# ReCAPA: Hierarchical Predictive Correction to Mitigate Cascading Failures

**Anonymous authors**
Paper under double-blind review

## Abstract

Vision–Language–Action (VLA) agents follow instructions to perform multi-step tasks in multimodal environments. To support planning and execution in such settings, many approaches typically adopt structured post-hoc or rely on fixed decomposition and rigid alignment to improve success rate. However, once an intermediate subgoal or action is mis-specified and without a flexible correction mechanism, local errors propagate through subsequent steps and eventually accumulate into cascading failures in long-horizon reasoning. To mitigate this compounding effect, we propose Reflective Contrastive Alignment and Planning Architecture (ReCAPA), a framework that uses predictive correction to anticipate deviations and adjust representations across three levels: actions, subgoals, and trajectories. Semantic alignment is enforced at all levels using a Sinkhorn-based module and a Score-field module. The corrective signals, derived from predictive correction and alignment mechanisms, jointly update the execution network during training, enabling it to flexibly adjust fine-grained steps to remain aligned with the overall intent. We further introduce two new metrics to quantify error propagation and recovery processes in tasks. Experiments show that ReCAPA achieves competitive results on embodied agent benchmarks such as VisualAgentBench, MineDojo, and MAP-THOR, outperforming strong proprietary and open-source Large Language Model (LLM) baselines.

## 1 Introduction

VLA agents powered by LMMs are increasingly applied to long-horizon tasks in embodied environments, such as household manipulation, indoor navigation, and multi-turn human-robot dialogue (Jiang et al., 2023). These tasks require perception, planning, and grounded execution under natural language guidance. Yet many systems struggle to generalize across multi-step environments, as they lack structured visual–language grounding and often collapse under semantic drift and cascading errors(Comanici et al., 2025; Anthropic, 2024a; Achiam et al., 2023).

Recent VLA agents such as Re-ReST (Dou et al., 2024), LLaMAR (Nayak et al., 2025), and CityNavAgent (Zhang et al., 2025) shift the focus from fine-grained token-level decisions to subplans, which serve as goal-oriented anchors that reduce the risk of step-by-step error accumulation. Both TrajPrompt (Tsao et al., 2024) and PRET(Lu et al., 2024) incorporate alignment between instructions and trajectories through semantic matching or trajectory fidelity, thereby strengthening the coherence between task intent and execution. However, semantic drift and error propagation remain the key bottlenecks of long-horizon reasoning. On one hand, post-hoc correction and predefined segmentation makes it difficult to flexibly adjust its actions when the environment or task changes. On the other hand, relying only on local-level alignment without global corrective signals leads to each step being optimized in isolation, thereby drifting from the overall intent easily. In benchmarks such as VirtualHome(Puig et al., 2018) and AI2-THOR(Kolve et al., 2017), even a single subgoal error can degrade the performance of subsequent steps by over 60% (Zhong et al., 2024; Zhu et al., 2021). In sum, current agents may encounter execution–goal divergence and cumulative rollout errors, increasing the risk of cascading failures over time.

Addressing cascading failures across different temporal scales is critical: action errors may compound rapidly in the short term, whereas strategy misalignments unfold more slowly and gradually distort the overall plan. These distinct propagation patterns suggest that effective correction should

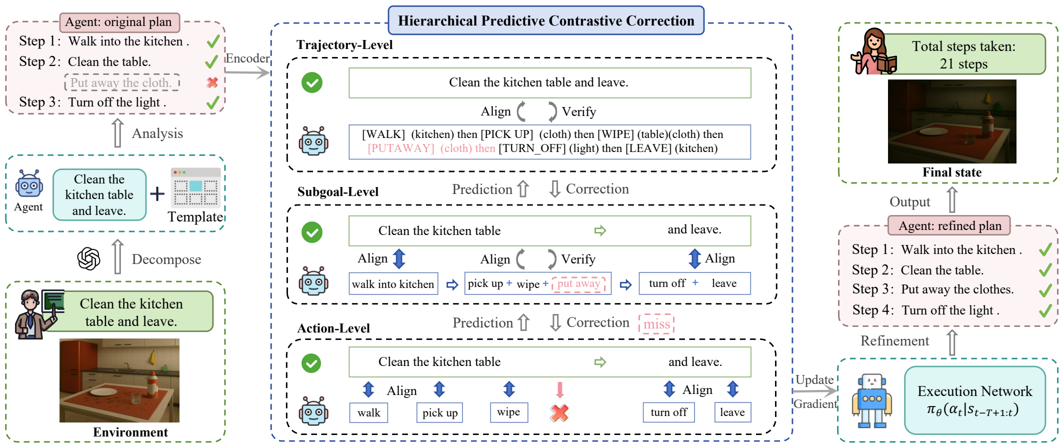
Agent: original plan
Step 1: Walk into the kitchen .
Step 2: Clean the table.
Put away the cloth.
Step 3: Turn off the light .
Analysis
Agent
Clean the kitchen table and leave.
Template
Decompose
Clean the kitchen table and leave.
Environment
Encoder
Hierarchical Predictive Contrastive Correction
Trajectory-Level
Clean the kitchen table and leave.
Align
Verify
[WALK] (kitchen) then [PICK UP] (cloth) then [WIPE] (table)(cloth) then [PUTAWAY] (cloth) then [TURN_OFF] (light) then [LEAVE] (kitchen)
Subgoal-Level
Prediction
Correction
Clean the kitchen table
and leave.
walk into kitchen
pick up + wipe + put away
turn off + leave
Action-Level
miss
walk
pick up
wipe
turn off
leave
Update Gradient
Total steps taken: 21 steps
Final state
Output
Agent: refined plan
Step 1: Walk into the kitchen .
Step 2: Clean the table.
Step 3: Put away the clothes.
Step 4: Turn off the light .
Refinement
Execution Network
$\pi_\theta(a_t|s_{t-T+1:t})$

Figure 1: Overview of the ReCAPA framework. The LLM first generates the agent's original plan through task decomposition. Hierarchical Predictive Contrastive Correction module executes the task and produces fine-grained corrective signals to guide execution updates.

span multiple levels, requiring consistent alignment across steps and overall goal. Yet many methods reflect only at a single layer, inevitably leaving other level propagation unchecked (Sun et al., 2023) (Zhou et al., 2024). To address this, representations are refined during training through higher-level supervision that enforce cross-level consistency alignment. At inference time, the execution network can anticipate deviations early, favor behavior compositions consistent with task intent.

Guided by the above considerations, we propose ReCAPA as shown in Figure 1. ReCAPA seperates trajectories into action-, subgoal-, and trajectory-levels. Unlike prior methods that rely on fixed decomposition and apply only after errors occur, ReCAPA proactively introduces Hierarchical Predictive Contrastive Correction (HPCC) and cross-level alignment signals to prevent errors from compounding. These alignment signals are obtained by comparing trajectory embeddings with the prompt embeddings, using a Sinkhorn-based (Cuturi, 2013) global module and a Score-fieldSong et al. (2021) local module. The Sinkhorn-based module aligns the overall trajectory distribution with the prompt, providing a global indicator at the trajectory-level that guides ReCAPA to reflect on task intent. In parallel, the Score-field module provides step-specific alignment across the remaining two levels. At the subgoal-level, HPCC forecasts the trajectory-level representation while the module evaluates the fit between trajectory rollouts and prediction, updating their parameters when inconsistencies arise. At the action-level, ReCAPA predicts the subgoal representation and uses prompt-based scores to revise fine-grained execution errors. Together, HPCC and prompt-trajectory alignment enable early, cross-level corrections that reduce drift.

To properly assess these benefits, evaluation should go beyond success rate (SR) to also capture how errors propagate, accumulate, and dissipate throughout execution, which existing benchmarks largely overlook. To fill this gap, we introduce two diagnostic metrics in long-horizon reasoning: Error Propagation Rate (EPR) quantifies how mistakes compound across steps, and Propagation Attenuation Coefficient (PAC) measures how errors attenuate or dissipate over time. Together, these metrics capture how failures both spread and decay, providing diagnostic tools to evaluate ReCAPA's stability. Our contributions demonstrate the following advancements:

- We propose ReCAPA, a framework operationalizes hierarchical correction by coupling multi-level predictive representations with prompt–trajectory distributional alignment, allowing deviations to be anticipated and corrected earlier in the rollout.
- We introduce two diagnostic metrics for error propagation in long-horizon reasoning: EPR quantifies the propagation of errors across future steps, while PAC captures the system's ability to recover by measuring how quickly post-error dissipates.
- ReCAPA outperforms strong LMM baselines in terms of success rate, achieving +5.65% on VisualAgentBench, +9% on MineDojo, and +7% on MAP-THOR.

## 2 Related works

**Decomposition Approaches** Task decomposition has been explored in methods such as HIRO (Nachum et al., 2018) which executes fixed interval subgoals and EPO (Zhao et al., 2024) which applies reward modeling for hierarchical planning. LLaMAR (Nayak et al., 2025) and CityNavAgent (Zhang et al., 2025) both follow pre-defined multi-stage subgoal pipelines, but they introduce novel modifications to improve task execution success. LLaMAR improves upon multi-stage task decomposition and policy optimization, while CityNavAgent decomposes subgoals and uses memory of past trajectories to aid planning. However, these static approaches may struggle to adapt when the initial decomposition is flawed, leading to errors in dynamic environments.

**Error-Correction Mechanisms** To address rigid planning, later works introduced feedback-based corrections. ReAct (Yao et al., 2023a), Reflexion (Shinn et al., 2023), and WALL-E (Zhou et al., 2024) provide step-level or episodic updates. AdaPlanner (Sun et al., 2023) revises subplans based on feedback to adapt to changing environments, while R3V (Cheng et al., 2024) self-reflects by generating multiple candidate paths and selecting to improve its trajectories. Although these methods incorporate dynamic error-correction mechanisms and can adapt through feedback from the environment or their own internal processes, they still find it challenging to maintain consistency between the steps and the higher-level intent at different stages of task execution.

**Integration Attempts** To maintain higher-level consistency during corrections, recent research has explored frameworks that integrate decomposition with semantic alignment. For example, TrajPrompt (Tsao et al., 2024) and HiP (Ajay et al., 2023) leverage vision–language information by mapping intermediate task steps into a shared semantic space defined by language and visual features. However, these methods primarily focus on aligning substeps, which can result in correct subgoals but failed actions, making it difficult to maintain consistent alignment between overall intent and fine-grained operations. In contrast, ReCAPA enforces cross-level predictive: lower levels forecast higher-level representations, and deviations trigger top-down corrections that use alignment signals to pull local decisions back to global goals and mitigate error propagation early.

## 3 Methodology

### 3.1 Framework Overview

Most existing methods rely on pre-defined segmentation or post-hoc , which are hard to flexibly correct mistakes and often lead to failures. In addition, local alignment is often insufficient, as it lacks global feedback to correct misordered segments or handle ambiguous cases. To overcome this, we introduce prediction to expose deviations early and correct them. During training, ReCAPA takes trajectory segments, prompt embeddings, and visual observations as input, with Hierarchical Predictive Contrastive (HPCC) representing trajectories at three levels. Prompt–trajectory alignment provides signals to guide local steps to stay aligned with overall task intent. These steps define the training losses, which are backpropagated through the execution network to guide action and object choices.

At inference, ReCAPA uses environmental observations, the prompt, and historical trajectories as inputs. The execution network generates trajectories, while the LLM (GPT-4o-mini) provides task decompositions and completion markers. The three-tier correction mechanism refines the trajectory by resampling actions, adjusting subtasks, and using Sinkhorn for prompt alignment.

### 3.2 Hierarchical Predictive Contrastive

At the core of ReCAPA, HPCC predicts higher-level semantics from lower-level steps and reflects back corrective signals, supportng consistent task representations. HPCC structures reasoning into three levels: actions, which capture how fine-grained steps compose into short-term subgoals (e.g., [GRAB], [WALK], [WIPE] $\rightarrow$ cleaning); subgoals, which forecast trajectory outcomes and enforce causal order (e.g., washing before drying); and trajectories, which encode the task's overall intent and outcome, anchoring prompt–trajectory alignment.

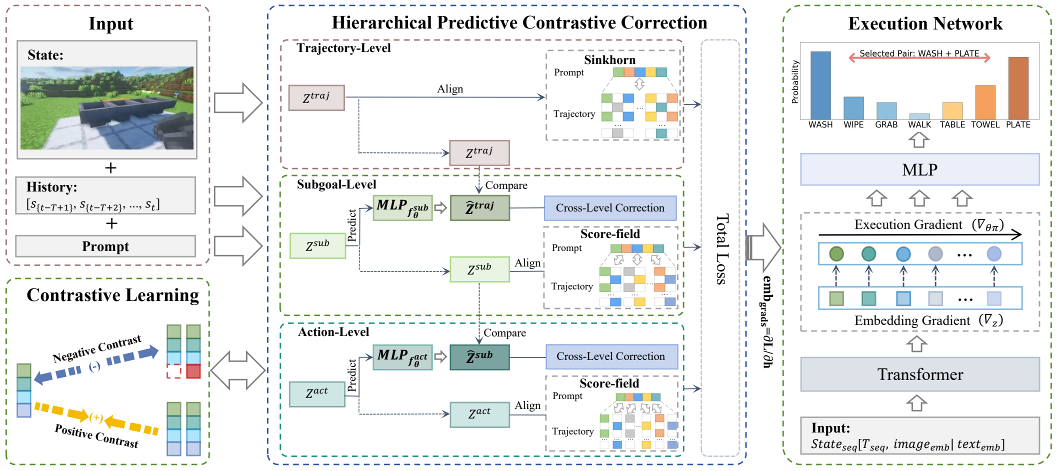

Figure 2: Overview of the ReCAPA training process. In ReCAPA, state, history, and prompt are encoded into hierarchical correction, with predictive and alignment losses guiding the execution network to produce discrete actions.

