TASK-SPECIFIC CONTEXT OPTIMIZATION FOR TRAINING-FREE ROBOT PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Frontier vision-language models (VLMs) can serve as off-the-shelf high-level planners in hierarchical robot policies, using multimodal reasoning to infer subtasks for low-level execution. However, their out-of-the-box use leaves ambiguity in offline subtask annotation (for low-level policy fine-tuning) and online subtask planning, resulting in inconsistent guidance that we find degrade long-horizon task performance. We introduce TAsk-Specific Context Optimization (TASCO), a framework that adapts frozen frontier VLMs through task-specific context to produce consistent offline annotations and align causal online predictions with them. Specifically, TASCO queries a separate VLM, termed the *context optimizer*, to induce this context from a reference demonstration, providing a common basis for consistent task decomposition. Next, for temporally consistent online planning, the context optimizer iteratively refines this context by diagnosing mismatches between offline annotations and online predictions, accepting updates only when their consistency improves. Across multiple frontier VLMs and low-level policies, TASCO substantially improves complex, long-horizon task performance on simulated and real-robot benchmarks, while naïve deployment yields surprisingly poor results. These findings establish TASCO as a model-agnostic framework for harnessing the growing reasoning capabilities of frontier VLMs by aligning high-level planning with low-level execution. Project page is available at https://iclr-tasco.github.io/.

1 INTRODUCTION

Hierarchical robot policies (Shi et al., 2025; Li et al., 2025; Huang et al., 2025; Zhang et al., 2025; Ji et al., 2026; Chen et al., 2026c; Yang et al., 2026; Zhang et al., 2026) have advanced long-horizon task execution by employing vision-language models (VLMs) as high-level planners to guide low-level visuomotor policies via subtask instructions. In particular, frontier VLMs offer strong multimodal reasoning capabilities, enabling their use as off-the-shelf high-level planners. Furthermore, they can annotate training demonstrations offline, providing subtask instructions for low-level policy training.

However, without task-specific fine-tuning, frontier VLMs face inherent ambiguity that can lead to inconsistent offline annotations and further deviations from them during online planning. In *offline annotation*, subtask instructions can vary in both granularity and phrasing for the same behavior, leaving ambiguity about 'what' instruction to assign in a given situation (see Figure 1(a)). Such variation can yield inconsistent subtask annotations across demonstrations, complicating low-level policy training. In *online planning*, the absence of future observations introduces an additional source of uncertainty about 'when' to transition between subtasks (see Figure 1(b)). Specifically, switching subtasks too early can leave the current subtask unfinished and expose the low-level policy to unfamiliar state-subtask pairs, while switching too late can cause repeated or unintended actions.

Crucially, we find that such temporal inconsistencies in subtask transitions noticeably degrade the performance of hierarchical robot policies on long-horizon tasks. To isolate these effects during training and execution, we conduct controlled experiments (see Section A for details) on RMBench (Chen et al., 2026b) using a simulator-based, oracle high-level planner and $\pi_{0.5}$ (Physical Intelligence, 2025) as the low-level policy. For *offline annotation*, we simulate temporal inconsistency by injecting jitter into the oracle subtask transitions used to train low-level policies, while keeping the oracle high-level planner fixed at evaluation. As shown in Figure 2(a), inconsistent training annotations can impair execution, even when transition jitter is limited to just half a high-level planning interval.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

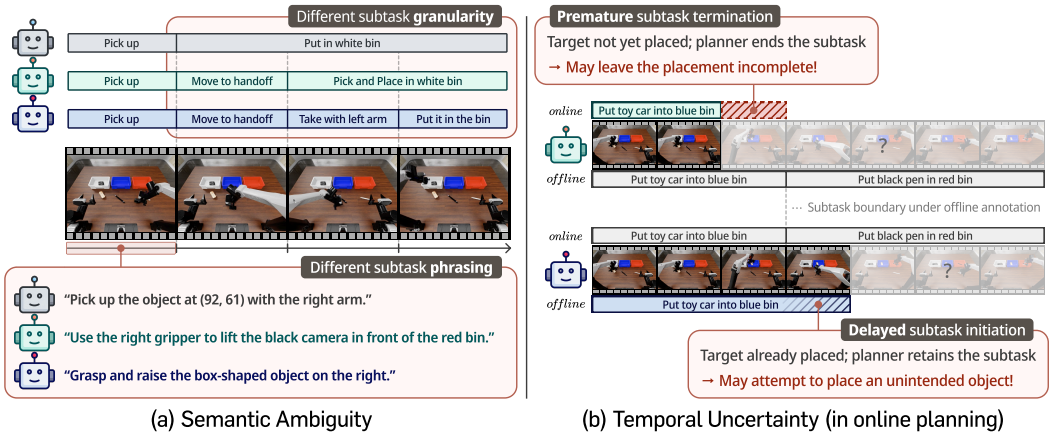

Figure 1: **Subtask inconsistencies in offline annotation and online planning.** (a) Annotations of the same demonstration can differ in subtask granularity and phrasing, which can yield inconsistent subtask supervision for the low-level policy. (b) Without future observations, premature subtask termination can expose the low-level policy to unfamiliar state-subtask pairs, whereas delayed initiation of the next subtask can cause repeated or unintended actions.

In contrast, for *online planning*, we simulate temporal inconsistency by applying constant offsets to the subtask transitions of the oracle high-level planner, while keeping the low-level policy trained on oracle annotations fixed. As shown in Figure 2(b), performance drops sharply with larger deviations from the oracle subtask transition times in either direction. Notably, such deviations are also observed across API-based frontier VLMs (OpenAI, 2026b; Anthropic, 2026; Google DeepMind, 2026b; SpaceXAI, 2026). In Figure 2(c), we plot each model's online transition offsets relative to its own offline annotations on the same RMBench demonstrations. This reveals deviations in both directions, with a bias toward earlier online transitions, which are associated with larger performance drops in Figure 2(b). This suggests that strong multimodal reasoning priors alone do not ensure consistent high-level guidance, raising the following research question: *How can frontier VLMs be adapted to produce consistent offline annotations and align online planning with them on target tasks?*

Our approach. We present TAsk-Specific Context Optimization (TASCO), a framework that adapts a frozen frontier VLM as a high-level planner for target tasks through explicit task-specific context. To construct and refine this context, TASCO queries a separate VLM, termed the *context optimizer*, in two stages: (a) *task-specific context induction* for consistent offline annotation and (b) *task-specific context refinement* for aligning online planning with the resulting annotations ('*task-specific*' omitted hereafter for brevity). During *context induction*, we prompt the context optimizer with the task instruction and a reference demonstration video to extract fine-grained task details left unspecified by the instruction alone. The high-level planner then applies the induced context to annotate all training demonstrations with subtask labels, yielding subtask-conditioned training data for the low-level visuomotor policy. During *context refinement*, we replay a small subset of training demonstrations under online observation constraints, allowing the high-level planner to access only observations available up to each planning step. We then prompt the context optimizer with the causal online predictions and fixed offline annotations to diagnose the resulting discrepancies and propose context updates to reduce them. Each proposed update is accepted only if replanning with the resulting context improves a quantitative measure of subtask agreement across the replayed demonstrations. At deployment, the high-level planner uses the refined context to guide the trained low-level policy.

We evaluate TASCO with multiple frontier VLMs as high-level planners and vision-language-action models (VLAs) as low-level policies across simulation and real-robot settings. With Gemini 3.7 Flash (Google DeepMind, 2026a) as the planner and $\pi_{0.5}$ (Physical Intelligence, 2025) as the policy, TASCO raises the RMBench success rate from 29.6% to 51.8% and the RoboDojo *Long-Horizon* score from 32.0 to 41.3 (Chen et al., 2026a). Furthermore, on real-robot tasks requiring comprehensive visual reasoning to repeatedly select an appropriate subtask among alternatives, TASCO improves performance by 51.7%p over the naïve hierarchical baseline. These results establish TASCO as a model-agnostic plug-in that consistently improves performance across diverse hierarchical robot policies by aligning online high-level guidance with offline training annotations.

