driven by large language models. *arXiv preprint arXiv:2506.15065*,

2025.

[2] Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

[3] Danil S. Grigorev, Alexey K. Kovalev, and Aleksandr I. Panov. VerifyLLM: LLM-based pre-execution task plan verification for robots. *arXiv preprint arXiv:2507.05118*, 2025.

[4] Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[5] Brian Ichter, Anthony Brohan, Yevgen Chebotar, et al. Do as I can, not as I say: Grounding language in robotic affordances. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 205, pages 287–318. PMLR, 2022.

[6] Hangzhan Jin, Sitao Luan, Sicheng Lyu, et al. RL fine-tuning heals OOD forgetting in SFT. *arXiv preprint arXiv:2509.12235*, 2025.

[7] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, et al. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, pages 611–626, 2023.

[8] Manling Li, Shiyu Zhao, Qineng Wang, et al. Embodied agent interface: Benchmarking LLMs for embodied decision making. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:100428–100534, 2024.

[9] Jason Xinyu Liu, Ankit Shah, George Konidaris, Stefanie Tellex, and David Paulius. Lang2LTL-2: Grounding spatiotemporal navigation commands using large language and vision-language models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2325–2332, 2024.

[10] Xavier Puig, Kevin Ra, Marko Boben, et al. VirtualHome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8494–8502, 2018.

[11] Zhihong Shao, Peiyi Wang, Qihao Zhu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[12] Miglani Shivam and Yorke-Smith Neil. NLtoPDDL: One-shot learning of PDDL models from natural language process manuals. In *ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, 2020.

[13] Sanjana Srivastava, Chengshu Li, Michael Lingelbach, et al. BEHAVIOR: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 164, pages 477–490. PMLR, 2021.

[14] Jason Wei, Xuezhi Wang, Dale Schuurmans, et al. Chain-of-Thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[15] An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[16] Shunyu Yao, Jeffrey Zhao, Dian Yu, et al. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[17] Yuzhong Zhao, Yue Liu, Junpeng Liu, et al. Geometric-Mean Policy Optimization. *arXiv preprint arXiv:2507.20673*, 2025.

A Implementation Details

A.1 Optimization Hyperparameters

We utilize AdamW with a learning rate of 1×10^{-6}, an undiscounted return ($\gamma = 1.0$), and a symmetric PPO clipping range of ± 0.4. The KL divergence penalty is applied as a loss term with a coefficient of 10^{-3} using the low-variance formulation. Entropy regularization is disabled.

A.2 Infrastructure and Batching

Rollouts are generated using vLLM [7] with tensor parallelism of 2, utilizing BF16 FSDP (without actor offload) and gradient checkpointing. The maximum prompt and response lengths are set to 8192 and 4096 tokens, respectively.

- **Goal Interpretation:** Trains with a global batch size of 64 (PPO mini-batch 32, micro-batch 4). The "thinking" template is enabled to strengthen graph grounding.
- **Transition Modeling:** Utilizes a 32/32/4 batch layout with thinking disabled.
- **Action Sequencing & Subgoal Decomposition:** Due to higher computational costs, we employ a 16/16/2 batch layout and disable thinking to prevent rollout timeouts. A KL micro-batch of 4 is shared between the actor and reference models.

A.3 Curriculum Schedule

For Subgoal Decomposition, we implement a three-phase curriculum:

1. **Warmup (0–1500 steps):** Emphasizes JSON formatting and recall.
2. **Ramp (1500–3000 steps):** Introduces ordering penalties.
3. **Final (3000–5000 steps):** Applies full weights.

Our logged run progressed through 4,500 steps, reaching a tail reward of 0.44 ± 0.05.

A.4 Reward Function Definitions and Hyperparameters

We now provide the exact definitions of the dense reward functions used in Section 3.3, followed by the corresponding hyperparameter settings. Each reward decomposes into interpretable components that capture task success, diagnostic structure, and penalties for hallucinated or invalid predictions.

For **Goal Interpretation**, the reward R_{GI} operates on the structured goal graph inferred from the instruction. Let $F1_{\text{overall}}$ denote the F1 score for full graph matching, $F1_{\text{state}}$ and $F1_{\text{rel}}$ the F1 scores over object states and relations respectively, and $\mathbb{I}_{\text{valid}} \in \{0, 1\}$ indicate whether the predicted graph satisfies basic well-formedness constraints (for example, type correctness and connectivity). Furthermore, let $\mathbb{I}_{\text{fmt}} \in \{0, 1\}$ flag formatting violations in the textual representation, and let $R_{\text{s\_hall}}$ and $R_{\text{o\_hall}}$ quantify state- and object-level hallucination scores. The goal-interpretation reward is then defined as

$$R_{\text{GI}} = w_o F1_{\text{overall}} + w_s F1_{\text{state}} + w_r F1_{\text{rel}} + w_v \mathbb{I}_{\text{valid}} - p_f \mathbb{I}_{\text{fmt}} - p_s R_{\text{s\_hall}} - p_o R_{\text{o\_hall}}. \quad (1)$$

This formulation rewards faithful reconstruction of the target goal graph while explicitly discouraging ill-typed outputs, malformed text, and hallucinated entities or attributes.