### 3.2.1 Cross-Level Prediction

HPCC takes the trajectory as input and turns it into multi-level representations at the action, subgoal, and trajectory scales. Fine-grained actions combine to form higher-level subgoals, which together structure the task and encode sequential behavior patterns to anticipate overall task semantics. Specifically, at each level $l \in \{\text{action}, \text{subgoal}\}$, the model predicts the representation at level $l+1$, based on a segment collection $\mathcal{T}^l$. A trajectory segment collection $\mathcal{T}^l$(optionally concatenated with visual embeddings from environment observations) at level $l$ is encoded by a Transformer-based module $E^l_\phi$ into $\mathbf{z}^l$, then processed by an MLP predictor $f^l_\theta$ to yield $\hat{\mathbf{z}}^{l+1} = f^l_\theta(\mathbf{z}^l)$. The predicted representation $\hat{\mathbf{z}}^{l+1}$ is then refined through the corrective mechanisms.

### 3.2.2 Corrective Pathway

Beyond prediction, a corrective pathway is introduced to propagate top–down indicators, allowing higher-level intent to continuously guide lower-level representations and prevent drift. Whenever the cross-level predictor's estimate $\hat{z}^{l+1}$ deviates from the target embedding $z^{l+1}$, the higher-level representation provides a corrective target to regularize the lower-level embedding $z^l$. The target embedding $z^{l+1}$ is obtained from the encoder at level $(l+1)$ and serves as a supervisory reference for the predictor.

Intuitively, lower-level embeddings should remain aligned with the semantic guidance provided by the higher level; otherwise, deviations at finer levels may accumulate and propagate upward. To prevent this drift, the higher-level embedding is used as a supervisory reference to regularize the predicted lower-level representation. Concretely, the cross-level contrastive loss $L^l_{\text{pred}}$ is implemented as an InfoNCE objective on $(\hat{z}^{l+1}, z^{l+1})$, encouraging the predicted representation to stay close to the intended higher-level signal while distinguishing it from distractors. During optimization, gradients are backpropagated only to the level-$l$ encoder, while the level-$(l+1)$ target is detached:

$$L^l_{\text{pred}} = -\log \frac{\exp\left(\text{sim}(\hat{\mathbf{z}}^{l+1}, \mathbf{z}^{l+1})/\tau\right)}{\exp\left(\text{sim}(\hat{\mathbf{z}}^{l+1}, \mathbf{z}^{l+1})/\tau\right) + \sum_j w_j \exp\left(\text{sim}(\hat{\mathbf{z}}^{l+1}, \mathbf{z}^{l+1}_{\text{neg},j})/\tau\right)}, \tag{1}$$

where $\hat{\mathbf{z}}^{l+1}$ represents the predicted output for level $l+1$ as the anchor, $\mathbf{z}^{l+1}$ (computed by $E^{l+1}_\phi$ on $\mathcal{T}^{l+1}$) is the positive sample, and the set $\{\mathbf{z}^{l+1}_{\text{neg},j}\}$ contains negative samples that act as distractors, each associated with a non-negative weight $w_j$ to emphasize ambiguous or challenging examples, while a temperature parameter $\tau > 0$ adjusts similarity sharpness and gradient stability.

### 3.3 Prompt-Trajectory Alignment

We introduce two complementary modules for prompt–trajectory alignment: a global Sinkhorn-based alignment, which leverages Optimal Transport (OT)(Peyré & Cuturi, 2018) to provide distributional consistency between trajectories and prompts, and a local Score-field alignment, which learns corrective gradients to adjust fine-grained actions toward the higher-level direction.

#### 3.3.1 Sinkhorn-based Alignment

The Sinkhorn-based alignment module provides a global objective that aligns entire trajectories with the instruction prompt. It uses a distributional alignment approach, enabling the entire trajectory to flexibly align with the task's semantic structure without requiring exact token-by-token matching, thus handling ambiguous or misaligned segments more effectively. It takes the trajectory distribution $\mu$ and the prompt embedding $\nu$ as input, and outputs a distributional alignment loss. To quantify the distributional discrepancy between the trajectory and the prompt, we employ the entropy-regularized optimal transport distance, leading to the following Sinkhorn divergence Formally, the Sinkhorn divergence is defined as:

$$L_{\text{sinkhorn}}(\mu, \nu) = OT_\epsilon(\mu, \nu) - \tfrac{1}{2} OT_\epsilon(\mu, \mu) - \tfrac{1}{2} OT_\epsilon(\nu, \nu), \tag{2}$$

where $OT_\epsilon$ denotes the entropy-regularized OT cost between distributions. Minimizing $L_{\text{sinkhorn}}$ encourages trajectory embedding to align semantically with the prompt in latent space.

#### 3.3.2 Score-field Alignment

The Score-field module provides a local objective that complements the global Sinkhorn alignment by adding the fine-grained corrections it lacks, pulling deviated actions back toward the global intent. It takes state embeddings $\mathbf{z}^l$ and the prompt embedding $\mathbf{p}$ as input, and outputs localized corrective gradients.

The score network $s_\psi(\mathbf{z}^l, \mathbf{p})$, an MLP, is trained to approximate this field with a denoising objective. Given state–prompt pairs $(\mathbf{z}^l, \mathbf{p})$, the network perturbs $\mathbf{z}^l$ with Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$ and learns to predict the denoising score $-\epsilon/\sigma^2$:

$$L_{\text{score}} = \mathbb{E}_{(\mathbf{z}^l, \mathbf{p}), \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I})} \left[ \| s_\psi(\mathbf{z}^l + \epsilon, \mathbf{p}) - (-\epsilon/\sigma^2) \|_2^2 \right]. \tag{3}$$

It trains $s_\psi$ to model a vector field that points towards high-density regions of the prompt-defined distribution. Consequently, any trajectory state $\mathbf{z}^l$ that lies in a low-density region indicating a deviation from the prompt's semantic intent is assigned a strong gradient by $s_\psi$. This gradient signal, when incorporated into the overall objective, forces deviant state representations to shift towards configurations more consistent with the prompt.

### 3.4 Training and Inference

At the first stage, we pre-train Transformer encoders with a contrastive task on trajectory segments. A window of state–action sequences is encoded and trained with InfoNCE (Eq. 3) using positives from the same trajectory and negatives from GPT-4o-mini. This yields a structured embedding space where each level not only separates valid from invalid patterns but also encodes the plausibility and sequential dependencies of trajectory segments. These structured embeddings form the basis for reliable cross-level alignment and effective corrective feedback.

Then we optimize ReCAPA with a joint objective that combines HPCC's predictive losses and prompt–trajectory alignment losses across hierarchical levels:

$$L_{\text{total}} = \sum_{l \in \{\text{action, subgoal}\}} \left( \lambda^l_{\text{pred}} L^l_{\text{pred}} + \lambda^l_{\text{score}} L^l_{\text{score}} \right) + \lambda_{\text{sinkhorn}} L_{\text{sinkhorn}}. \tag{4}$$

The hyperparameters $\lambda^l_{\text{pred}}$, $\lambda^l_{\text{score}}$, and $\lambda_{\text{sinkhorn}}$ balance the predictive, score-field, and Sinkhorn losses. $L^l_{\text{pred}}$ encourages forecasting of higher-level outcomes from lower-level segments, while $L^l_{\text{score}}$ and $L_{\text{sinkhorn}}$ enforce semantic consistency with the prompt.

During training, we concatenate each visual embedding with the current subgoal text, feed a history window into a Transformer. The joint objective $L_{\text{total}}$ propagates gradients by the chain rule so corrective signals from higher levels reach the execution network. The MLP in execution network outputs discrete logits, which are used to guide action and object choices based on the learned task representation. At inference, LLM (GPT-4o-mini) provides task decompositions and completion markers, serving as supervisory signals that trigger subgoal switching and trajectory-level re-evaluation. ReCAPA applies hierarchical biases: at the action-level, we select Top-K candidates, align their semantics with the subgoal text, re-rank using similarity and logits, and resample if scores are low; for the subgoal-level, a sliding window encodes state–action sequences and computes the similarity between the current and next windows, the LLM guides switching to the better-aligned subtask; at the trajectory-level, Sinkhorn evaluates prompt–trajectory alignment, when misalignment grows, increases the action acceptance threshold to enforce more conservative selection.

### 3.5 Error Propagation Metrics

Standard metrics such as SR or Success weighted by Path Length measure whether a task is eventually completed, but they do not fully capture how errors accumulate or dissipate during execution. In long-horizon reasoning, this distinction is critical: two agents may achieve the same final success rate, yet one suffers from cascading failures while the other recovers from early slips. Without tracking such dynamics, existing metrics can mask important differences in robustness. To address this gap, we introduce two formal measures that characterize error propagation and recovery.

**Error Propagation Rate (EPR).** Let $e_t \in \{0, 1\}$ denote the step-level error, with $t_0$ as the first error time. The EPR at lag $k$ is defined as:

$$\mathrm{EPR}_k = \Pr(e_{t_0+k} = 1 \mid e_{t_0} = 1) - \Pr(e_{t_0+k} = 1 \mid e_{t_0} = 0). \tag{5}$$

For example, $\mathrm{EPR}_3 = 0.4$ means the probability of another error three steps later increases by 40% compared to the case without an initial error. Moreover, with ergodic rollouts and proper conditioning, the estimator $\widehat{\mathrm{EPR}}_k$ is consistent, i.e., $\widehat{\mathrm{EPR}}_k \xrightarrow{p} \mathrm{EPR}_k$ (Appendix E).

**Propagation Attenuation Coefficient (PAC).** PAC is directly defined as:

$$\mathrm{PAC} = -\mathrm{slope}(\Delta,\, \ln \Pr\left(e_{t_0+\Delta} = 1 \,|\, e_{t_0} = 1\right)), \tag{6}$$

which measures the exponential decay rate of post-error risk: larger values indicate quicker recovery, while smaller values reveal that the system remains exposed to error accumulation.

## 4 Experiments and Results

### 4.1 Experimental Setup

To address trajectory drift and long-horizon planning, we evaluate ReCAPA on three benchmarks. MAP-THOR provides 120 interactive scenes while MineDojo is a Minecraft-based benchmark with 3,142 tasks. VisualAgentBench includes OmniGibson (household tasks) and Minecraft (navigation/crafting), measured by Average Success Rate (AVG) and F1. In addition to standard metrics, we report EPR and PAC to quantify error spread and recovery.

In ablations, the baseline w/o HPCC removes the HPCC module, while HPCC-AS (Action+Subgoal), HPCC-AT (Action+Trajectory), and HPCC-ST (Subgoal+Trajectory) use only two-level combinations; HPCC-Full includes all three. PPO replaces HPCC with Proximal Policy Optimization (Schulman et al., 2017), serving as a flat RL baseline. We also implement HIRO with two-level subgoal control, augmented with Sinkhorn and Score-field alignment. The baseline w/o Alignment removes all alignment losses, while Alignment-Full includes both. KL+Score-field replaces Sinkhorn with KL divergence while retaining Score-field.

For HPCC, we set an 8-step horizon at the action-level, adapt the subgoal horizon per benchmark, and treat the full sequence as the trajectory segment during training. At inference, trajectory windows are applied in a sliding manner over recent steps, where each new state–action pair shifts the window forward. With these hierarchical settings in place, ReCAPA adopts a mixed training protocol. On VisualAgentBench and MAP-THOR, we emphasize cross-domain transfer, pre-training on

Table 1: Performance on MAP-THOR(Nayak et al.) across models and metrics. MAP-THOR assessed via Success Rate (SR), Transport Rate (TR), Coverage, and Balance; Coverage measures successful interactions, while Balance captures the evenness of contributions to subtasks.

| **Model** | **SR** | **TR** | **Coverage** | **Balance** |
|---|---|---|---|---|
| **Single-LM/Agent Baselines** | | | | |
| ReAct (Yao et al., 2023b) | 0.34 | 0.72 | 0.92 | 0.67 |
| CoT (Wei et al., 2023) | 0.14 | 0.59 | 0.87 | 0.62 |
| SmartLLM (Kannan et al., 2024) | 0.11 | 0.23 | 0.91 | 0.45 |
| CoELA (Zhang et al., 2023) | 0.25 | 0.46 | 0.76 | 0.73 |
| **Multi-Modal/LLM-Enhanced Baselines** | | | | |
| GPT-4o (Hurst et al., 2024) | 0.51 | 0.85 | 0.95 | 0.83 |
| LLaVA (Liu et al., 2023) | 0.54 | 0.84 | 0.91 | 0.75 |
| IDEFICS-2 (Laurençon et al., 2024) | 0.57 | 0.86 | 0.94 | 0.78 |
| CogVLM (Wang et al., 2024) | 0.61 | 0.89 | 0.95 | 0.80 |
| GPT-4V (Achiam et al., 2023) | 0.66 | 0.91 | **0.97** | 0.82 |
| LLaMAR (Nayak et al., 2025) | 0.68 | 0.90 | 0.95 | 0.85 |
| **ReCAPA** | **0.75** | **0.93** | 0.95 | **0.93** |

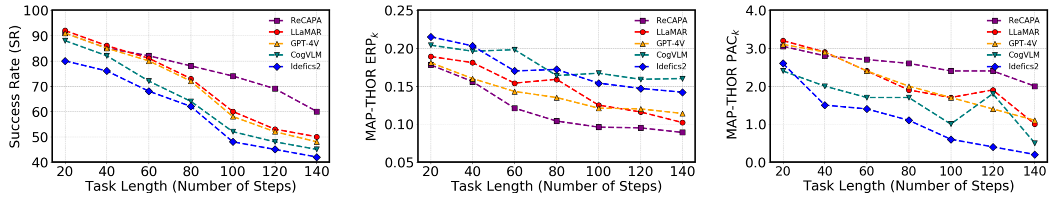

Figure 3: Left: Success rate curves across varying task lengths. Middle: EPR trends showing error propagation at different lags. Right: PAC decay rates on MAP-THOR.