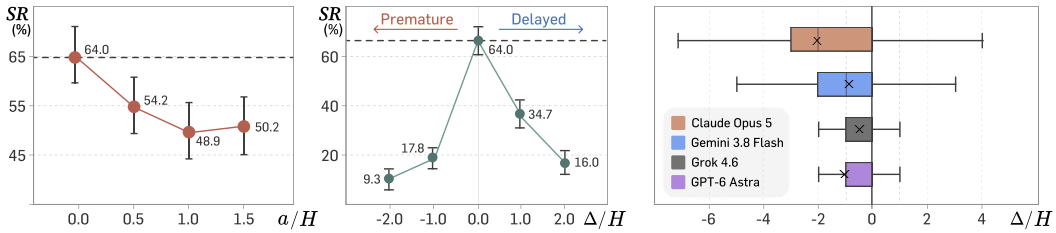

Figure 2: **Subtask transition inconsistency in hierarchical robot policies.** Controlled perturbations show how transition inconsistency affects task success: (a) offline annotation jitter with maximum magnitude a and (b) online transition offset Δ, where negative/positive offsets indicate premature/delayed transitions. (c) shows offline–online transition discrepancies for frontier VLMs, comparing offline annotations of trajectories with online predictions using observations available at each planning step. Here, H denotes one high-level planning interval (≈ 0.83 s in our setup).

Contributions. Our contributions are as follows:

- Motivated by subtask design ambiguity and the additional temporal mismatch introduced by online planning, we propose TASCO, a plug-in framework that adapts frozen frontier VLMs as high-level planners through task-specific context.
- We design a two-stage context optimization procedure that queries a separate VLM, termed the *context optimizer*. *Context induction* extracts fine-grained task details from a reference demonstration for consistent offline annotation, while *context refinement* diagnoses offline–online discrepancies and proposes updates, which TASCO accepts only when quantitative subtask agreement improves.
- We show through controlled experiments that inconsistent subtask transitions degrade long-horizon task performance. We further demonstrate that TASCO consistently outperforms naïve hierarchical baselines using the same models across frontier VLM planners, low-level VLA policies, and simulated and real-robot benchmarks.

2 Related Works

Hierarchical robot policies. Hierarchical robot policies separate task planning from motor control for long-horizon execution. Early systems used language models as high-level planners to select and sequence predefined robot skills or actions (Ahn et al., 2022; Singh et al., 2023). More recently, hierarchical systems have paired vision-language models (VLMs) as high-level planners with vision-language-action models (VLAs; Shi et al., 2025; Ji et al., 2026; Yang et al., 2026; Chen et al., 2026c) or world-action models (WAMs; Zhang et al., 2026) as low-level policies. These learned policies translate the planner's intermediate guidance and visual observations into motor actions. Controlled comparisons further show that planning representations affect hierarchical VLA performance (Gao et al., 2026). Across these systems, high-level guidance takes several forms: language subtasks (Shi et al., 2025; Ji et al., 2026; Zhang et al., 2026), 2D trajectories or structured spatial representations (Li et al., 2025; Zhang et al., 2025), and combinations of language with spatial grounding (Yang et al., 2026; Chen et al., 2026c). In this work, we focus on hierarchies with learned visuomotor low-level policies and use multimodal instructions that pair a textual subtask with an auxiliary visual keyframes.

Context optimization. Prompt optimization adapts frozen language models by revising the instructions and few-shot examples supplied in their prompts. Early methods generate and select instructions according to their performance on evaluation examples (Zhou et al., 2023; Yang et al., 2024). More recent work extends optimization to multi-stage language-model programs (Opsahl-Ong et al., 2024), while textual critiques of model outputs guide prompt revisions (Yuksekgonul et al., 2024; Agrawal et al., 2026). In robotics, demonstration videos have been used to derive task plans (Wang et al., 2025), and language-annotated demonstrations to construct structured action representations for learned skills (Liu et al., 2024). Together, these studies show how prompt revisions and demonstrations can provide task-specific guidance. In our setting, however, this guidance must remain consistent across full-trajectory offline annotation and causal online planning. TASCO therefore induces task-specific context from a reference demonstration for consistent offline annotation, then refines it using measured subtask agreement between fixed offline annotations and causal online predictions.

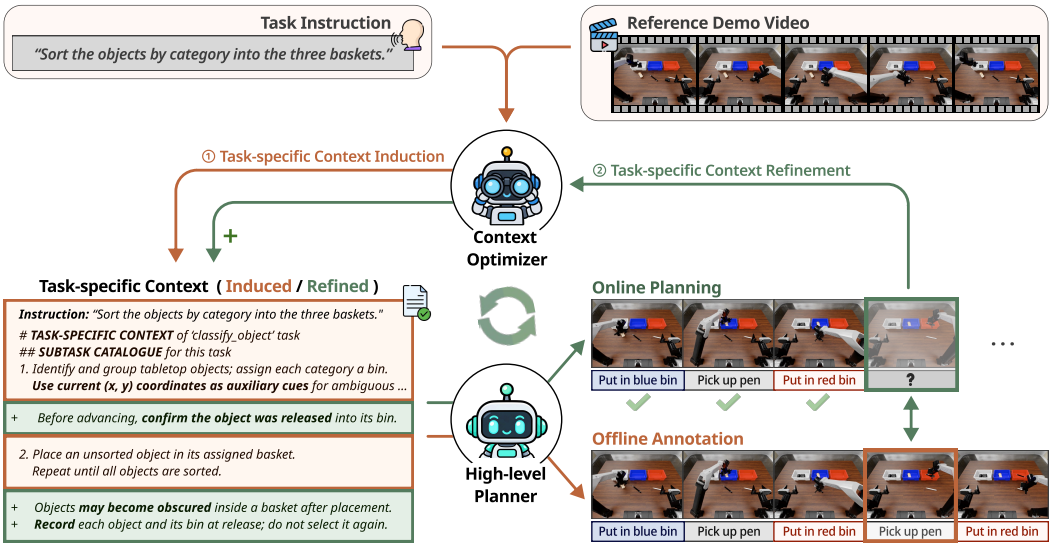

Figure 3: **An overview of TASCO.** TASCO adapts a frozen high-level planner through task-specific context to align offline annotation and online planning. A separate VLM, the *context optimizer*, performs *context induction* from a reference demonstration to support consistent annotation for low-level policy training, followed by *context refinement* that diagnoses offline–online mismatches and proposes updates. Updates are accepted only when replanning improves mean subtask agreement across replayed demonstrations, yielding the refined context used at deployment.

3 METHOD

We introduce TASCO, a framework that adapts a frozen frontier vision-language model (VLM) as a high-level planner through task-specific context. A separate frozen VLM, termed the *context optimizer* $\mathcal{Q}_\psi$, optimizes this context in two stages: (i) *context induction*, which extracts fine-grained task details from a reference demonstration for consistent offline annotation (Section 3.1); and (ii) *context refinement*, which aligns online planning with the resulting fixed offline annotations (Section 3.2).

We consider a hierarchical robot policy in which a high-level planner $\mathcal{P}_\phi$ guides a low-level visuomotor policy π_θ through subtask instructions. To support training and execution, the planner operates in two prompting modes. For training, its offline annotation mode $\mathcal{P}_\phi^{\text{off}}$ accesses complete demonstration videos to produce subtask annotations. During execution, its online planning mode $\mathcal{P}_\phi^{\text{on}}$ uses only observations available up to each planning step. Both modes share planner parameters ϕ, which remain fixed alongside the optimizer parameters ψ throughout context induction and refinement.

Formally, let ℓ denote the global task instruction. At each high-level planning step t, the planner receives recent observation frames $\mathbf{O}_t$ up to t and textual memory m_{t^-} from the preceding planning step t^-. Together, they form a causal representation of the observation history, $\mathcal{H}_t = (\mathbf{O}_t, m_{t^-})$. Given task-specific context $\mathcal{C}$, the planner processes this history and the task instruction to predict a subtask and update its memory:

$$(y_t, m_t) = \mathcal{P}_\phi^{\text{on}}(\mathcal{H}_t, \ell; \mathcal{C}), \tag{1}$$

where y_t is a textual subtask label guiding low-level control and m_t is the updated memory. The label can be paired with a retrieved visual keyframe to provide multimodal guidance. At each invocation, the low-level policy predicts an action chunk from the current observations, proprioceptive state, and latest guidance. Keyframe retrieval and low-level execution are detailed in Section B.3.

3.1 TASK-SPECIFIC CONTEXT INDUCTION

A global task instruction often leaves subtask granularity, phrasing, and transition criteria unspecified, allowing similar behaviors to receive different annotations across demonstrations. We therefore induce task-specific context from a reference demonstration to establish a shared annotation protocol.

Inducing context from a reference demonstration. For instruction ℓ, let $\mathcal{D}_\ell$ denote the training demonstrations and $\mathbf{V}^{\text{ref}}$ the video of one reference demonstration selected from this set. We prompt the context optimizer with the instruction and reference video:

$$\mathcal{C}^{(0)} = \mathcal{Q}_\psi(\ell, \mathbf{V}^{\text{ref}}; p_{\text{ind}}), \tag{2}$$

where p_{ind} is a fixed system prompt specifying the induction procedure. The prompt asks the optimizer to identify which subtasks to distinguish, their granularity and canonical phrasing, and criteria for transitioning between them. Grounded in the reference demonstration, $\mathcal{C}^{(0)}$ makes these task details explicit and lists the complete set of canonical subtask labels $\mathcal{Y}_\ell$.