For **Subgoal Decomposition**, the agent predicts an ordered list of symbolic subgoals that should collectively realize the target task. We decompose the reward as

$$R_{\text{SG}} = R_{\text{base}} + R_{\text{cov}} + R_{\text{fmt}} + R_{\text{len}}. \quad (2)$$

Here, R_{base} aggregates line-level and predicate-level F1 scores between predicted and reference subgoal sequences, thereby encouraging locally correct subgoal predicates. The term R_{cov} rewards semantic coverage over ground-truth categories, ensuring that all required high-level objectives are included. The term R_{fmt} penalizes violations of the required textual or structural format (for example, missing delimiters or incorrect argument structure), while R_{len} imposes a soft penalty on excessively long or redundant subgoal lists, nudging the agent toward concise yet complete decompositions.

For **Action Sequencing**, we evaluate low-level action programs that are executed in the *Behavior* simulator. Let $\mathbb{I}_{\text{goal}} \in \{0, 1\}$ be a binary indicator of task success at the end of the rollout, R_{pred} the ratio of satisfied symbolic predicates among those required by the task, and R_{exec} the fraction of actions that are successfully parsed and executed without runtime errors. We additionally define $\mathbb{I}_{\text{gram}} \in \{0, 1\}$ as an indicator of grammatical well-formedness of the action string, and P_{err} as per-action penalties capturing errors such as invalid affordances, missing required arguments, or out-of-context object references. The action-sequencing reward is given by

$$R_{\text{AS}} = w_g \mathbb{I}_{\text{goal}} + w_p R_{\text{pred}} + w_e R_{\text{exec}} + w_{gr} \mathbb{I}_{\text{gram}} - \sum P_{\text{err}}. \tag{3}$$

This reward integrates sparse success signals with dense execution feedback, providing gradients that discourage syntactic and semantic errors even when the overall task fails.

Finally, for **Transition Modeling**, we supervise the model to predict preconditions and effects for individual actions. Let S_{pre} and S_{eff} denote accuracy scores over precondition and effect predicates, respectively, computed by comparing predicted symbolic transitions to those observed in the simulator. The transition-modeling reward is defined as

$$R_{\text{TM}} = w_{\text{pre}} S_{\text{pre}} + w_{\text{eff}} S_{\text{eff}}. \tag{4}$$

By weighting precondition and effect accuracy symmetrically, this objective discourages the agent from systematically over- or under-estimating action consequences and thereby promotes robust local dynamics modeling.

The specific numerical values for all weights and penalties used in experiments are summarized in Tables 6, 7, 8, and 9 that follow.

Table 6: Hyperparameters for Goal Interpretation Reward (R_{GI})

Term	Symbol	Value
Overall F1 Weight	w_o	0.5
State F1 Weight	w_s	0.2
Relation F1 Weight	w_r	0.2
Validity Weight	w_v	0.1
Format Penalty	p_f	0.3
State Hallucination Penalty	p_s	0.2
Object Hallucination Penalty	p_o	0.2

Table 7: Hyperparameters for Action Sequencing Reward (R_{AS})

Term	Symbol	Value
Goal Success Weight	w_g	0.5
Predicate Ratio Weight	w_p	0.3
Execution Ratio Weight	w_e	0.1
Grammar Weight	w_{gr}	0.1
Hallucination Penalty	p_{hall}	0.2
Order Violation Penalty	p_{order}	0.1
Missing Step Penalty	p_{miss}	0.1
Affordance Penalty	p_{aff}	0.2
Additional Step Penalty	p_{add}	0.05

B Biography of all team members

Our team, participating under the name **JTEmbodiedAgent**, is composed of researchers from **JIUTIAN Research**, China Mobile Communications Group Co., Ltd. The team is dedicated to

Table 8: Hyperparameters for Subgoal Decomposition Reward (R_{SG})

Term	Symbol	Value
Line Match Weight	w_{line}	0.35
Predicate Match Weight	w_{pred}	0.25
Exact Match Weight	w_{exact}	0.1
Precision Weight	w_{prec}	0.2
Subsequence Weight	w_{subseq}	0.1
Recall Weight	w_{recall}	0.15
Coverage Weight	w_{cov}	0.15

Table 9: Hyperparameters for Transition Modeling Reward (R_{TM})

Term	Symbol	Value
Precondition Score Weight	w_{pre}	0.5
Effect Score Weight	w_{eff}	0.5

advancing the field of Embodied AI.