ProcTHOR (Deitke et al., 2022) and Behavior1K (Li et al., 2024a) and directly evaluating without fine-tuning. On MineDojo, we report both zero-shot and fine-tuned results, with fine-tuning adapted on LLM-generated trajectories for three iterations (Iter-3). All baselines follow their original protocols. Further details are in Appendix B.

### 4.2 Results and Analysis

Table 1, 2 and Table 3 summarize results across benchmarks. ReCAPA shows strong multi-task performance with an AVG. score of 58.65, excelling in manipulation and crafting on VisualAgentBench. On MAP-THOR, ReCAPA surpasses baselines with the highest SR of 0.75, TR of 0.93, and Balance of 0.93, though Coverage lags slightly behind GPT-4V. On MineDojo, ReCAPA outperforms prior LLM agents, leading 8/10 long-horizon tasks with higher success rates. It also achieves the lowest EPR$k$ and most favorable PAC trajectory as task length grows; e.g., at $k = 10$ on OmniGibson, $\mathrm{EPR10}$ is 0.082 versus around 0.3 for GPT-4o-mini and Gemini-2.5, and $> 0.45$ for Claude-4-sonnet as shown in Figure 3. Residual errors dissipate fastest for ReCAPA, reflected in the strongest PAC value in Figure 4. Additional comparisons appear in Appendix C.

To further assess the contributions of ReCAPA's core components, we conduct ablation studies targeting the HPCC and prompt-trajectory alignment modules across four different benchmarks as shown in Table 4. Removing HPCC leads to the largest performance drop, with SR on Behavior falling to 59.3 compared to 72.2 for HPCC-Full, confirming the importance of multi-level predictive. HIRO reaches 63.4 on Behavior and 62.7 on VirtualHome, higher than PPO at 60.2 and 60.6, but generally underperforms HPCC variants. Trajectory-level variants show clear gains, with HPCC-AT achieving 0.73 on MAP-THOR and HPCC-ST 0.69, both stronger than HPCC-AS. For alignment, Sinkhorn and Score-field are complementary: Sinkhorn alone gives higher scores than Score-field most of the time, and using both together achieves the best overall performance. KL+Score-field achieves the highest score of 67.0 on the Minecraft task in VisualAgentBench.

Table 2: Performance of different models on VisualAgentBench which include OmniGibson and Minecraft ((Li et al., 2023) (Mojang Studios, 2011). AVG. denotes the overall average score.

| Model | AVG. | OmniGibson | Minecraft |
|---|---|---|---|
| **Open-LMMs (Fine-tuning)** | | | |
| Qwen-VL (Bai et al., 2023) | 9.90 | 1.7 | 18.1 |
| CogVLM2 (Hong et al., 2024) | 13.55 | 6.6 | 20.5 |
| LLaVA-NeXT (Li et al., 2024b) | 16.60 | 9.4 | 23.8 |
| GLM-4V (GLM et al., 2024) | 14.35 | 8.8 | 19.9 |
| InternVL-2 (Chen et al., 2024) | 22.20 | 16.0 | 28.4 |
| **Proprietary-LMMs (Prompting)** | | | |
| qwen-vl-max (Bai et al., 2023) | 2.65 | 0.0 | 5.3 |
| Claude-3.5-Sonnet (Anthropic, 2025b) | 40.15 | 24.3 | 56.0 |
| GPT-4V (preview) (Achiam et al., 2023) | 41.95 | 36.5 | 47.4 |
| GPT-4o (Hurst et al., 2024) | 48.30 | 41.4 | 55.2 |
| Claude-4-Sonnet (Anthropic, 2025a) | 50.25 | 42.6 | 57.9 |
| GPT-4o mini (Zhu et al., 2023) | 54.15 | 46.7 | 61.6 |
| Gemini 2.5 Flash (Comanici et al., 2025) | 53.00 | 43.9 | 62.1 |
| **ReCAPA (Our work)** | **58.65** | **50.6** | **66.7** |

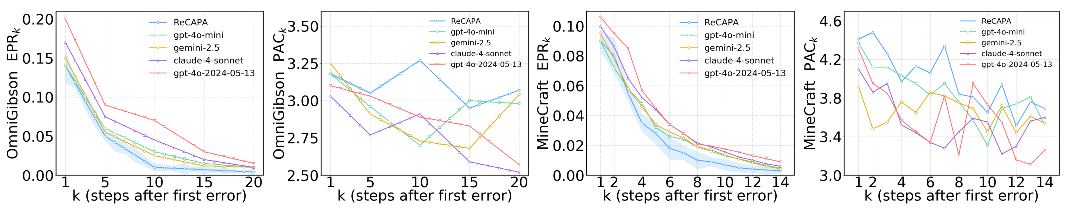

Figure 4: Results on VisualAgentBench. The left two plots show the EPR and PAC curves on OmniGibson, while the right two plots show the EPR and PAC curves on MineCraft. Shaded regions indicate 95% confidence intervals across three random seeds.

## 4.3 Discussion

Our results show that ReCAPA not only achieves higher success rates but also stabilizes execution by suppressing cascading failures and recovering from them. Across benchmarks, it generally outperforms strong proprietary and open-source LMMs. On VisualAgentBench and MineDojo, its advantage is clearest in compositional reasoning tasks, where it decomposes goals into valid subgoals and maintains multi-step consistency. On MAP-THOR, it demonstrates stronger robustness in long-horizon planning and balanced manipulation across diverse scenes. The lower coverage relative to GPT-4V arises because ReCAPA's hierarchical favors structural consistency and high-confidence interactions, while GPT-4V's broader exploration touches more objects. This reflects a fundamental trade-off in embodied agents: broader exploration increases coverage, while consistent enhances stability, and long-horizon reasoning requires balancing both.

Beyond overall success rates, we further analyze error propagation dynamics using our proposed EPR and PAC metrics. ReCAPA achieves the lowest EPR across benchmarks, showing that the impact of early mistakes dissipates more quickly than in other LMMs. This indicates that while errors remain, ReCAPA limits their spread and prevents small deviations from escalating into full failures. Similarly, ReCAPA maintains the highest PAC trajectory, indicating that errors dissipate more rapidly, and longer tasks provide more opportunities for recovery rather than compounding drift. Taken together, the two metrics highlight complementary aspects of robustness, with low EPR reflecting error prevention and high PAC reflecting error recovery, which prior evaluations often overlooked. More broadly, EPR and PAC provide a useful lens for analyzing long-horizon reasoning and may encourage future evaluation to move beyond stepwise accuracy toward explicitly quantifying how agents prevent and dissipate cascading errors.

Table 3: Comparison of several tasks selected from the MineDojo(Fan et al., 2022b) benchmark, covering simple resource gathering and multi-step synthesis or animal interactions. All reported values correspond to SR. The visual encoder is replaced with MINECLIP(Fan et al., 2022a).

| TASK | ／ | ▦ | ⛏ | ■ | 🪣 | □ | 🥩 | 🍖 | 🛏 |
|---|---|---|---|---|---|---|---|---|---|
| MINEAGENT (YU ET AL., 2024) | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | - | - | - | - |
| MINEAGENT (AUTOCRAFT) | 0.00 | 0.03 | 0.00 | 0.00 | 0.46 | 0.50 | 0.33 | 0.35 | 0.00 |
| PLAN4MC (YUAN ET AL., 2023) | 0.30 | 0.30 | 0.53 | 0.37 | 0.83 | 0.53 | 0.43 | 0.33 | 0.17 |
| RL-GPT (ZERO-SHOT) (LIU ET AL., 2024) | 0.26 | 0.30 | 0.53 | 0.47 | 0.79 | 0.53 | 0.43 | 0.35 | 0.30 |
| RL-GPT (ITER-3) | **0.65** | **0.65** | 0.67 | 0.67 | 0.85 | 0.56 | 0.46 | 0.38 | 0.32 |
| RECAPA (ZERO-SHOT) | 0.57 | 0.43 | 0.67 | 0.60 | 0.83 | 0.67 | 0.53 | 0.35 | 0.30 |
| RECAPA (ITER-3) | 0.63 | **0.65** | **0.80** | **0.73** | **0.95** | **0.73** | **0.60** | **0.53** | **0.40** |

Table 4: This ablation study aims to address the role of layers and alignment strategies on four benchmarks (Li et al., 2025) (Shridhar et al., 2021). All reported values correspond to SR.

| **Method** | **EmbodiedAgentInterface** | | **AlfWorld** | **VisualAgentBench** | | **MAP-THOR** |
|---|---|---|---|---|---|---|
| | Behavior | VirtualHome | | OmniGibson | Minecraft | |
| w/o-HPCC | 59.3 | 60.1 | 80 | 42.7 | 56.3 | 0.63 |
| PPO | 60.2 | 60.6 | 79 | 41.5 | 57.8 | 0.59 |
| HIRO | 63.4 | 62.7 | 94 | 44.0 | 60.2 | 0.63 |
| HPCC-AS | 63.6 | 61.4 | 86 | 43.4 | 62.5 | 0.65 |
| HPCC-AT | 65.1 | **70.9** | 94 | 47.9 | 57.5 | 0.73 |
| HPCC-ST | 66.3 | 66.3 | 91 | 48.1 | 60.4 | 0.69 |
| HPCC-Full | **72.2** | 70.5 | **96** | **50.6** | **66.7** | **0.75** |
| w/o-Alignment | 65.8 | 67.2 | 92 | 46.1 | 62.4 | 0.69 |
| Sinkhorn | 66.1 | 69.4 | 95 | 49.3 | 65.6 | 0.74 |
| Score-field | 64.4 | 67.9 | 92 | 46.8 | 66.3 | 0.72 |
| KL + score-field | 70.3 | 68.1 | 95 | 49.6 | **67.0** | 0.74 |
| Alignment-Full | **72.2** | **70.5** | **96** | **50.6** | 66.7 | **0.75** |

In ablation studies, by linking global trajectories with local actions or subgoals, HPCC-AT and HPCC-ST reduce the drift of locally steps from global goals. This suggests that effective long-horizon reasoning requires cross-level guidance, whereas LLMs, though large, often optimize only for local coherence and struggle to maintain consistency over extended horizons. HIRO executes fixed-interval subgoals open-loop, so when the environment shifts it lacks flexible adjustments and actions drift from global goals easily. HPCC instead introduces an adaptive correction strategy, where cross-level feedback adjust local actions and keep them aligned with global goals. For alignment, KL+Score-field's sensitivity strongly penalizes minor mismatches; in Minecraft, where distributions are skewed, this sensitivity helps capture rare but decisive events. However, it also destabilizes signals, making KL-based alignment less reliable than Alignment-Full, which achieves stronger overall performance across most tasks.

## 5 CONCLUSION

To mitigate semantic drift in long-horizon reasoning for embodied agents, we proposed ReCAPA, a framework that integrates hierarchical correction with prompt-trajectory alignment. Experiments on VisualAgentBench, MineDojo and MAP-THOR both demonstrate that ReCAPA outperforms strong baselines. While ReCAPA achieves strong empirical results, it exhibits two key limitations. (1) The correction mechanism operates through discrete scoring at the levels, lacking continuous stepwise feedback to correct deviations during rollout; and (2) the hierarchical generation module employs deterministic mappings for computing next-layer embeddings, which restricts the model to a single trajectory path per subgoal and fails to capture the uncertainties in hierarchical planning. To address these limitations, we propose an uncertainty-gated corrective framework that injects alignment residuals as stepwise signals during rollout, while large residuals trigger diffusion-based branching for multiple plausible continuations. Future work will implement and evaluate these extensions to enhance reasoning and generalization.

## REFERENCES

Joshua Achiam, Samuel Adler, Jacob Ahern, Sandhini Agarwal, Lillian Ahmad, Ilge Akkaya, et al. Gpt-4 technical report. https://arxiv.org/abs/2303.08774, 2023. arXiv:2303.08774.

Anurag Ajay, Seungwook Han, Yilun Du, Shuang Li, Abhi Gupta, Tommi Jaakkola, Josh Tenenbaum, Leslie Kaelbling, Akash Srivastava, and Pulkit Agrawal. Compositional foundation models for hierarchical planning, 2023. URL https://arxiv.org/abs/2309.08587.