Offline annotation with the induced context. Using $\mathcal{C}^{(0)}$, the high-level planner annotates all demonstrations in $\mathcal{D}_\ell$. For demonstration i, it jointly processes observations spanning the full video $\mathbf{V}^{(i)}$ to produce chronologically ordered subtask labels:

$$\left(\bar{y}_t^{(i)}\right)_{t\in\mathcal{T}_i} = \mathcal{P}_\phi^{\text{off}}(\mathbf{V}^{(i)}, \ell; \mathcal{C}^{(0)}), \tag{3}$$

where $\mathcal{T}_i$ contains annotation timesteps sampled at a fixed interval and $\bar{y}_t^{(i)} \in \mathcal{Y}_\ell$ is the label at timestep t. Access to preceding and subsequent observations allows each timestep to be interpreted within the complete execution. The resulting labels, alongside optional visual keyframes, condition low-level policy training. These offline annotations also serve as fixed targets for context refinement.

3.2 Task-specific Context refinement

Although the induced context establishes a shared annotation protocol, the absence of future observations introduces additional uncertainty about subtask transition timing during online planning. We refine the context by replaying training demonstrations under online observation constraints and aligning causal predictions with their fixed offline annotations.

Measuring subtask agreement. We use a fixed subset of N demonstrations from $\mathcal{D}_\ell$, indexed by $i = 1, \ldots, N$ throughout this subsection. For each context $\mathcal{C}$, the planner processes their recorded observations sequentially, using the annotation timesteps $\mathcal{T}_i$ as planning timesteps. At each step, it follows Equation (1), accessing only recent observations up to that step and its own preceding memory generated under $\mathcal{C}$. The offline annotations generated with $\mathcal{C}^{(0)}$ remain fixed.

Let $\hat{y}_t^{(i)}(\mathcal{C}) \in \mathcal{Y}_\ell$ denote the resulting online prediction. Because the context specifies the canonical labels $\mathcal{Y}_\ell$, both modes use the same candidate set. We measure agreement as the fraction of planning steps with matching labels:

$$A_i(\mathcal{C}) = \frac{1}{|\mathcal{T}_i|} \sum_{t\in\mathcal{T}_i} \mathbb{I}\left[\hat{y}_t^{(i)}(\mathcal{C}) = \bar{y}_t^{(i)}\right], \qquad A(\mathcal{C}) = \frac{1}{N}\sum_{i=1}^{N} A_i(\mathcal{C}), \tag{4}$$

where $\mathbb{I}[\cdot]$ is the indicator function. Here, A_i measures agreement within demonstration i, while A averages agreement across demonstrations. The framework uses A_i for demonstration selection and A for update acceptance; neither numerical score is supplied to the context optimizer.

Diagnosing discrepancies and proposing context updates. Starting from $\mathcal{C}^{(0)}$, we perform R refinement iterations. At iteration $r \in \{0, \ldots, R-1\}$, we select a demonstration with the lowest $A_i(\mathcal{C}^{(r)})$. The optimizer receives $\mathcal{C}^{(r)}$ and the demonstration's complete offline annotation and causal online prediction sequences, paired at every planning timestep.

The prompt asks the optimizer to locate divergent intervals and identify premature or delayed online transitions relative to the offline annotations. It then asks the optimizer to infer likely causes under online observation constraints: which information available offline is missing from online decisions, and which task-specific cues or transition criteria the context leaves implicit. Based on this diagnosis, the optimizer proposes a candidate context $\widetilde{\mathcal{C}}^{(r)}$ that makes these details explicit and clarifies how available observations and memory should guide transitions. The candidate preserves $\mathcal{Y}_\ell$ and label meanings to remain compatible with low-level policy training. Replay details, demonstration selection, and a refinement example are provided in Section B.4.

Accepting context updates through replanning. An update proposed from one demonstration may not improve agreement on the others. We therefore replan over all N demonstrations with $\widetilde{\mathcal{C}}^{(r)}$ under

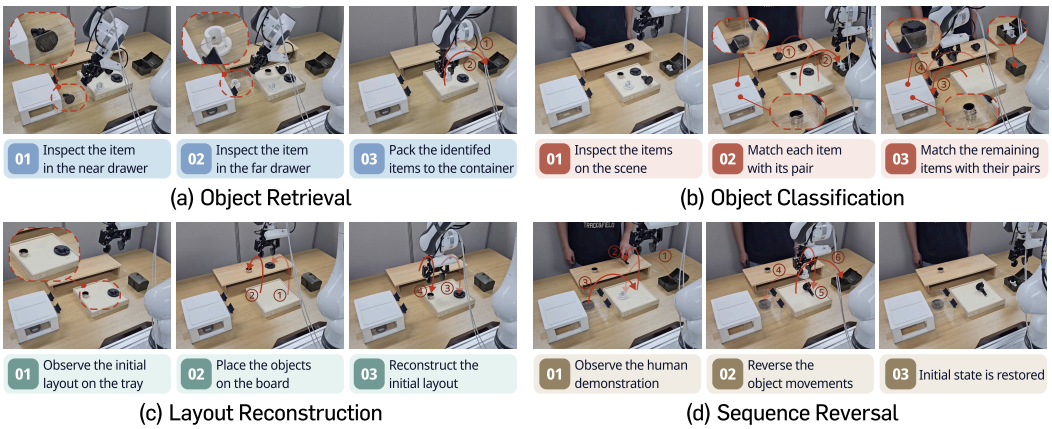

Figure 4: **Real-world evaluation tasks.** We design four long-horizon manipulation tasks to test whether task-specific context helps the high-level planner use earlier visual observations and select appropriate subtask transitions during execution. Representative frames show successive stages of each task and the changing visual evidence available for subsequent subtask decisions.

the same online observation constraints and recompute agreement with the fixed offline annotations. The update is accepted only if measured mean agreement strictly improves:

$$\mathcal{C}^{(r+1)} = \begin{cases} \widetilde{\mathcal{C}}^{(r)}, & \text{if } A(\widetilde{\mathcal{C}}^{(r)}) > A(\mathcal{C}^{(r)}), \\ \mathcal{C}^{(r)}, & \text{otherwise.} \end{cases} \tag{5}$$

Thus, each update is evaluated across the task's replay subset while the offline annotations and both VLMs remain fixed. At deployment, the high-level planner uses $\mathcal{C}^{(R)}$ to guide the low-level policy trained on annotations generated with $\mathcal{C}^{(0)}$.

4 Experiments

We investigate the following questions:

- How broadly does TASCO improve long-horizon execution across high-level planners, low-level policies, and simulation and real-robot tasks? (Tables 1 and 2 in Section 4.2)
- Does grounding task-specific context in a reference demonstration improve offline annotation consistency and downstream execution? (Table 3(a) and Figure 6 in Section 4.3)
- Does context refinement reduce the mismatch between offline annotations and online planning and further improve execution? (Tables 3(b), 3(c) and Figure 6 in Section 4.3)

4.1 Experimental setup

Benchmarks and metrics. In simulation, we evaluate TASCO on RMBench (9 tasks; Chen et al., 2026b) and a 5-task subset of the RoboDojo *Long-Horizon* suite (Chen et al., 2026a), reporting mean success rate (%) on both benchmarks (see Figure 8). We evaluate each simulated task over 50 trials. For RoboDojo, we additionally report its benchmark-defined partial-credit score in parentheses; the evaluated subset excludes tasks with 0% success for the flat low-level policy due to insufficient low-level execution precision. In real-robot evaluation, we use the single-arm platform with a gripper shown in Figure 7 to perform four long-horizon manipulation tasks illustrated in Figure 4. Each task requires visual reasoning and repeated subtask selection. We report task-specific partial-credit scores (%) averaged over 12 trials per task and an unweighted average across the four tasks.

Models and baselines. In simulation, we pair API-based Gemini 3.7 Flash (Google DeepMind, 2026a) and GPT-5.6 Luna (OpenAI, 2026a) high-level planners with $\pi_{0.5}$ (Physical Intelligence, 2025) and RLDX-1 (Kim et al., 2026) low-level policies. The real-robot evaluation uses Gemini 3.7 Flash as the high-level planner and $\pi_{0.5}$ as the low-level policy. Claude Opus 5 (Anthropic, 2026) serves as the context optimizer in both settings. Our baselines include a flat low-level policy trained

Table 1: **Simulation benchmark results across high-level planners and low-level policies.** We compare hierarchical policies with API-based GPT-5.6 Luna or Gemini 3.7 Flash high-level planners against flat and fine-tuned Qwen3-VL-8B baselines, using $\pi_{0.5}$ and RLDX-1 as low-level policies. We report mean success rate (%) on RMBench (9 tasks) and RoboDojo *Long-Horizon* (5-task subset); for RoboDojo, the benchmark-defined partial-credit score is shown in parentheses. Qwen3-VL-8B is fine-tuned on *Baseline* annotations generated by Gemini 3.7 Flash. TASCO and Refined-TASCO use induced and refined contexts, respectively, with identical TASCO training annotations. **Bold** and underline indicate best and runner-up success rates within each column, respectively.

High-Level Planner	Subtask Annotations	RMBench		RoboDojo (Long-Horizon†)	
		$\pi_{0.5}$	RLDX-1	$\pi_{0.5}$	RLDX-1
None	N/A	18.0	22.7	22.8 (31.4)	4.0 (7.5)
Qwen3-VL-8B	Baseline	19.3	13.8	18.8 (27.2)	5.6 (9.3)
GPT-5.6 Luna	Baseline	17.3	12.0	22.0 (27.9)	5.6 (10.8)
GPT-5.6 Luna + TASCO	TASCO	37.8	23.6	18.4 (28.0)	11.2 (14.8)
GPT-5.6 Luna + Refined-TASCO	TASCO	40.9	29.8	25.6 (34.3)	12.0 (15.6)
Gemini 3.7 Flash	Baseline	29.6	38.4	23.2 (32.0)	7.2 (11.1)
Gemini 3.7 Flash + TASCO	TASCO	<u>44.2</u>	<u>48.2</u>	<u>31.2</u> (38.1)	<u>12.8</u> (17.4)
Gemini 3.7 Flash + Refined-TASCO	TASCO	**51.8**	**53.1**	**34.8** (41.3)	**14.4** (19.4)

†Excludes tasks with 0% flat-policy success due to imprecise execution.

without subtask annotations and hierarchical policies using either a fine-tuned Qwen3-VL-8B (Bai et al., 2025) or an API-based frontier VLM as the high-level planner. For the API-based *Baseline*, a human-labeled subtask set, following the use of canonical plans by Gao et al. (2026), guides VLM-generated offline annotation and online planning. The Qwen3-VL-8B high-level planner is fine-tuned on Baseline annotations generated by Gemini 3.7 Flash. Model implementations, inference settings, and training hyperparameters are provided in Section B.2.

Task-specific context optimization. For each task, we randomly select one reference demonstration from its training set to induce $\mathcal{C}^{(0)}$. The induction-only TASCO variant uses this context for both offline annotation and online planning; Refined-TASCO instead uses $\mathcal{C}^{(R)}$ online while retaining the same offline annotations and trained low-level policy. Their comparison thus isolates the effect of context refinement. Unless otherwise specified, refinement replays $N = 3$ training demonstrations for $R = 2$ iterations per task. We accept an update only if mean subtask agreement with the fixed offline annotations strictly improves, as specified in Equations (4) and (5).

4.2 Main Results

Simulation benchmarks. As shown in Table 1, context induction improves RMBench success over Baseline by 9.8–20.5%p across all four settings. Together with the offline annotation analysis below, these gains suggest that more consistent subtask annotations improve supervision for subtask-conditioned low-level policies. With the trained low-level policies fixed, online context refinement adds 3.1–7.6%p, reaching 53.1% with Gemini 3.7 Flash and RLDX-1. Both variants outperform the flat low-level policies in every RMBench setting. On RoboDojo, Refined-TASCO exceeds Baseline by 3.6–11.6%p across all four settings. Even where induction lowers success under GPT-5.6 Luna with $\pi_{0.5}$, refinement raises it above Baseline. Across both benchmarks, refinement improves upon induction in every setting; Gemini 3.7 Flash with Refined-TASCO achieves the highest success rates for both low-level policies (see full task-wise results in Section C).

Real-robot evaluation. As shown in Table 2, the preset-based hierarchical policy with Gemini 3.7 Flash as the high-level planner achieves a mean partial-credit score of 23.8%. Context induction raises the score to 46.2%, and refinement increases it to 75.5% with the low-level policy held fixed. Refined-TASCO outperforms the flat low-level policy and the hierarchy with a fine-tuned Qwen3-VL-8B high-level planner, achieving the highest score on each task. On *Layout Reconstruction*, induction improves the score only slightly, whereas refinement raises it from 45.8% to 77.0%. This highlights the benefit of aligning online planning with the annotations used to train the low-level policy.

Qualitative rollout. Figure 5 shows an *Object Retrieval* rollout, in which the high-level planner inspects two drawer compartments before retrieving their matching objects from the center plate. The induced context gives visually similar candidates distinct names, such as “black bracket” and “black

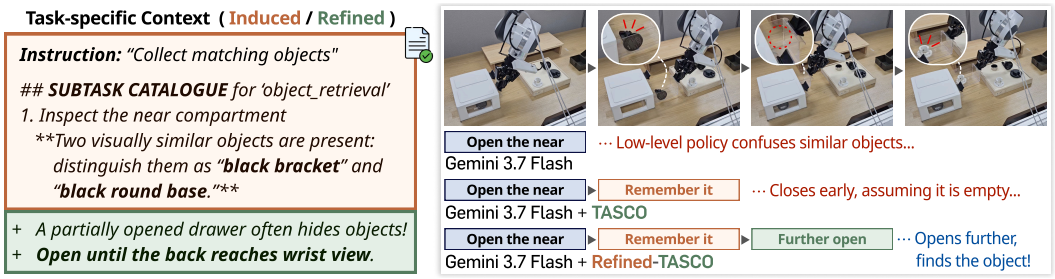

Figure 5: **Qualitative effect of context induction and refinement.** Left: TASCO adds detailed task context from a reference demonstration, while Refined-TASCO adds an observable criterion requiring the drawer contents to be visible before the opening subtask ends. Right: video frames and predicted subtask sequences contrast a premature transition with the corrected rollout.

Table 2: **Real-world evaluation results.** We compare hierarchical policies with an API-based Gemini 3.7 Flash high-level planner against flat and fine-tuned Qwen3-VL-8B baselines, all using $\pi_{0.5}$ as the low-level policy. We report mean scores (%) over 12 trials per task on 4 real-world tasks, using task-specific partial-credit metrics. Qwen3-VL-8B is fine-tuned on baseline annotations from Gemini 3.7 Flash. TASCO and Refined-TASCO share low-level training annotations, isolating the effect of context refinement. **Bold** and underline indicate best and runner-up results per column, respectively.

High-Level Planner	Subtask Annotations	Object Retrieval	Object Classification	Layout Reconstruction	Sequence Reversal	Average
None	N/A	25.0	8.3	35.4	11.1	20.0
Qwen3-VL-8B	Baseline	41.7	14.5	43.8	11.1	27.8
Gemini 3.7 Flash	Baseline	25.0	12.5	43.8	13.9	23.8
Gemini 3.7 Flash + TASCO	TASCO	<u>58.3</u>	<u>33.3</u>	<u>45.8</u>	<u>47.2</u>	<u>46.2</u>
Gemini 3.7 Flash + Refined-TASCO	TASCO	**91.7**	**58.3**	**77.0**	**75.0**	**75.5**

round base", providing consistent object references to the low-level policy. During online planning, however, the high-level planner closes a partly opened drawer and mistakes its obscured contents for an empty compartment. Refinement requires the drawer interior to be visible before closing, allowing the object to be identified and the retrieval sequence to continue. Together, these changes illustrate how consistent object references and observable transition criteria support long-horizon execution.

4.3 Ablation Studies

We further examine how reference demonstrations and induced annotations affect execution, then analyze transition consistency and the effects of refinement iterations, update acceptance, and replay set size. Unless otherwise specified, refinement ablations use Gemini 3.7 Flash as the high-level planner and $\pi_{0.5}$ as the low-level policy.

Effect of the reference demonstration. To assess reference grounding, we replace the reference video with the human-labeled subtask preset used by Baseline, while still inducing a context for offline annotation and online planning. A subtask preset specifies the available labels but not how their boundaries appear in the task's visual trajectory. As shown in Table 3(a), using the reference video improves success by 4.6–10.7%p across all four settings. Although context induced from the instruction and subtask preset improves upon Baseline, the consistent gap shows the value of execution details provided by the reference demonstration. Separately, an instruction-only setting omits the subtask preset during offline annotation and online planning. It achieves mean success of 20.9% across the four combinations, compared with 24.3% for Baseline, although it performs better with $\pi_{0.5}$ under both high-level planners (see Table 6 in Section D).