Anthropic. Claude 3 model card. https://www.anthropic.com/news/claude-3, 2024. Accessed: 2025-09-03.

Anthropic. Claude 3.5 sonnet. https://www.anthropic.com/news/claude-3-5-sonnet, June 2024a. News release.

Anthropic. The claude 3 model family: Opus, sonnet, haiku. https://www.anthropic.com/index/claude-3-model-card, 2024b. Accessed: 2025-07-24.

Anthropic. Claude sonnet 4. https://www.anthropic.com/claude/sonnet, 2025a. Accessed: 2025-09-06.

Anthropic. Claude 3.5 sonnet. https://www.anthropic.com/news/claude-3-5-sonnet, June 2025b. Accessed: 2025-09-06.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond, 2023. URL https://arxiv.org/abs/2308.12966.

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

Kanzhi Cheng, Yantao Li, Fangzhi Xu, Jianbing Zhang, Hao Zhou, and Yang Liu. Vision-language models can self-improve reasoning via reflection, 2024. URL https://arxiv.org/abs/2411.00855.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. Procthor: Large-scale embodied ai using procedural generation, 2022. URL https://arxiv.org/abs/2206.06994.

Zi-Yi Dou, Cheng-Fu Yang, Xueqing Wu, Kai-Wei Chang, and Nanyun Peng. Re-rest: Reflection-reinforced self-training for language agents. *arXiv preprint arXiv:2406.01495*, 2024.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022a. URL https://openreview.net/forum?id=rc8o_j8I8PX.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge, 2022b. URL https://arxiv.org/abs/2206.08853.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

Wenyi Hong, Weihan Wang, Ming Ding, Wenmeng Yu, Qingsong Lv, Yan Wang, Yean Cheng, Shiyu Huang, Junhui Ji, Zhao Xue, Lei Zhao, Zhuoyi Yang, Xiaotao Gu, Xiaohan Zhang, Guanyu Feng, Da Yin, Zihan Wang, Ji Qi, Xixuan Song, Peng Zhang, Debing Liu, Bin Xu, Juanzi Li, Yuxiao Dong, and Jie Tang. Cogvlm2: Visual language models for image and video understanding, 2024. URL `https://arxiv.org/abs/2408.16500`.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL `https://arxiv.org/abs/2310.06825`.

Shyam Sundar Kannan, Vishnunandan LN Venkatesh, and Byung-Cheol Min. Smart-llm: Smart multi-agent robot task planning using large language models. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 12140–12147. IEEE, 2024.

Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, Matthew Han, Jia Deng, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. What matters when building vision-language models?, 2024. URL `https://arxiv.org/abs/2405.02246`.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, Hang Yin, Michael Lingelbach, Minjune Hwang, Ayano Hiranaka, Sujay Garlanka, Arman Aydin, Sharon Lee, Jiankai Sun, Mona Anvari, Manasi Sharma, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Yunzhu Li, Silvio Savarese, Hyowon Gweon, C. Karen Liu, Jiajun Wu, and Li Fei-Fei. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation, 2024a. URL `https://arxiv.org/abs/2403.09227`.

Feng Li, Renrui Zhang, Hao Zhang, Yuanhan Zhang, Bo Li, Wei Li, Zejun Ma, and Chunyuan Li. Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models, 2024b. URL `https://arxiv.org/abs/2407.07895`.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li Erran Li, Ruohan Zhang, Weiyu Liu, Percy Liang, Li Fei-Fei, Jiayuan Mao, and Jiajun Wu. Embodied agent interface: Benchmarking llms for embodied decision making, 2025. URL `https://arxiv.org/abs/2410.07166`.

Yunzhu Li, Fei Xia, Fanbo Xiang, Roberto Martín-Martín, Michael Lingelbach, Sanjeev Srivastava, Andy Shih, Emily Wong, Roman Shapovalov, Shyamal Buch, et al. Omnigibson: A simulation environment for embodied ai at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 650–651, 2023.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. URL `https://arxiv.org/abs/2304.08485`.

Shaoteng Liu, Haoqi Yuan, Minda Hu, Yanwei Li, Yukang Chen, Shu Liu, Zongqing Lu, and Jiaya Jia. Rl-gpt: Integrating reinforcement learning and code-as-policy, 2024. URL `https://arxiv.org/abs/2402.19299`.

Renjie Lu, Jingke Meng, and Wei-Shi Zheng. Pret: Planning with directed fidelity trajectory for vision and language navigation. *arXiv preprint arXiv:2407.11487*, 2024.

Mojang Studios. Minecraft. `https://www.minecraft.net/`, 2011. Version: Java Edition.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 3303–3313, 2018.

Siddharth Nayak, Adelmo Morrison, Marina Ten Have, Vittal Thirumalai, Jackson Zhang, Darren Chen, Aditya Kapoor, Eric Robinson, Karthik Gopalakrishnan, James Harrison, Anuj Mahajan, Brian Ichter, and Hamsa Balakrishnan. Map-thor: Benchmarking long-horizon multi-agent planning frameworks in partially observable environments. URL `https://api.semanticscholar.org/CorpusID:274523374`.

Siddharth Nayak, Adelmo Morrison Orozco, Marina Ten Have, Vittal Thirumalai, Jackson Zhang, Darren Chen, Aditya Kapoor, Eric Robinson, Karthik Gopalakrishnan, James Harrison, Brian Ichter, Anuj Mahajan, and Hamsa Balakrishnan. Llamar: Long-horizon planning for multi-agent robots in partially observable environments, 2025. URL `https://arxiv.org/abs/2407.10031`.

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2018.

Xavier Puig, Kiana Ehsani Ra, Marko Boben, Jiaman Shen, Shuran Yang, Manolis Savva, Angel X. Chang, Jitendra Malik, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8494–8502, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL `https://arxiv.org/abs/1707.06347`.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL `https://arxiv.org/abs/2303.11366`.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL `https://arxiv.org/abs/2010.03768`.

Yang Song, Jascha Sohl-Dickstein, Durk P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models, 2023. URL `https://arxiv.org/abs/2305.16653`.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Li-Wu Tsao, Hao-Tang Tsui, Yu-Rou Tuan, Pei-Chi Chen, Kuan-Lin Wang, Jhih-Ciang Wu, Hong-Han Shuai, and Wen-Huang Cheng. Trajprompt: Aligning color trajectory with vision-language representations. In *European Conference on Computer Vision*, pp. 275–292. Springer, 2024.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Song XiXuan, et al. Cogvlm: Visual expert for pretrained language models. *Advances in Neural Information Processing Systems*, 37:121475–121499, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL `https://arxiv.org/abs/2201.11903`.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023b. URL https://arxiv.org/abs/2210.03629.

Beibei Yu, Tao Shen, Hongbin Na, Ling Chen, and Denqi Li. Mineagent: Towards remote-sensing mineral exploration with multimodal large language models, 2024. URL https://arxiv.org/abs/2412.17339.

Haoqi Yuan, Chi Zhang, Hongcheng Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu. Skill reinforcement learning and planning for open-world long-horizon tasks, 2023. URL https://arxiv.org/abs/2303.16563.

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 preprint arXiv:2307.02485*, 2023.

Weichen Zhang, Chen Gao, Shiquan Yu, Ruiying Peng, Baining Zhao, Qian Zhang, Jinqiang Cui, Xinlei Chen, and Yong Li. Citynavagent: Aerial vision-and-language navigation with hierarchical semantic planning and global memory, 2025. URL https://arxiv.org/abs/2505.05622.

Qi Zhao, Haotian Fu, Chen Sun, and George Konidaris. Epo: Hierarchical llm agents with environment preference optimization, 2024. URL https://arxiv.org/abs/2408.16090.

Tianyang Zhong, Zhengliang Liu, Yi Pan, Yutong Zhang, Yifan Zhou, Shizhe Liang, Zihao Wu, Yanjun Lyu, Peng Shu, Xiaowei Yu, et al. Evaluation of openai o1: Opportunities and challenges of agi. *arXiv preprint arXiv:2409.18486*, 2024.

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, 2024. URL https://arxiv.org/abs/2410.07484.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models, 2023. URL https://arxiv.org/abs/2304.10592.

Yifeng Zhu, Jonathan Tremblay, Stan Birchfield, and Yuke Zhu. Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs, 2021. URL https://arxiv.org/abs/2012.07277.

## A LLM Declaration

The LLM was employed solely to polish wording and improve readability. It was not used for idea generation, methodological development, or result interpretation, and all scientific contributions were made independently by the authors.

## B Detailed Experimental Procedures

ReCAPA employs a two-stage training process. The first stage is offline pre-training on diverse expert and LLM-generated trajectories to establish hierarchical predictive and alignment capabilities. In the second stage, we adopt benchmark-specific protocols: some benchmarks (e.g., MineDojo) involve supervised adaptation using in-domain trajectories, while others (e.g., VisualAgentBench and MAP-THOR) emphasize pure cross-domain transfer without task-specific fine-tuning.

### B.1 Stage 1: Offline Pre-training

The offline pre-training phase leverages expert demonstration trajectories to establish a foundational understanding of embodied interaction and planning for ReCAPA. This phase is critical for equipping the model with the necessary knowledge to generalize across tasks before any domain-specific fine-tuning is applied. We utilize two distinct datasets, `BEHAVIOR-1K` and `ProcTHOR`, to cover a broad range of scenarios and ensure diverse task representation.

- **BEHAVIOR-1K:** This benchmark focuses on complex, everyday human activities in simulated environments. For this dataset, we select 300 representative tasks and use 3 to 5 expert demonstration trajectories per task. The fine-grained interaction data allows the model to learn the dynamics of typical human behavior in an embodied context.

- **ProcTHOR:** In addition to `BEHAVIOR-1K`, we incorporate 300 diverse tasks from the procedurally generated `ProcTHOR` environment. This dataset features various scene layouts and object arrangements, and we collect approximately 10 expert trajectories for each task. The diversity of this dataset ensures the model is not overly dependent on specific environmental configurations, enhancing its generalization ability.

This foundational pre-training step is conducted in an offline manner, allowing the model to absorb critical information about trajectories without the complexity of real-time interaction. Although the total number of trajectories is smaller than in large-scale pretraining corpora, the diversity across BEHAVIOR-1K and ProcTHOR ensures broad task coverage and equips ReCAPA with transferable knowledge of embodied dynamics. This design emphasizes sample efficiency and cross-domain generalization: in Stage 2, the model is evaluated both with limited in-domain adaptation (e.g., MineDojo) and under pure transfer settings (e.g., VisualAgentBench and MAP-THOR), highlighting ReCAPA's ability to generalize beyond its training distribution.

### B.2 Stage 2: Domain-Specific Adaptation and Transfer

Following the pre-training phase, we distinguish two types of evaluation protocols. For MineDojo and AlfWorld, ReCAPA is adapted on LLM-generated trajectories from the target domain for several iterations, each iteration consisting of supervised updates on new rollouts with the hierarchical predictive and alignment objectives. Hard negatives are re-sampled at every round, and explicit gradient mapping ensures that corrections at each level remain disentangled.

For VisualAgentBench, MAP-THOR and EmbodiedAgentInterface, we emphasize pure cross-domain transfer: the model is pre-trained on ProcTHOR and BEHAVIOR-1K and directly evaluated on the target benchmarks without task-specific fine-tuning. This setting highlights ReCAPA's generalization ability under strict transfer conditions. Baselines are evaluated under their original protocols to ensure fair comparison.

Table 5: PAC vs. Task Length: Measures the attenuation rate of the impact of early errors on subsequent steps (the higher the value, the faster the recovery and the weaker the error propagation)

| **Model** | **20** | **40** | **60** | **80** | **100** | **120** | **140** |
|---|---|---|---|---|---|---|---|
| **ReCAPA** (Our work) | 3.1 | 2.8 | 2.7 | 2.6 | 2.4 | 2.2 | 2.0 |
| **LLaMAR** (Nayak et al., 2025) | 3.2 | 2.9 | 2.4 | 1.9 | 1.7 | 1.4 | 1.0 |
| **GPT-4V** (Achiam et al., 2023) | 3.1 | 2.9 | 2.4 | 2.0 | 1.7 | 1.4 | 1.1 |
| **CogVLM** (Wang et al., 2024) | 2.4 | 2.0 | 1.7 | 1.4 | 1.0 | 0.8 | 0.5 |
| **IDEFICS-2** (Laurençon et al., 2024) | 2.6 | 1.5 | 1.2 | 1.1 | 0.6 | 0.3 | 0.0 |

Table 6: Performance of ReCAPA Variants under Layer-wise Ablation across Benchmarks. This table estimates the expected impact of removing individual layers or flattening the hierarchy (FlatHead) on ReCAPA's performance across diverse environments, highlighting the necessity of each abstraction level

| **Method** | **EmbodiedAgentInterface** | | **AlfWorld** | **VisualAgentBench** | | **MAP-THOR** |
|---|---|---|---|---|---|---|
| | **Behavior** | **VirtualHome** | | **OmniGibson** | **Minecraft** | |
| w/o Subgoal-Level | 62.8 | 61.5 | 87 | 44.8 | 61.4 | 0.62 |
| w/o Trajectory-Level | 65.4 | 64.5 | 86 | 47.4 | 60.5 | 0.67 |
| FlatHead | 49.3 | 55.4 | 78 | 35.6 | 44.1 | 0.52 |
| ReCAPA | **72.2** | **70.5** | **96** | **50.6** | **66.7** | **0.75** |

## B.3 Training Configuration

For all environments, we use the AdamW optimizer, with a learning rate of 1e-4, a weight decay of 0.01, and a batch size of 32. The learning rate follows a cosine schedule with a linear warm-up over the first 1,000 steps. Training is conducted for a total of 200,000 steps, with the model being trained on 4 NVIDIA H20 GPUs to ensure efficient scaling across large datasets. The text encoder (nomic-embed-text-v1.5) encodes prompt tokens, while the vision encoder (nomic-embed-vision-v1.5) encodes environmental observations.