Annotations for a fine-tuned high-level planner. To test whether TASCO annotations also benefit a hierarchy with a locally fine-tuned high-level planner, we train Qwen3-VL-8B and each low-level policy on either Baseline or TASCO annotations. Training on TASCO annotations improves RMBench success by 6.9%p with $\pi_{0.5}$ and 10.9%p with RLDX-1. These gains show that the induced annotations benefit hierarchical policy training beyond the frontier VLM that generated them.

Table 3: **Ablations of context induction and refinement.** Mean success rate (%) across the 9 RMBench tasks. (a) GPT-5.6 Luna and Gemini 3.7 Flash use a human-labeled subtask preset (no demo) or a reference demonstration (demo) to induce contexts for annotation and planning. (b,c) Online refinement uses Gemini 3.7 Flash and $\pi_{0.5}$, accepting updates only when task-wise tick agreement improves. Bold marks the highest result within each comparison.

(a) Reference demonstration.			(b) Refinement iterations.			(c) Number of demos.		
High-level Planner	$\pi_{0.5}$	RLDX-1	Iterations	# Demos	SR (%)	Iterations	# Demos	SR (%)
GPT-5.6 Luna (w/o demo)	27.1	17.3	0	—	44.2	2	1	44.9
GPT-5.6 Luna	**37.8**	**23.6**	1	3	44.9	2	3	**51.8**
Gemini 3.7 Flash (w/o demo)	36.4	43.6	2	3	51.8	2	5	49.3
Gemini 3.7 Flash	**44.2**	**48.2**	3	3	**52.0**	2	10	40.0

Transition consistency. Simulator-defined subtask boundaries provide a consistent temporal reference across episodes without assuming uniquely correct transition times. As shown in Figure 6, context induction narrows the offline offset distribution relative to Baseline, consistent with more uniform annotation timing. For both Baseline and TASCO, online subtask transitions tend to occur earlier than their respective offline annotations. That shift persists after induction, suggesting that the induced context alone does not fully resolve online uncertainty about transition timing. Refinement moves the online transition distribution closer to that of the fixed TASCO offline annotations, consistent with reduced early-transition bias. This distributional analysis complements per-timestep subtask agreement used during refinement.

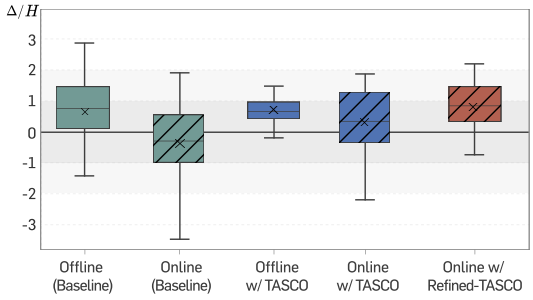

Figure 6: **Subtask transition offsets relative to simulator-based oracle labels.** Box plots show offline annotation and online planning offsets for Baseline, TASCO, and Refined-TASCO. Positive offsets indicate later transitions, and one tick corresponds to 0.8 s. Insets report the standard deviation and offline–online bias. Refined-TASCO shares the offline annotations with TASCO.

Refinement iterations and acceptance. As shown in Table 3(b), the mean success rate increases from 44.2% without refinement to 44.9% after one iteration and 51.8% after two iterations. A third iteration yields only a marginal increase to 52.0%. After two iterations, accepting every proposed update yields a 44.0% success rate, whereas acceptance based on task-wise tick agreement reaches 51.8%. With the trained low-level policy fixed across these comparisons, this difference supports evaluating proposed context updates before using them for online planning.

Number of demonstrations for refinement. As shown in Table 3(c), success with two refinement iterations peaks at 51.8% using three replayed demonstrations but falls to 40.0% when the replay set includes ten demonstrations. This decline shows that a larger replay set does not guarantee better online task performance, even when update acceptance is based on subtask agreement across that set.

5 Conclusion

To our knowledge, this work is the first to systematically analyze and address subtask inconsistency across offline annotation and online planning in hierarchical robot policies built on frozen frontier VLMs. Building on this analysis, we propose TASCO, a framework that uses a separate VLM as a context optimizer to induce and refine task-specific context. By grounding task decomposition in a reference demonstration and refining the resulting context through measured offline–online agreement, TASCO promotes consistent annotations and aligns online planning with low-level policy training. Experiments across simulation and real-robot benchmarks demonstrate substantial gains across diverse hierarchical policies and consistent benefits from context refinement. These findings suggest that addressing task-specific ambiguity and temporal mismatch can help robot policies translate the growing reasoning capabilities of frontier VLMs into more reliable long-horizon execution.

AI USE STATEMENT

In this work, we used generative AI tools to assist with drafting and polishing the manuscript and to generate portions of the research code. The research ideas, hypotheses, methodology, and experimental design were developed by the authors without generative AI assistance. We have reviewed all AI-assisted work. For example, all LLM-generated code was directly reviewed and verified for correctness by every author, and all AI-assisted text was reviewed and revised by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide full hyperparameter and implementation details in Section 4.1 and Section B.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *ICLR*, volume 2026, pp. 8479–8565, 2026.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Anthropic. Claude opus 5 system card, July 2026. URL https://anthropic.com/claude-opus-5-system-card. System Card.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Tianxing Chen, Yue Chen, Zixuan Li, Junyuan Tang, Kailun Su, Haoran Lu, Weijie Wan, Baijun Chen, Songling Liu, Haowen Yan, et al. Robodojo: A unified sim-and-real benchmark for comprehensive evaluation of generalist robot manipulation policies. *arXiv preprint arXiv:2607.04434*, 2026a.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, et al. Rmbench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv preprint arXiv:2603.01229*, 2026b.

William Chen, Jagdeep Singh Bhatia, Catherine Glossop, Nikhil Mathihalli, Ria Doshi, Andy Tang, Danny Driess, Karl Pertsch, and Sergey Levine. Steerable vision-language-action policies for embodied reasoning and hierarchical control. *arXiv preprint arXiv:2602.13193*, 2026c.

Chongkai Gao, Zixuan Liu, Zhenghao Chi, Junshan Huang, Xin Fei, Yiwen Hou, Yuxuan Zhang, Yudi Lin, Zhirui Fang, and Lin Shao. Vla-os: Structuring and dissecting planning representations and paradigms in vision-language-action models. In *NeurIPS*, volume 38, pp. 136705–136736, 2026.

Google DeepMind. Gemini 3.7 flash, August 2026a. URL https://deepmind.google/models/model-cards/gemini-3-7-flash/. Model Card.

Google DeepMind. Gemini 3.8 flash, September 2026b. URL https://deepmind.google/models/model-cards/gemini-3-8-flash/. Model Card.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Yu-Chiang Frank Wang, and Fu-En Yang. Thinkact: Vision-language-action reasoning via reinforced visual latent planning, 2025. In *NeurIPS*, 2025.

Li Ji, Siyin Wang, Pengfang Qian, Xiaopeng Yu, Yihai Tian, Zhaoye Fei, Jingjing Gong, and Xipeng Qiu. Hime: Hierarchical embodied memory for long-horizon vision-language-action control. *arXiv preprint arXiv:2607.03449*, 2026.

Dongyoung Kim, Huiwon Jang, Myungkyu Koo, Suhyeok Jang, Taeyoung Kim, Beomjun Kim, Byungjun Yoon, Changsung Jang, Daewon Choi, Dongsu Han, et al. Rldx-1 technical report. *arXiv preprint arXiv:2605.03269*, 2026.

Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Raymond Yu, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, et al. Hamster: Hierarchical action models for open-world robot manipulation (2025). In *ICLR*, 2025.

Weiyu Liu, Neil Nie, Ruohan Zhang, Jiayuan Mao, and Jiajun Wu. Learning compositional behaviors from demonstration and language. In *CoRL*, 2024.

OpenAI. GPT-5.6 system card, July 2026a. URL https://deploymentsafety.openai.com/gpt-5-6. System Card.

OpenAI. GPT-6 Astra system card, September 2026b. URL https://deploymentsafety.openai.com/gpt-6-astra. System Card.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *EMNLP*, pp. 9340–9366, 2024.

Physical Intelligence. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *CoRL*, 2025.

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

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *ICRA*, 2023.

SpaceXAI. Introducing grok 4.6, August 2026. URL https://x.ai/news/grok-4-6. Technical Report.

Beichen Wang, Juexiao Zhang, Shuwen Dong, Irving Fang, and Chen Feng. Vlm see, robot do: Human demo video to robot action plan via vision language model. In *IROS*, pp. 17215–17222. IEEE, 2025.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *ICLR*, volume 2024, pp. 12028–12068, 2024.

Tianshuo Yang, Guanyu Chen, Yutian Chen, Zhixuan Liang, Yitian Liu, Zanxin Chen, Chunpu Xu, Haotian Liang, Jiangmiao Pang, Yao Mu, et al. Hivla: A visual-grounded-centric hierarchical embodied manipulation system. *arXiv preprint arXiv:2604.14125*, 2026.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Kaidong Zhang, Rongtao Xu, Pengzhen Ren, Junfan Lin, Hefeng Wu, Liang Lin, and Xiaodan Liang. Robridge: A hierarchical architecture bridging cognition and execution for general robotic manipulation. In *ICCV*, pp. 14590–14601. IEEE, 2025.

Qihang Zhang, Lin Li, Luyao Zhang, Shuai Yang, Yiming Luo, Shuaiting Li, Ruilin Wang, Junke Wang, Jiahao Shao, Gangwei Xu, et al. Native video-action pretraining for generalizable robot control. *arXiv preprint arXiv:2607.08639*, 2026.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *ICLR*, 2023.

A Subtask Transition Experiments

This section details the protocols used in Figure 2 to isolate the effects of inconsistent subtask transitions in offline annotation and online planning on task success, and to measure offline–online discrepancies in frontier VLMs. For Figure 2(a,b), we define oracle subtasks and their transition criteria from RMBench simulator states. We train $\pi_{0.5}$ on the full training set of 50 demonstrations per task, with one high-level planning interval, or tick, equal to $H = 25$ frames. Each condition is evaluated on 25 episodes per task across nine tasks, and task success is averaged across tasks. We obtain 95% percentile intervals from 20,000 task-stratified bootstrap draws: each draw resamples 25 binary outcomes within each task before averaging the nine task success rates.

For Figure 2(a), we perturb the training annotations by independently shifting each oracle subtask transition by a uniformly sampled offset bounded by $a \in \{0, 12, 25, 37\}$ frames. These bounds are displayed as approximately $\{0, 0.5, 1, 1.5\}$ ticks. The subtask sequence, observations, and demonstrated actions remain unchanged. We train a separate low-level policy for each jitter bound, then evaluate every policy with the unperturbed oracle planner. Thus, the conditions differ in their training annotations, not their online planning rule.

For Figure 2(b), we instead keep the policy trained on unperturbed oracle annotations fixed and apply a constant offset $\Delta/H \in \{-2, -1, 0, 1, 2\}$ to the oracle planner's transitions during rollout. The same offset is applied to every transition: negative values provide the next subtask early, whereas positive values retain the current subtask longer. Task success is measured from the resulting rollouts.

For Figure 2(c), each frontier VLM listed in Table 4 receives the oracle subtask preset without task-specific context; provider IDs and inference settings are given in Section B.2. It annotates a demonstration with access to the full trajectory, then predicts subtasks at the same ticks with future frames withheld. A transition is the first tick of a new nonempty subtask label; consecutive identical labels are collapsed, and the initial label is excluded. We match each offline transition to the first subsequent online entry into the same subtask after the previous match, preserving transition order. The plotted offset is $(t_{\text{online}} - t_{\text{offline}})/H$ for matched transitions only; unmatched offline transitions and surplus online transitions are excluded. Boxes show interquartile ranges, center lines show medians, crosses show means, and whiskers extend to the most extreme observations within 1.5 interquartile ranges of the quartiles.

B Implementation Details

B.1 Real-Robot Hardware

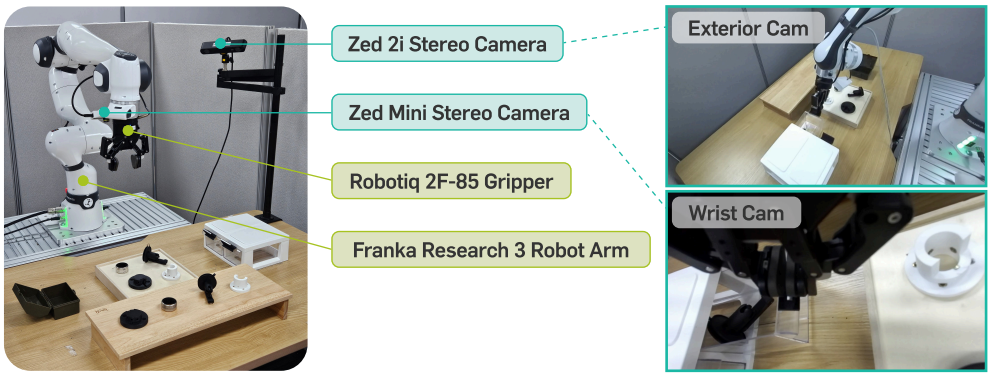

Figure 7: **Real-robot platform and camera views.** (a) Franka Research 3 arm, Robotiq 2F-85 gripper, and two stereo cameras. (b) Exterior and wrist-camera observations.

Our real-robot experiments use a Franka Research 3 arm equipped with a Robotiq 2F-85 gripper. A ZED 2i stereo camera captures the workspace from an exterior viewpoint, while a wrist-mounted ZED Mini stereo camera provides a close-up view of object interactions.

B.2 Model Details

Table 4: **API-based VLMs.** Provider model IDs and their primary roles in our experiments.

Primary role	Model	Provider ID
Context optimizer	Claude Opus 5	`bedrock/global.anthropic.claude-opus-5`
High-level planner	Gemini 3.7 Flash	`gemini/gemini-3.7-flash`
	GPT-5.6 Luna	`openai/gpt-5.6-luna`
Subtask transition analysis	GPT-6 Astra	`openai/gpt-6-astra`
	Gemini 3.8 Flash	`gemini/gemini-3.8-flash`
	Grok 4.6	`bedrock/global.xai.grok-4.6`
	GPT-5.6 Sol	`openai/gpt-5.6-sol`

API-based VLMs. Table 4 lists the provider IDs of the frontier VLMs used in our experiments. Claude Opus 5 serves as the context optimizer, while Gemini 3.7 Flash and GPT-5.6 Luna serve as the high-level planners in the main evaluations. The models marked *Subtask transition analysis* are evaluated in Section A. We set the temperature to 0.0 for all API calls and use `reasoning_effort=high` for context induction and refinement, and `reasoning_effort=low` for offline annotation and online planning.

Local high-level planner. We fine-tune Qwen3-VL-8B-Instruct (Bai et al., 2025)[1] using its released fine-tuning code[2]. We use LoRA with rank 32, scaling factor $\alpha = 64$, and dropout 0.

Low-level policies. We initialize $\pi_{0.5}$ (Physical Intelligence, 2025)[3] and RLDX-1 (Kim et al., 2026)[4] from their respective checkpoints. We fine-tune them using the LeRobot implementation[5] and the released RLDX-1 implementation[6], respectively. We follow each implementation's training procedure, with optimization settings summarized in Table 5.

Table 5: **Fine-tuning hyperparameters.** We report settings for the local high-level planner (Qwen3-VL-8B-Instruct) and the low-level policies ($\pi_{0.5}$ and RLDX-1). All three use AdamW with $\epsilon = 10^{-8}$ and a cosine learning-rate schedule.

Hyperparameter	Qwen3-VL-8B-Instruct	$\pi_{0.5}$	RLDX-1
Batch size	16	64	64 (RMBench) / 128 (RoboDojo)
Training steps	10,000	60,000	60,000
AdamW (β_1, β_2)	(0.9, 0.999)	(0.9, 0.95)	(0.9, 0.999)
Weight decay	0	0.01	10^{-5}
Learning rate	10^{-5}	2.5×10^{-5}	10^{-4}
Warmup steps	300	1,000	3,000

B.3 Hierarchical Policy Execution

Visual keyframes. The high-level planner can supplement textual subtask labels with visual keyframes showing object appearance or spatial layout. To support retrieval, it marks observed frames for storage with captions and generates queries to retrieve relevant stored keyframes. These auxiliary outputs are omitted from Equation (1), which shows only the subtask label and updated textual memory. Let $\mathbf{v}_t$ denote the keyframe retrieved at planning step t, with $\mathbf{v}_t = \varnothing$ when no keyframe is supplied.