In summary, the two-stage training protocol—comprising offline pre-training and online fine-tuning—equips ReCAPA with the capacity to handle complex, long-horizon tasks in varied environments. The incorporation of expert trajectories during pre-training and the dynamic, memory-augmented fine-tuning mechanism ensures that ReCAPA continuously improves its performance and generalizes effectively to unseen scenarios. As shown in Table 5, it presents the PAC results on MAP-THOR, showing how different models recover from early errors across varying task lengths.

To assess the independent contribution of each hierarchical level in ReCAPA's two-level structure (subgoal $\rightarrow$ trajectory), we conduct a layer-wise ablation study. While the full model integrates across all levels, it remains unclear whether each level provides unique benefits. In this experiment, we selectively remove one module at a time—trajectory-level or subgoal-level, keeping the rest of the architecture intact. This design allows us to evaluate whether each layer offers distinct abstraction or semantic supervision, whether model performance depends on cross-scale reasoning, and whether the joint use of all levels outperforms partial configurations. In addition, we introduce a FlatHead variant, which collapses the entire hierarchy into a single decoder without explicit levels, to test whether the hierarchical structure itself is essential for semantic reasoning.

Table 7: Comparative Evaluation of ReCAPA and Hierarchical Variants: Effects of Coupling and Alignment Strategy. ReCAPA integrates all levels via joint and multi-scale alignment, allowing semantic corrections to propagate throughout the hierarchy

| Method | EmbodiedAgentInterface | | AlfWorld | VisualAgentBench | | MAP-THOR |
|---|---|---|---|---|---|---|
| | Behavior | VirtualHome | | OmniGibson | Minecraft | |
| Separate–BottomUp | 62.1 | 62.7 | 79 | 43.9 | 57.9 | 0.59 |
| Separate–Parallel | 63.4 | 60.8 | 76 | 45.4 | 60.5 | 0.54 |
| Separate–TopDown | 66.7 | 64.5 | 86 | 47.4 | 62.5 | 0.67 |
| Frozen Traj-Level | 68.9 | 68.0 | 90 | 48.3 | 63.9 | 0.71 |
| ReCAPA | **72.2** | **70.5** | **96** | **50.6** | **66.7** | **0.75** |

## C Complementary Results and Analysis

### C.1 Layer Ablation

The ablation results in Table 6 highlight the necessity of maintaining a multi-level structure in ReCAPA. Removing any individual layer leads to a consistent drop in performance across all benchmarks, confirming that each level contributes uniquely to hierarchical reasoning. Notably, the removal of the mid-level results in the most severe degradation, especially on Behavior (–9.4 points) and VirtualHome (–9.0), suggesting that the mid-level layer plays a critical role in bridging trajectory-level plans with low-level execution. The w/o Trajectory-Level variant also suffers substantial loss in ALFWorld and MAP-THOR, demonstrating that long-horizon environments with sparse rewards and delayed feedback heavily depend on long-range planning. The FlatHead baseline, which removes the entire hierarchy, performs the worst across all benchmarks—underscoring the indispensable value of structured abstraction and layered semantic supervision for complex task generalization. These findings validate ReCAPA's core design principle: performance in long-horizon embodied tasks emerges from coordinated across abstraction levels.

### C.2 Coupling Strategy Comparison

To further examine the role of joint optimization across hierarchical layers, we introduce several additional ablation variants targeting the training strategy of HCPR:

- **Frozen Traj-Level**: The Traj-level module remains active during forward execution but its parameters are frozen throughout training. Only the mid- and low-level modules are updated, isolating the contribution of Traj-level gradient signals.
- **Separate–BottomUp**: Each layer is trained sequentially in a bottom-up manner: the action-level module is trained first, then frozen; the subgoal-level is trained next with action-level frozen; finally, the trajectory-level module is trained on top. This mimics a stage-wise curriculum from action primitives to subgoal planning.
- **Separate–TopDown**: The reverse of BottomUp. Training proceeds from the trajectory-level module down to the action-level, with each previously trained module frozen at its respective stage. This configuration reflects top-down reasoning pipelines, starting from global goals to execution-level commands.
- **Separate–Parallel**: All three layers are trained independently without inter-layer gradient flow. Each module is optimized on its respective sub-task using its local alignment and trajectory data. This configuration serves as a strong baseline to test whether cross-layer interactions are necessary.

Each of these configurations is trained under the same data regime and alignment loss structure (Sinkhorn + Score Field), allowing for controlled comparisons with the jointly optimized ReCAPA model. We report task-level metrics to evaluate performance degradation and convergence stability.

Table 8: This ablation study evaluates the impact of prediction and components in HPCC by isolating the effect of removing prediction, , or both.

| Variant | EmbodiedAgentInterface | | AlfWorld | VisualAgentBench | | MAP-THOR |
|---|---|---|---|---|---|---|
| | Behavior | VirtualHome | | OmniGibson | Minecraft | |
| w/o pred and refl | 62.6 | 63.1 | 85 | 44.5 | 60.5 | 0.66 |
| Only Prediction (no refl) | 65.4 | 66.3 | 90 | 47.8 | 60.9 | 0.72 |
| Only (no pred) | 67.1 | 64.9 | 91 | 49.2 | 62.0 | 0.72 |
| Full HPCC (pred + refl) | 72.2 | 70.5 | 96 | 50.6 | 66.7 | 0.75 |

Table 7 presents a comparative study of different hierarchical training strategies across six embodied benchmarks. ReCAPA achieves the highest performance on all tasks, demonstrating the benefit of fully joint training across three corrective modules. In contrast, all decoupled baselines underperform to varying degrees, each revealing critical weaknesses in alternative optimization schemes.

The Separate–BottomUp configuration performs the worst overall, as it trains low-level modules in isolation before exposing them to task objectives. This leads to suboptimal primitive behaviors that constrain downstream learning and confirms that without task-aware supervision results in inefficient or misaligned action policies. The Separate–TopDown baseline performs moderately better , but still lags behind ReCAPA. Despite training from task goals downward, the lack of feedback from lower layers causes the top-level planner to overfit to idealized subgoal sequences that may not align with actual execution capabilities—resulting in a "planning-execution mismatch." The Separate–Parallel setting confirms this issue from another angle: although each module becomes competent in isolation, the lack of cross-layer adaptation leads to representational inconsistency and semantic drift between layers. The resulting interface mismatch limits coordination across hierarchical stages. Frozen Traj-Level, which freezes the top-level module and only updates the mid- and low-level components, yields decent performance, but falls short of ReCAPA. This highlights that trajectory-level goal representations also require continual adaptation to downstream dynamics in order to maintain semantic coherence.

Taken together, these results empirically validate our theoretical claim that hierarchical must be jointly optimized to achieve sample-efficient and semantically aligned behavior. The observed performance gaps support our convergence analysis under multi-objective optimization and further justify the design of ReCAPA's multi-level update strategy.

### C.3 Ablation on Prediction and Alignment

To further disentangle the respective contributions of prediction and within HPCC, as shown in Table 8 we designed four ablation variants. The first, w/o pred and refl, removes both components entirely, leaving only the hierarchical execution head to generate actions and serving as a baseline without any auxiliary consistency signals. The second, Only Prediction, retains the forward prediction modules that forecast higher-level trajectory or subgoal embeddings to regularize lower-level execution, but discards the modules that would otherwise check and realign actions during rollouts; this setting isolates the benefit of anticipatory guidance without any corrective feedback. The third, Only , removes prediction while preserving the consistency checks and corrective updates after each execution step, thereby examining the effect of purely reactive recovery in the absence of foresight. Finally, Full HPCC activates both prediction and to form a closed loop in which lower-level policies both anticipate higher-level representations and immediately repair inconsistencies as they arise. Comparing these four settings allows us to characterize the distinct roles of prediction (proactive prevention) and (reactive correction), as well as their synergy when combined.

This ablation highlights the distinct yet complementary roles of prediction and within HPCC. Eliminating both modules severely degrades performance, confirming that hierarchical execution heads alone are insufficient for mitigating long-horizon drift. When only prediction is retained, the agent benefits from anticipatory alignment signals that reduce short-horizon inconsistencies (e.g., improved results on VisualAgentBench), but the absence of reactive correction allows early errors to

Table 9: This ablation study evaluates the impact of gradient-based updates on execution learning across benchmarks by isolating the effect of the update mechanism that transforms semantic alignment signals into trajectory-aware execution adjustments

| **Method** | **EmbodiedAgentInterface** | | **AlfWorld** | **VisualAgentBench** | | **MAP-THOR** |
|---|---|---|---|---|---|---|
| | **Behavior** | **VirtualHome** | | **OmniGibson** | **Minecraft** | |
| w/o Advantage Weighting | 68.5 | 66.3 | 91 | 46.7 | 62.6 | 0.70 |
| w/o Execution Update | 67.1 | 60.2 | 88 | 41.9 | 58.5 | 0.66 |
| w/o Execution Network | 65.8 | 64.7 | 56 | 49.1 | 64.5 | 0.60 |
| ReCAPA | **72.2** | **70.5** | **96** | **50.6** | **66.7** | **0.75** |

cascade unchecked, resulting in limited gains on AlfWorld and MAP-THOR. Conversely, keeping only provides the ability to recover after errors occur, which is particularly beneficial for tasks with extended horizons where error accumulation dominates; however, without forward prediction, the system lacks proactive guidance and remains vulnerable to subtle misalignments in representation space. Full HPCC consistently outperforms all ablated variants because it unifies both mechanisms: prediction serves as an early warning system that lowers the likelihood of compounding failures, while functions as a recovery channel that actively attenuates error propagation.

## C.4 Gradient-based Execution Update Ablation

To evaluate the impact of the execution update module, we introduce two ablation variants:

- **w/o Advantage Weighting**: Removes the advantage weighting term Adv$(z_l)$ in the execution gradient update. The execution is updated using a standard unweighted log-likelihood objective:
$$\nabla_\theta J(\theta) = \mathbb{E}_{(s,a)\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)]$$
This variant still leverages the full objective $\mathcal{L}_{\text{total}}$ for modules but removes its influence on scaling.
- **w/o Execution Update**: Completely disables gradient-based Execution updates from $\mathcal{L}_{\text{total}}$. The execution network is trained solely via behavioral cloning from expert demonstrations, without corrective gradients. Only the alignment and modules receive updates from $\mathcal{L}_{\text{total}}$.
- **w/o Execution Network**: This baseline completely removes the execution network, using only the three-tier prompt-guided mechanism with GPT-4o-mini. No training or updates are performed on the execution network, and the system relies solely on the pre-defined prompt and modules to guide task completion. The modules do not perform dynamic self-correction or optimization, but instead rely on static task decomposition and goal alignment, with no execution adjustments or gradient updates.

Both variants isolate the contribution of execution update to long-horizon adaptation by examining whether gradient-aligned updates improve task consistency and execution fidelity.

As shown in Table 9 , both ablated variants suffer consistent performance degradation compared to the full ReCAPA model. The removal of advantage weighting (*w/o Advantage Weighting*) leads to moderate drops (2–4 points) across all benchmarks, highlighting that trajectory-level semantic gradients—while still present—are less effective without contextual scaling. The more severe drop in *w/o Execution Update* underscores the critical role of execution update: completely removing gradient-based updates from $\mathcal{L}_{\text{total}}$ results in sharp declines, especially on long-horizon tasks such as MAP-THOR (–0.09) and Minecraft (–8.2), where corrective updates are essential for resolving temporal credit assignment. The "w/o Execution Network" variant experiences notable performance drops compared to the full ReCAPA model across all benchmarks. For instance, in VisualAgent-Bench, the drop is 10.6 points, and in MAP-THOR, it's 0.15.