Low-level action prediction. The textual subtask label and optional visual keyframe together provide multimodal guidance for low-level control. This guidance is updated at high-level planning steps and

[1] https://huggingface.co/Qwen/Qwen3-VL-8B-Instruct.
[2] https://github.com/QwenLM/Qwen3-VL/tree/main/qwen-vl-finetune.
[3] https://huggingface.co/lerobot/pi05_base.
[4] https://huggingface.co/RLWRLD/RLDX-1-PT.
[5] https://github.com/huggingface/lerobot/tree/main/src/lerobot/policies/pi05.
[6] https://github.com/RLWRLD/RLDX-1.

Table 6: **Effect of task-specific context on RMBench.** Mean success rate (%) over 9 tasks; Avg. averages the two low-level policy columns. Instruction-only uses the global task instruction for offline annotation and online planning. Baseline additionally provides a human-labeled subtask set, analogous to the canonical plans used by Gao et al. (2026), whereas TASCO uses context induced from the instruction and a reference demonstration. Each low-level policy is trained on annotations generated under its corresponding condition. **Bold** and underline indicate best and runner-up results within each high-level planner and metric.

		RMBench SR (%)		
High-Level Planner	Subtask Annotations	$\pi_{0.5}$	RLDX-1	Avg.
GPT-5.6 Luna	Instruction-only	<u>24.0</u>	8.9	<u>16.5</u>
	Baseline (canonical subtasks)	17.3	<u>12.0</u>	14.7
	TASCO	**37.8**	**23.6**	**30.7**
Gemini 3.7 Flash	Instruction-only	<u>30.7</u>	20.0	25.4
	Baseline (canonical subtasks)	29.6	<u>38.4</u>	<u>34.0</u>
	TASCO	**44.2**	**48.2**	**46.2**

held fixed between them. Each low-level invocation therefore uses the latest available pair:

$$z_t = \big(y_{\tau(t)}, \mathbf{v}_{\tau(t)}\big), \qquad \tau(t) = \max\{u \in \mathcal{T} : u \le t\}, \tag{6}$$

where $\mathcal{T}$ is the set of high-level planning timesteps and $\tau(t)$ identifies the most recent planning step. The low-level policy then predicts a chunk of k actions conditioned on this guidance:

$$[\mathbf{a}_t, \ldots, \mathbf{a}_{t+k-1}] = \pi_\theta(\mathbf{o}_t, \mathbf{s}_t, z_t), \tag{7}$$

where $\mathbf{o}_t$ and $\mathbf{s}_t$ denote the visual observations and robot's proprioceptive state at timestep t, respectively. For training, offline subtask labels and optional retrieved keyframes are paired with recorded observations, proprioceptive states, and expert actions.

B.4 Context Refinement Details

Causal demonstration replay. We reuse the offline annotations generated with $\mathcal{C}^{(0)}$ for the fixed subset of N training demonstrations. For each evaluated context $\mathcal{C}$, the planner processes recorded observations chronologically at the same annotation timesteps $\mathcal{T}_i$. At each step, it follows Equation (1), accessing only recent observations up to that step and its preceding textual memory. Memory is initialized separately for each replay, with an initial memory replacing m_{t^-} at the first planning step. Subsequent memory is generated from the planner's own online outputs under $\mathcal{C}$. Because the observations come from recorded trajectories, predictions do not alter subsequent observations.

Agreement weighting. The mean agreement A in Equation (4) gives each demonstration equal weight regardless of duration. By comparing canonical subtask labels at corresponding timesteps, the score evaluates the textual component of guidance and is sensitive to both subtask selection and transition timing.

Demonstration selection. At iteration r, the demonstration selected for diagnosis has index

$$i_r^\star \in \operatorname*{arg\,min}_{i \in \{1,\ldots,N\}} A_i(\mathcal{C}^{(r)}). \tag{8}$$

Here, $\star$ marks a selected minimizer, and $i_r^\star$ indexes the demonstration used for diagnosis. The context optimizer receives $\mathcal{C}^{(r)}$ and the complete offline annotation and causal online prediction sequences for demonstration $i_r^\star$. These sequences cover both matching and mismatched intervals, paired at every planning timestep, so discrepancies can be interpreted within the full subtask sequence. The numerical agreement scores are used by the framework for selection and acceptance but are not supplied to the optimizer.

Example context refinement. Suppose the fixed annotation protocol marks drawer opening as complete when its contents become visible. If the paired label sequences show premature online transitions, the optimizer may propose an explicit instruction to verify this cue in the wrist-camera view before switching subtasks. The revision makes an observable completion cue explicit while

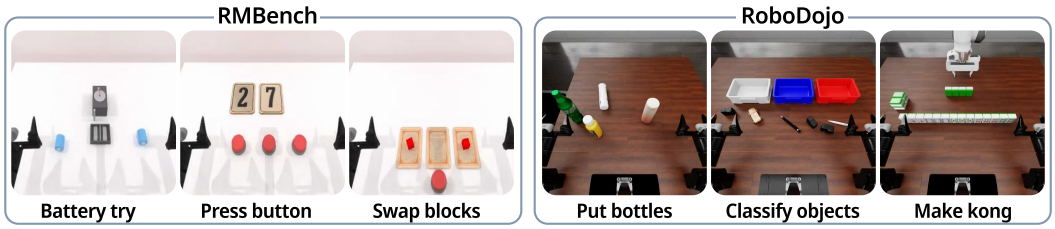

Figure 8: **Representative tasks from the simulation benchmarks.** Left: *Battery Try*, *Press Button*, and *Swap Blocks* from RMBench. Right: *Put Bottles Into Dustbin*, *Classify Objects*, and *Make Kong* from RoboDojo *Long-Horizon*. These examples are drawn from the nine RMBench tasks and five RoboDojo tasks evaluated.

Table 7: **Task-wise RMBench performance with $\pi_{0.5}$.** Results report success rate (%) for each task. Induced and Refined correspond to TASCO and Refined-TASCO in Table 1, respectively. Intermediate averages cover the first five and last four tasks; the overall average covers all nine tasks.

	Flat	Qwen3-VL-8B	GPT-5.6 Luna			Gemini 3.7 Flash		
Task	N/A	Baseline	Baseline	Induced	Refined	Baseline	Induced	Refined
Observe and Pick Up	18.0	32.0	20.0	40.0	44.0	18.0	54.0	56.0
Rearrange Blocks	32.0	40.0	60.0	80.0	88.0	18.0	80.0	86.0
Put Back Block	6.0	16.0	8.0	20.0	48.0	34.0	28.0	78.0
Swap Blocks	24.0	20.0	20.0	48.0	44.0	46.0	52.0	46.0
Swap T	16.0	26.0	16.0	76.0	64.0	60.0	78.0	76.0
M(1) Average	19.2	26.8	24.8	52.8	57.6	35.2	58.4	68.4
Battery Try	26.0	34.0	24.0	40.0	28.0	26.0	50.0	54.0
Blocks Ranking Try	36.0	0.0	4.0	4.0	8.0	0.0	6.0	6.0
Cover Blocks	4.0	6.0	4.0	32.0	44.0	64.0	48.0	52.0
Press Button	0.0	0.0	0.0	0.0	0.0	0.0	2.0	12.0
M(n) Average	16.5	10.0	8.0	19.0	20.0	22.5	26.5	31.0
Overall average	18.0	19.3	17.3	37.8	40.9	29.6	44.2	51.8

preserving the existing subtask labels and their meanings. To assess its effect beyond the selected demonstration, the framework replays all N demonstrations under the same online observation constraints. The candidate is retained only if mean agreement with the fixed offline annotations strictly improves, as specified in Equation (5).

C Task-wise Simulation Results

Figure 8 shows three tasks from RMBench and three from the RoboDojo *Long-Horizon* subset. Tables 7 to 10 report full task-wise results for the settings summarized in Table 1. Flat denotes a low-level policy without a high-level planner; Qwen3-VL-8B is fine-tuned on *Baseline* annotations. For the API-based high-level planners, Baseline uses the human-labeled subtask preset, Induced uses TASCO, and Refined uses Refined-TASCO. Each task is evaluated over 50 trials.

We evaluate five RoboDojo *Long-Horizon* tasks, excluding the following: `fill_egg_holder`, `organize_table`, and `play_stacking_toy`. The official RoboDojo leaderboard reports 0.0% success for $\pi_{0.5}$ on these three tasks.[7] We exclude them because their low-level execution demands would obscure differences in high-level planning.

D Instruction-Only Annotation and Planning

We examine an instruction-only setting without the human-labeled subtask preset used by Baseline. The high-level planner receives only the global task instruction during both offline annotation and

[7] https://robodojo-benchmark.com/leaderboard

Table 8: **Task-wise RMBench performance with RLDX-1.** Results report success rate (%) for each task. Induced and Refined correspond to TASCO and Refined-TASCO in Table 1, respectively. Intermediate averages cover the first five and last four tasks; the overall average covers all nine tasks.

	Flat	Qwen3-VL-8B	GPT-5.6 Luna			Gemini 3.7 Flash		
Task	N/A	Baseline	Baseline	Induced	Refined	Baseline	Induced	Refined
Observe and Pick Up	0.0	6.0	12.0	4.0	8.0	4.0	12.0	14.0
Rearrange Blocks	52.0	64.0	8.0	92.0	96.0	92.0	100.0	100.0
Put Back Block	16.0	14.0	0.0	8.0	28.0	66.0	90.0	98.0
Swap Blocks	4.0	2.0	4.0	0.0	8.0	26.0	20.0	12.0
Swap T	18.0	26.0	24.0	40.0	44.0	44.0	96.0	92.0
M(1) Average	18.0	22.4	9.6	28.8	36.8	46.4	63.6	63.2
Battery Try	42.0	12.0	28.0	40.0	44.0	6.0	32.0	44.0
Blocks Ranking Try	62.0	0.0	0.0	0.0	0.0	14.0	26.0	10.0
Cover Blocks	8.0	0.0	32.0	28.0	40.0	94.0	58.0	90.0
Press Button	2.0	0.0	0.0	0.0	0.0	0.0	0.0	18.0
M(n) Average	28.5	3.0	15.0	17.0	21.0	28.5	29.0	40.5
Overall average	22.7	13.8	12.0	23.6	29.8	38.4	48.2	53.1

Table 9: **Task-wise RoboDojo *Long-Horizon* performance with $\pi_{0.5}$.** Results cover the evaluated five-task subset. Each cell reports success rate (%); the benchmark-defined partial-credit score appears in parentheses. Induced and Refined correspond to TASCO and Refined-TASCO in Table 1, respectively.

	Flat	Qwen3-VL-8B	GPT-5.6 Luna			Gemini 3.7 Flash		
Task	N/A	Baseline	Baseline	Induced	Refined	Baseline	Induced	Refined
Fill Pen Holder	4.0 (17.1)	2.0 (22.4)	10.0 (18.5)	4.0 (14.8)	20.0 (33.6)	6.0 (23.4)	28.0 (37.0)	32.0 (40.0)
Classify Objects	14.0 (27.3)	2.0 (12.0)	10.0 (17.5)	12.0 (22.8)	12.0 (21.0)	12.0 (21.2)	8.0 (19.6)	8.0 (19.6)
Put Bottles Into Dustbin	70.0 (80.8)	72.0 (81.4)	70.0 (80.5)	52.0 (70.0)	68.0 (78.4)	72.0 (81.7)	90.0 (94.0)	90.0 (93.4)
Play Tic-Tac-Toe	0.0 (5.6)	0.0 (2.2)	0.0 (3.0)	4.0 (12.4)	0.0 (10.4)	2.0 (9.9)	2.0 (11.7)	2.0 (11.7)
Make Kong	26.0 (26.0)	18.0 (18.0)	20.0 (20.0)	20.0 (20.0)	28.0 (28.0)	24.0 (24.0)	28.0 (28.0)	42.0 (42.0)
Average	22.8 (31.4)	18.8 (27.2)	22.0 (27.9)	18.4 (28.0)	25.6 (34.3)	23.2 (32.0)	31.2 (38.1)	34.8 (41.3)

Table 10: **Task-wise RoboDojo *Long-Horizon* performance with RLDX-1.** Results cover the five-task subset. Cells report success rate (%); benchmark-defined partial-credit scores appear in parentheses. Induced and Refined denote TASCO and Refined-TASCO in Table 1, respectively.

	Flat	Qwen3-VL-8B	GPT-5.6 Luna			Gemini 3.7 Flash		
Task	N/A	Baseline	Baseline	Induced	Refined	Baseline	Induced	Refined
Fill Pen Holder	0.0 (3.7)	0.0 (1.3)	0.0 (2.0)	0.0 (0.8)	0.0 (3.2)	0.0 (1.8)	0.0 (3.0)	0.0 (2.4)
Classify Objects	0.0 (0.6)	0.0 (0.6)	0.0 (0.6)	0.0 (0.0)	0.0 (0.6)	0.0 (0.0)	0.0 (2.4)	0.0 (0.0)
Put Bottles Into Dustbin	4.0 (12.1)	6.0 (15.8)	4.0 (12.4)	4.0 (11.8)	4.0 (13.2)	0.0 (9.0)	12.0 (23.2)	8.0 (17.8)
Play Tic-Tac-Toe	0.0 (4.9)	0.0 (6.7)	0.0 (14.8)	0.0 (9.6)	0.0 (4.8)	4.0 (12.8)	4.0 (10.6)	8.0 (20.8)
Make Kong	16.0 (16.0)	22.0 (22.0)	24.0 (24.0)	52.0 (52.0)	56.0 (56.0)	32.0 (32.0)	48.0 (48.0)	56.0 (56.0)
Average	4.0 (7.5)	5.6 (9.3)	5.6 (10.8)	11.2 (14.8)	12.0 (15.6)	7.2 (11.1)	12.8 (17.4)	14.4 (19.4)

online planning, and each low-level policy is trained on annotations generated under its corresponding condition. We compare this setting with Baseline and TASCO on RMBench.

As shown in Table 6, the instruction-only setting performs below Baseline with RLDX-1 but above it with $\pi_{0.5}$ under both high-level planners. Across the four evaluated settings, its mean success rate is 20.9%, compared with 24.3% for Baseline. TASCO achieves the highest success rate in every combination. Thus, the subtask preset provides a modest average benefit over instruction-only annotation and planning, but its effect is not uniform across low-level policies.

E LIMITATIONS

TASCO improves the consistency of subtask guidance, but its benefits remain limited by the low-level policy's manipulation precision. This limitation motivates the exclusion of three RoboDojo tasks on which the official leaderboard reports 0% success for $\pi_{0.5}$, as detailed in Section C. For tasks

with similarly demanding control requirements, better high-level planning alone may therefore be insufficient. API-based high-level planning also makes online execution sensitive to model response times. Developing seamless, low-latency interaction between frontier models and low-level policies could mitigate this delay while bringing the models' improving reasoning capabilities into robot control.

F Context Optimizer System Prompts

We provide the complete system prompts used for task-specific context induction and refinement below. Task-specific inputs are supplied separately as described in Sections 3.1 and 3.2.

F.1 Task-specific Context Induction

Context induction system prompt

```
You are a robot-manipulation task analyst.
You are given one task: its single-line instruction and one
    demonstration episode sampled to head-camera frames labelled with
    frame index and timestamp.
Write the complete task context as a single JSON object and nothing
    else.

The context is read by a separate annotator that, every 25 frames,
    sees only the instruction, its last subtask, its own memory text,
    the last four head-camera frames and a one-shot demonstration pack,
     and copies the current subtask verbatim from a menu built from
    your catalogue.
Only the labels, memory operations, done_when criteria and tick rules
    reach it; every rule you want obeyed must be written there.

The demonstration is one draw from the task's randomisation: encode
    every variation axis as a parameter, permutation, rotation, loop or
     optional entry so that the expanded catalogue covers every episode
    .
Name objects by the position they occupy when the step begins.
Store a keyframe only for a fact a later step needs that will no
    longer be visible, at most once per task.
Every done_when conjoins visible states (TRIGGER), what does not end
    the step (REJECT) and any ordering constraint (GATE), and names the
     acting gripper.
```

F.2 Task-specific Context Refinement

Context refinement system prompt

```
You refine the task context of an online planner.
At every tick (25 frames) the planner labels the tick with one subtask
     from the context's fixed catalogue, seeing only the last four head
    -camera frames, a one-shot demonstration pack and its own memory
    text, never the future.
The target is the offline annotation of the same episode, made with
    the whole episode visible; there is no ground truth, the offline
    labels are the reference.

You receive the task goal, the current context and the demonstration
    on which the planner agreed least with the offline labels: its
    video, its offline and online label timelines with every
    disagreement and the planner's memory, and the history of earlier
    rounds.
```

```
Classify each disagreement as EARLY, LATE, WRONG, MISSING or EXTRA,
    say what the planner could not know without future frames, and edit
     the context so that it is decided like the offline annotation from
     now on.
The catalogue labels are fixed; edit only the completion criteria,
    memory operations, sequence rule and tick rules, within the size
    budget, at most six catalogue updates and two rule rewrites per
    round; if the previous attempt was rolled back, change a different
    mechanism. Reply with one JSON object: the analysis, the updates
    and their rationale.
```