Interestingly, the gap between the three variants is most evident in environments that require reasoning over extended plans (e.g., AlfWorld and VisualAgentBench). This supports our design intuition:

Table 10: Results on the Behavior task of the Embodied Agent Interface benchmark. ReCAPA shows strong performance with leading scores in Goal F1 (84.8), Action Sequencing (77.0/84.0), and Subgoal Decomposition (53.0/60.0), surpassing baselines such as o1-preview and Claude-4. These results highlight its effectiveness in long-horizon reasoning and precise subgoal planning through hierarchical architecture and alignment mechanisms

| **Models** | **Perf.** | **Goal F1** | **Action Seq.** | | **Subgoal Dec.** | | **F1** |
|---|---|---|---|---|---|---|---|
| | | | *Task* | *Exec.* | *Task* | *Exec.* | |
| **o1-preview** Zhong et al. (2024) | **74.9** | 81.6 | **81.0** | **91.0** | **57.0** | **62.0** | **70.8** |
| ReCAPA | 72.2 | **84.8** | 77.0 | 84.0 | 53.0 | 60.0 | 69.8 |
| Claude-4-Sonnet Anthropic (2024) | 68.5 | 84.3 | 68.0 | 75.0 | 47.0 | 52.0 | 69.2 |
| Claude-3.5-Sonnet Anthropic (2024a) | 64.2 | 82.7 | 60.0 | 69.0 | 39.0 | 44.0 | 67.9 |
| Claude-3-Opus Anthropic (2024b) | 60.4 | 77.0 | 51.0 | 59.0 | 41.0 | 47.0 | 63.4 |
| GPT-4o Hurst et al. (2024) | 59.8 | 79.2 | 47.0 | 53.0 | 49.0 | 55.0 | 60.9 |
| o1-mini Jaech et al. (2024) | 57.5 | 76.4 | 56.0 | 65.0 | 31.0 | 39.0 | 56.4 |
| Claude-3-Sonnet Anthropic (2024b) | 55.1 | 69.4 | 44.0 | 57.0 | 39.0 | 43.0 | 56.2 |
| Gemini-1.5-Flash Team et al. (2024) | 52.1 | 74.8 | 40.0 | 52.0 | 34.0 | 42.0 | 53.4 |
| Mistral-Large Jiang et al. (2023) | 50.4 | 74.3 | 33.0 | 50.0 | 31.0 | 38.0 | 49.5 |

the execution update module not only injects semantically aligned feedback but also acts as a bridge between the abstract modules and the concrete action policies. The absence of the execution network in the ablation study further emphasizes the importance of the network in dynamically refining policies. Without the execution network, the system lacks the ability to continuously adapt its actions based on the feedback from the modules, limiting its capacity for long-horizon adaptation. These results confirm that gradient-aligned feedback is necessary for robust execution refinement, particularly in environments requiring extended planning and continuous execution updates.

### C.5 Evaluation on Embodied Agent Interface

As show in Table 10, it evaluates ReCAPA and competing models on the Embodied Agent Interface benchmark, focusing on Behavior and VirtualHome tasks. The metrics include Performance (Perf.), Goal F1, Action Sequence accuracy (both Task and Execution levels), and Subgoal Decomposition (Task and Execution), with an overall F1 score. ReCAPA achieves robust performance (72.2 Perf., 84.8 Goal F1), trailing only o1-preview in Perf. but excelling in Goal F1. The training methodology for these tasks follows a two-stage protocol: offline pre-training on state-action-reward trajectories initialized via GPT-4o API, followed by benchmark-specific optimization. For Behavior tasks, ReCAPA leverages ProcTHOR's procedurally generated environments (30 tasks, 10 expert trajectories per task) to enhance scene generalization, avoiding overfitting. The model is trained end-to-end with AdamW (lr=1e-4, weight decay=0.01, batch size=32) for 200K steps on 4 NVIDIA H20 GPUs, using a cosine learning rate schedule with 1,000-step warm-up. We obtain base text and vision embeddings from pre-trained Nomic encoders (nomic-embed-text-v1.5 and nomic-embed-vision-v1.5), which remain frozen; only the HPCC and alignment modules are optimized. This approach ensures strong performance in both trajectory-level goal reasoning (84.8 F1) and action sequencing (77.0 Task, 84.0 Exec.), as reflected in the results.

As show in Table 11, it provides a comprehensive evaluation across six key metrics: Overall Performance, F1, Action Sequencing (Task SR, Exec. SR), Subgoal Decomposition (F1), and Transition Modeling (Plan SR). ReCAPA outperforms competitors in Overall Perf. (70.5) and Transition Modeling (84.6 Plan SR), demonstrating its strength in long-horizon planning and trajectory stability. The training pipeline mirrors the methodology for MAP-THOR and VisualAgentBench, emphasizing cross-domain transfer through few-shot pre-training on ProcTHOR (no BEHAVIOR-1K in this phase) and task-specific fine-tuning. For AlfWorld tasks, ReCAPA employs RAFA-style multi-round interactions with memory-augmented prompts to improve online adaptation. The technical setup matches Table 6's (AdamW, 200K steps), with alignment losses (Full Alignment) and hierar-

Table 11: Results on the VirtualHome task of the Embodied Agent Interface benchmark. ReCAPA shows consistently strong performance across six metrics, leading in Overall Perf. (70.5), Subgoal Decomposition (Task SR: 94.5, Exec. SR: 91.7), and Transition Modeling (F1: 64.9, Plan SR: 84.6). Compared to o1-preview and Claude models, it demonstrates superior hierarchical reasoning and robust embodied planning in complex VirtualHome environments

| **Model Family** | **Overall Perf.** | **F1** | **Action Sequencing** | **Subgoal Decomposition** | | **Transition Modeling** | |
|---|---|---|---|---|---|---|---|
| | | | *Task SR* | *Task SR* | *Exec. SR* | *F1* | *Plan SR* |
| ReCAPA | **70.5** | **44.2** | 72.8 | **94.5** | **91.7** | **64.9** | 84.6 |
| Claude-4-Sonnet Anthropic (2024) | 69.1 | 43.5 | **73.6** | 92.7 | 90.9 | 51.4 | 86.8 |
| o1-preview Zhong et al. (2024) | 65.8 | 42.7 | 71.1 | 93.2 | 89.4 | 48.0 | 72.4 |
| Gemini-Pro Team et al. (2024) | 65.3 | 37.9 | 73.1 | 91.1 | 87.0 | 34.1 | **91.9** |
| Claude-3-Sonnet Anthropic (2024b) | 64.9 | 33.0 | 72.8 | 92.0 | 89.1 | 48.9 | 80.5 |
| GPT-4o Hurst et al. (2024) | 60.8 | 36.5 | 61.6 | 91.1 | 87.6 | 46.7 | 68.2 |
| Claude-3-Opus Anthropic (2024b) | 59.9 | 31.4 | 66.2 | 89.9 | 86.7 | 48.8 | 61.8 |
| o1-mini Jaech et al. (2024) | 57.9 | 31.2 | 65.9 | 84.6 | 79.3 | 41.5 | 69.0 |

Table 12: Models performance on AlfWorld benchmark

| **Method** | **Success rate (%) $\uparrow$** | | | | | | |
|---|---|---|---|---|---|---|---|
| | Avg. | Pick | Clean | Heat | Cool | Examine | Picktwo |
| **Vision-language models** | | | | | | | |
| MiniGPT-4* | 16 | 4 | 0 | 19 | 17 | 67 | 6 |
| BLIP-2* | 4 | 0 | 6 | 4 | 11 | 6 | 0 |
| LLaMA-Adapter* | 13 | 17 | 10 | 27 | 22 | 0 | 0 |
| InstructBLIP* | 22 | 50 | 26 | 23 | 6 | 17 | 0 |
| EMMA* | **82** | **71** | **94** | **85** | **83** | **88** | **67** |
| **Language models** | | | | | | | |
| BUTLER* | 26 | 31 | 41 | 60 | 27 | 12 | 29 |
| DEPS | 76 | 93 | 50 | 80 | **100** | **100** | 0 |
| AutoGen* | 77 | 92 | 74 | 78 | 86 | 83 | 41 |
| ReAct | 74 | 79 | 54 | 96 | 85 | 83 | 51 |
| AdaPlanner | 91 | 100 | **100** | 89 | **100** | 97 | 47 |
| Reflexion | 86 | 92 | 94 | 70 | 81 | 90 | 88 |
| RAFA | 95 | 100 | 97 | 91 | 95 | **100** | 82 |
| WALL-E 1.0 | 95 | 100 | 97 | **100** | 86 | 85 | **100** |
| WALL-E 2.0 | **98** | 100 | **100** | 96 | **100** | 100 | 94 |
| **ReCAPA (ours)** | **96** | **100** | **97** | 94 | 95 | 96 | 94 |

chical (HPCC-Full) critical to its success in Subgoal Decomposition (64.9 F1) and Action Sequencing (94.5 Exec. SR). Claude-4-Sonnet and Gemini-Pro show competitive Plan SR (86.8, 91.9), but ReCAPA balances all metrics, underscoring its versatility.

### C.6 Performance analysis and discussion

From the results in Table 12, our model ReCAPA consistently demonstrates clear advantages over both proprietary and open-source LMM baselines. In particular, ReCAPA shows strong performance on long-horizon tasks such as PICKTWO, where success rates remain significantly higher than all other baselines. This confirms that our hierarchical planning and mechanisms can effectively manage compositional goals that require multiple coordinated steps.

Despite its overall superiority, ReCAPA is not without limitations. WALL-E 2.0—augmented with reinforcement learning fine-tuning—achieves slightly higher scores on some subtasks. This gap reflects a trade-off: ReCAPA is primarily designed for sparse or reward-agnostic settings, which limits

its ability to exploit dense feedback signals as efficiently as reinforcement learning–enhanced baselines. Overall, the results demonstrate that ReCAPA offers a balanced and generalizable solution: it consistently outperforms competitors in challenging long-horizon tasks, while only marginally trailing in AlfWorld where dense rewards provide a strong supervision signal.

## D Failure Case Analysis

To complement the quantitative results (EPR, PAC, and ablations), we provide qualitative analyses of typical failure modes. As shown in Table 13, these examples illustrate how cascading errors arise and how ReCAPA mitigates them compared to ablated variants.

We categorize common failure types observed during evaluation:

| **Failure Type** | **Description / Example** |
|---|---|
| Subgoal Ordering Error | Executing subgoals in the wrong order (e.g., attempting to close the fridge before retrieving the milk). This leads to invalid or incomplete task outcomes. |
| Entity Grounding Error | Misidentifying or manipulating the wrong object (e.g., taking juice instead of the intended milk). |
| Premature Termination | Ending the task early before completing all required subgoals (e.g., retrieving the milk but leaving the fridge door open). |
| Looping / Redundancy | Performing unnecessary actions without contributing to the goal (e.g., taking an extra apple after already retrieving the milk). |

Table 13: Taxonomy of common failure types in hierarchical agent execution, adapted to the fridge–milk task with descriptions and examples.

- **Prompt:** “Take the milk from the fridge.”
- **Full ReCAPA:** Executes correctly by [Open fridge] → [Take milk] → [Close fridge].
- **w/o HPCC:** Retrieves the milk but then performs an unnecessary action [Take apple] and forgets to [Close fridge], leading to redundancy and premature termination.

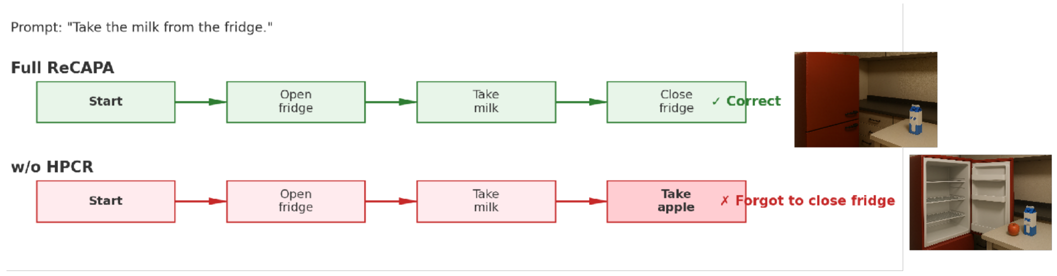

Figure 5: Representative failure case for the prompt “Take the milk from the fridge.” Full ReCAPA executes correctly by opening the fridge, retrieving the milk, and closing the fridge. In contrast, the ablated model (w/o HPCC) retrieves the milk but then takes an unrelated item and forgets to close the fridge, leaving the environment in an invalid state and illustrating how local missteps cascade into task failure.

As shown in Figure 5, execution begins correctly as the agent approaches the pot and moves toward the sink. The action-level representation fails to predict the correct subgoal-level semantic target (“place the pot into the sink”), causing action–subgoal alignment to drop. Subgoal-level forecasting then drifts toward an incorrect direction, which subsequently causes global prompt–trajectory alignment to decrease.

ReCAPA detects this cross-level inconsistency and resamples the action until the pot is correctly placed in the sink. Once the corrected action is taken, the restored alignment propagates upward: subgoal-level semantics realign with the intended target, and the trajectory-level embedding recovers. Execution then resumes and completes normally.

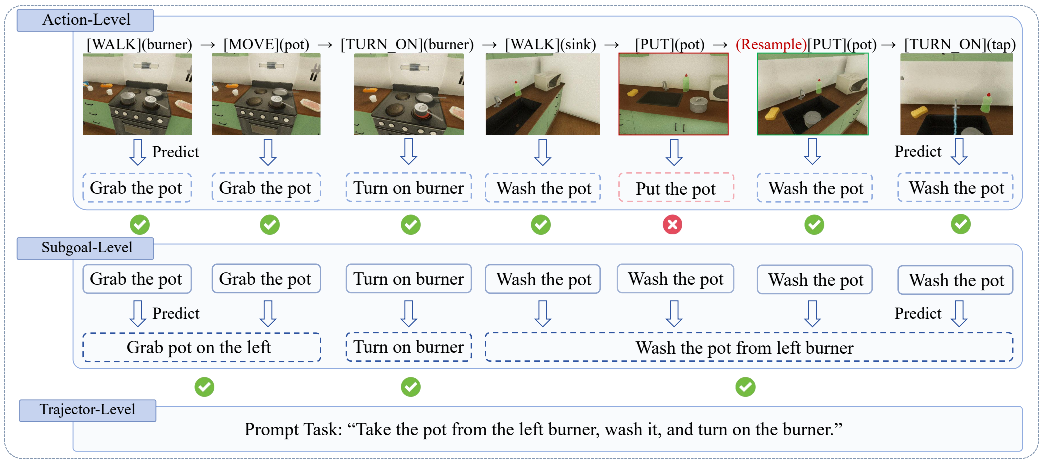

Figure 6: Illustration of ReCAPA's hierarchical correction during inference.

## E Error Propagation Metrics (EPR and PAC)

### E.1 Error Propagation Rate (EPR)

#### E.1.1 Definition

We define the *Error Propagation Rate (EPR)* at lag $k$ as the marginal increase in error probability $k$ steps after the first error in an episode:

$$\mathrm{EPR}_k \;=\; \Pr(e_{t_0+k}=1 \mid e_{t_0}=1) \;-\; \Pr(e_{t_0+k}=1 \mid e_{t_0}=0),$$

where $e_t \in \{0,1\}$ is the step-level error indicator and $t_0$ denotes the first-error time in the trajectory. Intuitively, $\mathrm{EPR}_k$ isolates the excess risk attributable to an initial failure, relative to a matched no-error baseline.

#### E.1.2 Interpretation and Properties

**Range and edge cases.** $\mathrm{EPR}_k \in [-1,1]$. $\mathrm{EPR}_k \approx 0$ implies effective containment; large positive values indicate cascading failures; negative values may indicate *active recovery*, i.e., the model becomes less error-prone after an initial mistake.

**Connection to hazard/recovery dynamics.** If errors can be modeled by a Markovian error-state abstraction, $\mathrm{EPR}_k$ equals the difference in $k$-step transition probabilities into the error state under two initial conditions ($e_{t_0}$=1 vs. 0). It is therefore a direct proxy for cascade tendency. Compared to the Propagation Attenuation Coefficient (PAC), which measures the *decay rate* of post-error risk, EPR reflects the *marginal elevation* of risk due to an initial error.

**Consistency.** Suppose the rollout process is ergodic and the matching function $\phi(\mathcal{F}_{t_0})$ successfully blocks confounding (i.e., controls for task stage and context). Then $\widehat{\mathrm{EPR}}_k$ converges in probability to $\mathrm{EPR}_k$ as the number of episodes grows. If censoring arises (e.g., episodes ending before $t_0+k$), an inverse-probability-of-censoring weighted (IPCW) estimator ensures consistency.

**Granularity.** EPR can be computed at different levels:

- *Action-level* ($e_t^{(\mathrm{act})}$): low-level execution slips.
- *Subgoal-level* ($e_t^{(\mathrm{sub})}$): structural or DAG violations.

Comparing $\mathrm{EPR}_k^{(\text{act})}$ vs. $\mathrm{EPR}_k^{(\text{sub})}$ helps identify whether cascades are driven by local control errors or higher-level planning flaws.

### E.1.3 Estimation Protocol

The empirical procedure is as follows:

1. **Identify first-error times.** For each trajectory $i$, locate $t_0^{(i)}$ (the earliest $t$ such that $e_t = 1$). If no error occurs, the trajectory is excluded from case construction.

2. **Construct case and control sets.** For each $(i, t_0^{(i)})$, treat the pair $(i, t_0^{(i)})$ as a "case." Find a "control" $(j, \tilde{t})$ from another trajectory $j$ such that:

   - $e_{\tilde{t}}^{(j)} = 0$ with no prior errors,
   - $\tilde{t} + k \le T^{(j)}$,
   - Contexts are matched: $\phi(\mathcal{F}_{\tilde{t}}^{(j)}) \approx \phi(\mathcal{F}_{t_0^{(i)}}^{(i)})$, where $\phi$ encodes subgoal ID, horizon length, scene category, and latent trajectory state.

3. **Compute probabilities.** Estimate $\widehat{p}_{\text{case}}(k) = \Pr(e_{t_0+k} = 1 \mid e_{t_0} = 1)$ and $\widehat{p}_{\text{ctrl}}(k) = \Pr(e_{\tilde{t}+k} = 1 \mid e_{\tilde{t}} = 0)$ from matched pairs.

4. **Form EPR estimate.**
$$\widehat{\mathrm{EPR}}_k \;=\; \widehat{p}_{\text{case}}(k) \;-\; \widehat{p}_{\text{ctrl}}(k).$$

5. **Summarize across $k$.** In addition to plotting $\widehat{\mathrm{EPR}}_k$ as a function of $k$, we report:

   - AUC-EPR$_W = \sum_{k=1}^{W} \widehat{\mathrm{EPR}}_k$ for horizons $W \in \{3, 5\}$,
   - the slope of $\widehat{\mathrm{EPR}}_k$ over $k$ as a compact one-number summary of cascade growth.

### E.1.4 Reporting Recommendations

**Visualization.** Always report $\widehat{\mathrm{EPR}}_k$ vs. $k$ with 95% confidence intervals (bootstrapped by episode).

**Summary statistics.** Report AUC-EPR$_W$ at $W = 3, 5$ as a concise summary statistic. For long-horizon tasks, include slope-of-$k$ analysis to quantify cascade growth.

**Comparisons.** When comparing models, include both absolute differences and relative reductions in EPR. Use identical $e_t$ definitions and matching hyperparameters across models to ensure fairness.

—

### E.2 Practical Considerations

**Censoring.** Restrict $W$ to be below the 25th percentile of remaining horizon $(T - t_0)$ to avoid heavy censoring bias.

**Variance estimation.** Use per-episode bootstrap resampling for CI bands.

**Level separation.** Always report both action-level and subgoal-level EPR curves to clarify the source of propagation.

**Robustness checks.** Verify results are stable to the choice of $\phi$ (matching function) and distance metric.

### E.3 Propagation Attenuation Coefficient (PAC)

**Definition.** We define the *Propagation Attenuation Coefficient (PAC)* at lag $k$ as the relative decay rate of post-error risk:

$$\mathrm{PAC}_k \;=\; \frac{\Pr(e_{t_0+k}=1 \mid e_{t_0}=1)}{\Pr(e_{t_0+1}=1 \mid e_{t_0}=1)},$$

where $e_t$ is the error indicator and $t_0$ denotes the first-error time. Intuitively, PAC measures how quickly the elevated error probability induced by the first error attenuates over time. A value close to $1$ indicates persistent risk, while a value below $1$ indicates attenuation.

#### E.3.1 Interpretation and Properties

**Range and meaning.** $\mathrm{PAC}_k \in [0,\infty)$.

- $\mathrm{PAC}_k \approx 1 \;\Rightarrow\;$ error risk is persistent, cascades continue.
- $\mathrm{PAC}_k < 1 \;\Rightarrow\;$ error risk attenuates; the system recovers.
- $\mathrm{PAC}_k > 1 \;\Rightarrow\;$ error risk escalates faster than the initial shock (rare but possible in unstable systems).

**Connection to survival/hazard analysis.** PAC can be viewed as an analogue of a hazard decay factor: it compares the conditional error hazard at lag *k* to that immediately after the error. Whereas EPR captures the *absolute marginal risk increase*, PAC quantifies the *relative decay speed* of this risk.

**Granularity.** PAC can be applied at both:

- *Action-level*: robustness of local control after a slip.
- *Subgoal-level*: structural recovery after violating a planning dependency.

#### E.3.2 Estimation Protocol

The empirical procedure is as follows:

1. **Identify first-error times.** Same as EPR: find $t_0^{(i)}$ for each trajectory.
2. **Compute conditional error probabilities.** For each $k$, estimate

$$\widehat{q}(k) = \Pr(e_{t_0+k}=1 \mid e_{t_0}=1).$$

3. **Form PAC estimate.** Normalize by the immediate post-error risk:

$$\widehat{\mathrm{PAC}}_k = \frac{\widehat{q}(k)}{\widehat{q}(1)}.$$

4. **Summarize across $k$.** Plot $\widehat{\mathrm{PAC}}_k$ vs. $k$; additionally, compute area-under-curve (AUC) metrics:

$$\text{AUC-PAC}_W = \frac{1}{W}\sum_{k=1}^{W}\widehat{\mathrm{PAC}}_k,$$

as a compact indicator of recovery speed.

#### E.3.3 Reporting Recommendations

**Visualization.** Always report $\widehat{\mathrm{PAC}}_k$ vs. $k$ with 95% confidence intervals.

**Summary statistics.** Report AUC-PAC$_W$ at $W = 3, 5$ as a concise recovery-speed measure.

**Comparisons.** Compare models both in terms of absolute persistence (PAC close to 1) and relative acceleration/attenuation trends.

### E.3.4 Practical Considerations

**Normalization stability.** If $\widehat{q}(1)$ is very small, PAC may be unstable. Exclude cases with vanishing immediate risk or regularize with a small $\epsilon$.

**Censoring.** Restrict horizon $W$ to avoid censoring, as in EPR.

**Variance estimation.** Use bootstrap resampling over episodes.

**Level separation.** Report both action-level and subgoal-level PAC curves, since persistence patterns often differ across levels.

**Consistency (Remark).** Since PAC is estimated from $\widehat{q}(k)$ values that are themselves consistent estimators of conditional error probabilities, $\widehat{\lambda}$ inherits consistency under standard assumptions for exponential regression fits. We omit a formal proof for brevity.

## F Theoretical Analysis

### F.1 Execution Improvement via Alignment Loss Minimization

It has been shown that minimizing alignment losses such as InfoNCE, DPO, or Sinkhorn divergence over the execution network $\pi_\theta(a|s)$ can lead to improvements in expected task success. These losses are typically defined to encourage the execution to assign higher probability to preferred or expert actions while penalizing suboptimal ones. To analyze their impact on execution improvement, the alignment loss is reformulated as a surrogate objective over the log-probability $\log \pi_\theta(a|s)$.

Formally, a general alignment loss can be expressed as:

$$\mathcal{L}_{\text{align}}(\pi_\theta) = \mathbb{E}_{(s,a^+,\{a_i^-\})}\left[\ell\left(\log \pi_\theta(a^+|s), \{\log \pi_\theta(a_i^-|s)\}\right)\right], \tag{7}$$

where $a^+$ represents the preferred (e.g., expert or high-reward) action, and $\{a_i^-\}$ are sampled negatives. The contrastive function $\ell$ encourages the log-probability of $a^+$ to be separated from that of the negatives. Under this formulation, the gradient of the alignment loss can be written as:

$$\nabla_\theta \mathcal{L}_{\text{align}}(\pi_\theta) = -\mathbb{E}_{(s,a)}\left[\hat{A}(s,a)\nabla_\theta \log \pi_\theta(a|s)\right], \tag{8}$$

where $\hat{A}(s,a)$ is a weight derived from relative preferences, reward differences, or likelihood ratios. This expression mirrors the form of a execution gradient update, where $\hat{A}(s,a)$ acts as a surrogate advantage estimator.

In the case of InfoNCE, the alignment loss takes the form of a softmax log-likelihood objective over sampled actions:

$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{\exp(\log \pi_\theta(a^+|s))}{\exp(\log \pi_\theta(a^+|s)) + \sum_i \exp(\log \pi_\theta(a_i^-|s))}, \tag{9}$$

which induces gradients that push up the probability of the positive action $a^+$ while pulling down the negatives. The resulting update direction has been shown to approximate the advantage-weighted execution gradient when $a^+$ is selected according to reward or preference feedback.

For Sinkhorn-based alignment losses, a probabilistic interpretation has been proposed by viewing the alignment as a soft permutation induced via entropic optimal transport. Specifically, given a cost matrix between token embeddings in the prompt and the trajectory, a doubly stochastic transport plan is computed using the Sinkhorn-Knopp algorithm. The plan induces a distribution over token-to-token matches, and the loss is minimized when the trajectory embedding distribution is optimally aligned (in transport cost) with the prompt. When the cost is reward-informed or semantically structured, this process implicitly enforces reward-relevant permutation between prompt instructions and executed actions. The resulting transport-based gradient aligns the latent plan structure with the execution behavior.

By the execution gradient theorem, these gradient directions align with execution improvement provided that the surrogate signal $\hat{A}(s,a)$ is a consistent estimator of the true advantage $A^{\pi}(s,a)$. Under standard assumptions (bounded variance, small step size), minimizing alignment losses therefore promotes higher expected cumulative reward $J(\theta)$. Therefore, updates that reduce these alignment losses can be expected to lead to execution improvement in practice. This theoretical insight is consistent with contrastive execution gradient methods such as CoPG, and the expected success rate gains have been empirically verified in the experiments above.

### F.2 Distributional Alignment Theory for Contrastive Loss

The alignment of model-generated trajectories with prompt intent can be viewed as a distribution matching problem. Let $\mathcal{D}_{\text{traj}}$ denote the distribution over generated trajectory embeddings and $\mathcal{D}_{\text{target}}$ the idealized distribution implied by ground-truth behaviors or reward-aligned samples. During contrastive training, modules produce positive and negative samples whose embedding distributions are encouraged to match the target via minimization of contrastive losses such as InfoNCE or Sinkhorn divergence.

This process can be interpreted through the lens of distributional discrepancy minimization. For InfoNCE, the objective implicitly minimizes an upper bound on the Jensen–Shannon divergence between $\mathcal{D}_{\text{traj}}$ and $\mathcal{D}_{\text{target}}$ by maximizing the mutual information between aligned pairs. Under this view, contrastive learning serves to pull the generated trajectory distribution toward the reward-consistent region of the target space.

When Sinkhorn divergence is used, the discrepancy between $\mathcal{D}_{\text{traj}}$ and $\mathcal{D}_{\text{target}}$ is explicitly reduced through an entropy-regularized optimal transport plan. Given empirical samples from both distributions, a transport map is computed that minimizes cost while maintaining marginal consistency. The resulting divergence has been shown to upper bound the Wasserstein distance under entropic smoothness, and can be used to quantify how closely the generated execution adheres to the prompt-induced distribution.

Under mild assumptions (bounded support, Lipschitz continuity of cost), the contrastive alignment loss $L_{\text{align}}(\theta)$ is provably minimized when the distributional discrepancy vanishes, i.e.,

$$\text{div}(\mathcal{D}_{\text{traj}} \| \mathcal{D}_{\text{target}}) \leq \mathcal{O}(L_{\text{align}}(\theta)) + \varepsilon, \tag{10}$$

where $\text{div}(\cdot)$ may denote JS, Sinkhorn, or Wasserstein distance, and $\varepsilon$ denotes residual stochasticity. As a result, minimizing the contrastive loss leads the model toward a low-drift regime where trajectory samples remain semantically consistent with target plans.

### F.3 Representation Alignment Bound

We study when minimizing representation-level alignment losses between prompts and trajectories controls a discrepancy and mitigates semantic drift. Let $P = \mathcal{D}_{\text{prompt}}$ be the prompt-conditional distribution and $Q_\theta = \mathcal{D}_{\text{traj}}(\theta)$ the trajectory distribution induced by execution $\pi_\theta$ in a shared embedding space $\mathcal{Z} \subset \mathbb{R}^d$. Write $L_{\text{align}}(\theta)$ for an alignment objective such as InfoNCE or a Sinkhorn-based transport loss. Assume embeddings are bounded; the ground cost $c : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}_{\geq 0}$ is bounded and Lipschitz; and for OT we use $\varepsilon$–entropy-regularized transport with the debiased Sinkhorn divergence $S_\varepsilon$. Let $\hat{P}_m, \hat{Q}_n$ be empirical distributions from $m$ and $n$ samples, and let $\Theta$ be a execution class with capacity term $\mathfrak{R}_n(\Theta)$.

With probability at least $1-\delta$ over the draws of $(\hat{P}_m, \hat{Q}_n)$, there exists a constant $C(\varepsilon, c)$ such that

$$\sup_{\theta \in \Theta} \left| D(P, Q_\theta) - \widetilde{L}_{\text{align}}(\hat{P}_m, \hat{Q}_n; \theta) \right| \leq C(\varepsilon, c) \left( \sqrt{\frac{\log(1/\delta)}{m}} + \sqrt{\frac{\log(1/\delta)}{n}} \right) + \mathfrak{R}_n(\Theta) + b(\varepsilon), \tag{11}$$

where $D$ is a population-level divergence matched to the alignment loss, $\widetilde{L}_{\text{align}}$ is its empirical counterpart (debiased for OT), and $b(\varepsilon)$ is the regularization bias that vanishes as $\varepsilon \downarrow 0$.

For InfoNCE, the expected contrastive risk upper-bounds an $f$–divergence between the joint and product of marginals; in particular,

$$\mathrm{JS}\big(P, Q_\theta\big) \;\le\; \alpha\,\mathbb{E}[L_{\text{InfoNCE}}(\theta)] + c_0,$$

for constants $(\alpha, c_0)$ determined by the negative-sampling scheme and temperature, and $\widetilde{L}_{\text{align}}$ is the minibatch InfoNCE. For Sinkhorn-based alignment, take $D = W_1$ and use the debiased divergence:

$$W_1\big(P, Q_\theta\big) \;\le\; S_\varepsilon\big(P, Q_\theta\big) + b(\varepsilon), \qquad \widetilde{L}_{\text{align}}(\hat{P}_m, \hat{Q}_n; \theta) = S_\varepsilon(\hat{P}_m, \hat{Q}_n).$$

Plugging either instance into equation 11 gives a high-probability generalization guarantee from empirical alignment to population discrepancy.

*Assumption (OT setting).* We assume the cost function $c$ is bounded and Lipschitz, and that both $P$ and $Q$ have bounded support. The entropic Sinkhorn divergence $S_\varepsilon$ uniformly approximates the 1-Wasserstein distance $W_1$ up to a bias $b(\varepsilon)$, with standard statistical rates $O(m^{-1/2} + n^{-1/2})$.

*Corollary (Sinkhorn specialization).* Under the above assumptions, for any $\delta \in (0, 1)$,

$$\sup_{\theta\in\Theta} W_1\big(P, Q_\theta\big) \;\le\; \sup_{\theta\in\Theta} S_\varepsilon\big(\hat{P}_m, \hat{Q}_n; \theta\big) \;+\; C(\varepsilon, c)\left(\sqrt{\tfrac{\log(1/\delta)}{m}} + \sqrt{\tfrac{\log(1/\delta)}{n}}\right) \;+\; \mathfrak{R}_n(\Theta) \;+\; b(\varepsilon). \tag{12}$$

Thus, uniformly controlling the empirical Sinkhorn loss suffices to bound the population trajectory–prompt divergence up to estimation and regularization terms.

Since the inequality in equation 12 holds uniformly over all $\theta \in \Theta$, it applies to each training iterate $\theta_t$. Averaging over $t = 1, \ldots, T$ yields

$$\frac{1}{T}\sum_{t=1}^{T} D\big(P, Q_{\theta_t}\big) \;\le\; \frac{1}{T}\sum_{t=1}^{T} \widetilde{L}_{\text{align}}(\hat{P}_m, \hat{Q}_n; \theta_t) + \mathcal{E}_{\text{stat}}(m, n, \delta, \Theta) + b(\varepsilon), \tag{13}$$

where $\mathcal{E}_{\text{stat}}$ collects the $\tilde{O}(m^{-1/2} + n^{-1/2})$ and capacity terms. Thus, as optimization reduces empirical alignment loss and $\varepsilon$ is chosen small but stable, the divergence decreases correspondingly, limiting semantic drift. Choosing $D = W_1$ and $\widetilde{L}_{\text{align}} = S_\varepsilon$ gives the Sinkhorn training-iterate bound

$$\frac{1}{T}\sum_{t=1}^{T} W_1\big(P, Q_{\theta_t}\big) \;\le\; \frac{1}{T}\sum_{t=1}^{T} S_\varepsilon\big(\hat{P}_m, \hat{Q}_n; \theta_t\big) + \mathcal{E}_{\text{stat}}(m, n, \delta, \Theta) + b(\varepsilon). \tag{14}$$

The alignment functionals above are Lipschitz-stable with respect to empirical measures (bounded smooth scores for InfoNCE; stability of $OT_\varepsilon$ under bounded Lipschitz $c$ for Sinkhorn). Concentration (e.g., McDiarmid or transport inequalities) gives $\tilde{O}(m^{-1/2} + n^{-1/2})$ control for fixed $\theta$; uniformity over $\Theta$ follows by symmetrization and a capacity term $\mathfrak{R}_n(\Theta)$. For OT, relate $S_\varepsilon$ to $W_1$ and isolate $b(\varepsilon)$; for InfoNCE, use standard $f$–divergence or MI control to upper-bound JS. Combining these ingredients gives equation 11 and, by averaging, equation 13.

## F.4 Hierarchical Convergence Bound

The proposed hierarchical framework operates across multiple abstraction levels—actions, subgoals, and task-level intents—each equipped with a dedicated alignment loss. To ensure stable joint optimization, it is necessary to establish convergence guarantees or lower bounds on sample efficiency when all levels are trained simultaneously.

The overall optimization can be viewed as a multi-objective gradient process, where each level $l \in \{\text{action}, \text{subgoal}, \text{trajectory}\}$ minimizes its own alignment loss $\mathcal{L}^{(l)}_{\text{align}}(\theta^{(l)})$ while contributing to the global performance. By treating the joint update as a composite execution gradient step over a stacked parameter space $\theta = [\theta^{(1)}, \theta^{(2)}, \theta^{(3)}]$, the learning dynamics can be analyzed using mirror descent under block-decomposed gradient feedback.

Under standard smoothness and bounded gradient assumptions, it can be shown that the average alignment loss across levels satisfies the following rate:

$$\min_{1\le t\le T} \frac{1}{3}\sum_{l=1}^{3} \mathcal{L}^{(l)}_{\text{align}}(\theta^{(l)}_t) - \mathcal{L}^{(l)*} \le \mathcal{O}(1/\sqrt{T}), \tag{15}$$

where $\mathcal{L}^{(l)*}$ denotes the optimal alignment loss at level $l$, and $T$ is the number of joint gradient updates. This sublinear convergence is consistent with standard mirror descent bounds in stochastic settings and ensures no individual layer dominates optimization dynamics.

Furthermore, when coordination across levels is regularized (e.g., by shared representations or alignment constraints), the convergence can be accelerated. In particular, when surrogate gradients are aligned and cross-level variance is bounded, an improved rate of $\mathcal{O}(1/T)$ can be achieved, indicating that hierarchical updates benefit from structural decomposition.

These results suggest that hierarchical not only enables semantic control across abstraction levels but also preserves convergence efficiency. The modular structure improves gradient conditioning and reduces update variance, leading to better sample efficiency compared to flat architectures. Empirical convergence patterns across levels are reported, where the multi-level alignment loss steadily decreases throughout training.

### F.5 Joint Convergence of Hierarchical

In the proposed architecture, is performed across multiple levels of abstraction, and each level contributes a separate alignment loss that is jointly optimized. To ensure stable training, it is essential to establish that the overall optimization remains convergent when these multi-level alignment objectives are updated concurrently.

The full stack can be modeled as a constrained optimization problem, where lower-level alignment losses act as structured regularization terms within a meta-execution update. Let $\pi_\theta$ denote the overall execution composed of nested sub-policies across levels. The global objective can then be written as:

$$\min_{\theta} \; \mathcal{L}_{\text{task}}(\pi_\theta) \quad \text{subject to} \quad \mathcal{L}_{\text{align}}^{(l)}(\pi_\theta) \le \epsilon_l, \quad \forall l \in \{1, 2, 3\}, \tag{16}$$

where $\mathcal{L}_{\text{task}}$ denotes the primary reward-based objective and $\mathcal{L}_{\text{align}}^{(l)}$ are alignment constraints at different levels.

This structure parallels that of constrained execution gradient optimization. When solved via primal-dual or Lagrangian methods, convergence to a locally optimal solution can be guaranteed under standard assumptions of smoothness and bounded constraint curvature. Moreover, recent meta-RL results show that hierarchical or nested policies trained with level-specific constraints converge under similar conditions. In this formulation, alignment losses are interpreted as soft constraints enforcing semantic consistency within the multi-level planning hierarchy.

In addition, when modules are updated via trust-region or clipping mechanisms (e.g., using variants of constrained execution optimization), their individual losses can be stabilized without disrupting global reward improvement. As a result, the multi-level architecture jointly converges toward a stable equilibrium where both alignment fidelity and task reward are satisfied.

These results provide theoretical support that the proposed hierarchical system maintains optimization stability even under simultaneous multi-level loss supervision. Empirical convergence patterns further support this, showing that execution improvement persists despite the layered optimization structure.

## G Prompt Templates

### G.1 Hard Negative Generation

**You are an expert in analyzing robotic task trajectories.** Your task is to take a successful trajectory and create a **hard negative** version of it. A hard negative is a trajectory that looks plausible but is flawed in a subtle, specific way. You must introduce **ONLY ONE** of the following error types, as specified by the user.

- **Action Error**
  Replace a single critical action with a plausible but incorrect one. Example: In a "make coffee" task, instead of `pickup cup`, you might use `pickup filter`, which is a related object but wrong for the step of pouring coffee.

- **Timing Error**
  Change the order of two critical, non-dependent actions where the order subtly matters for efficiency or naturalness. Example: In `make breakfast`, `pour coffee` then `toast bread` is fine, but reversing them might result in cold coffee.
- **Logic Error**
  Create a trajectory that violates physical or common-sense rules. Example: Trying to `pour water` before the cup is picked up.
- **Sequence Error**
  Reverse the order of two dependent actions, making the sequence impossible. Example: `place apple in microwave` before `open microwave`.

You will be given a successful trajectory and an error type to introduce. Your response must be **only** the flawed trajectory, formatted exactly like the input.

### MULTI-AGENT SYSTEM PROMPTS

**Agent 0: Executor**
You are a robot agent focused on execution. Given the current state and a high-level goal, your task is to select the most logical and immediate action to perform next. Focus only on the next step.

**Agent 1: Monitor/Critic**
You are a monitoring agent. Your job is to observe the trajectory and evaluate progress. Output `continue`, `no_op`, `alert`, or `replan` based on trajectory status.

**Agent 2: Planner**
You are a high-level planning agent. Your job is to identify the next major sub-goal or phase in the task. Output abstract-level goals like `acquire_all_ingredients`.

**Agent 3: Alignment Agent**
You ensure the plan stays consistent with the user's instruction. Output `continue`, `realign`, or `correct_course` based on semantic alignment.