TRAJECTORY-AWARE EXPERIENCE RETRIEVAL WITH CONTRASTING SUCCESS AND FAILURE MEMORIES FOR EMBODIED VLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language models (VLMs) have improved embodied agents, but errors in perception, planning, and execution often accumulate in unseen environments. Existing memory-based methods do not fully address this limitation: they retrieve experiences using the task instruction and current observation, favoring goal-level similarity over the agent's actual execution history. Yet the guidance an agent needs depends on that history, as continuing a successful trajectory requires different guidance from recovering after an error. To address these limitations, we propose **Trajectory-aware Experience Retrieval for embodied Agents** (TERA), a training-free framework that retrieves guidance based on the trajectory executed so far. TERA summarizes each multimodal trajectory, infers failure causes directly from failed executions, and distills situation-conditioned guidance into separate success and failure memory banks. At inference time, TERA queries both banks using a summary of the current trajectory, enabling the planner to contrast guidance on how to proceed with guidance on what to avoid and thereby improve action selection. Across 16 VLM backbones, TERA improves the mean success rate on EB-Manipulation by up to 7.9% and improves Skill Match, Entity Match, and Exact Match on VLABench by up to 13.2%, 12.9%, and 9.4%, respectively. Ablation studies show that both trajectory-aware retrieval and the separation of success- and failure-derived guidance contribute to these gains.

1 INTRODUCTION

Recent advances in large language models (LLMs) and vision-language models (VLMs) have accelerated the development of embodied agents that interpret environments through textual representations and multimodal observations, respectively, follow language instructions, and generate actions for diverse tasks, including household tasks (Shridhar et al., 2020; 2021), gameplay (Fan et al., 2022; Wang et al., 2023), and robotic manipulation (Huang et al., 2023; Driess et al., 2023). Despite this progress, LLM agents access the environment only through textual descriptions, which omit much of the visual detail required for embodied execution. Even VLM agents remain brittle in unseen environments, where errors in perception, planning, and execution can accumulate. To mitigate this, prior work provides expert demonstrations of the target task in context (Yin et al., 2025; Sridhar et al., 2025) or reuses past execution experiences as external guidance for unseen tasks (Zhao et al., 2024; Sarch et al., 2024; Li et al., 2024).

Expert demonstrations in context show how to act in the environment, but when the target task differs in type, agents often produce incorrect actions or imitate the demonstration without its underlying intent (Zhou et al., 2025). Memory-based methods reuse prior executions by storing completed tasks as experiences (Li et al., 2024) or generalized guidance and retrieving relevant entries for new tasks (Zhao et al., 2024; Fu et al., 2024a; Ouyang et al., 2026). Some methods query memory using only the task instruction (Sun et al., 2023; Fang et al., 2026), while others additionally incorporate the current observation (Sarch et al., 2023; 2024). Such queries describe the task goal and present scene and retrieve experiences from semantically similar tasks, but they leave out the actions the agent has already executed. The guidance an agent needs differs accordingly—continuing a successful trajectory calls for different experiences than recovering from a failed grasp or a wrongly

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.

selected object—yet a query without this temporal context cannot distinguish between the two. (See Figure 3)

Beyond how memories are retrieved, what they represent also shapes the guidance an agent receives. Some methods derive memories solely from textual interaction histories (Shinn et al., 2023; Ouyang et al., 2026), whereas others incorporate visual observations to capture the states the agent actually encountered (Sarch et al., 2024; Li et al., 2024). This distinction is particularly important in embodied settings: successful trajectories can provide reusable strategies, whereas failed trajectories require an account of what went wrong. Success-based memory methods preserve strategies that led to task completion but provide limited guidance on what to avoid (Sarch et al., 2023; Wang et al., 2024a), whereas failure-based approaches often depend on matched successful executions to identify what went wrong (Sun et al., 2023; Zhang et al., 2024; Fu et al., 2024b). Recent methods draw on both outcomes but consolidate them into shared generalized guidance (Zhao et al., 2024; Fu et al., 2024a), rather than preserving success- and failure-derived knowledge as distinct signals. The merged guidance therefore carries content unrelated to the agent's current situation, so relevant knowledge arrives together with guidance the agent does not need. Together, these limitations suggest that an embodied experience memory should retrieve guidance according to the agent's current execution situation and preserve successful and failed experiences as distinct sources of knowledge.

To address these limitations, we propose **Trajectory-aware Experience Retrieval for embodied Agents (TERA)**, a training-free framework. TERA summarizes each completed execution from its trajectory images, agent reasoning, and executed actions, and for failed executions additionally infers the failure cause from the same evidence, without requiring a matched successful execution. It then distills each summary into situation-conditioned guidance and stores them in separate success and failure memory banks. At inference time, TERA summarizes the trajectory accumulated so far, retrieves relevant guidance from both banks, and presents them as distinct guidance to the planner, allowing it to contrast effective strategies with failure-avoidance guidance. As illustrated in Figure 1, this guidance help the agent continue planning until all required objects are collected, whereas the baseline terminates prematurely. We evaluate TERA on 16 VLM backbones across EB-Manipulation and VLABench, where it improves most models, with the most consistent gains on proprietary and larger open-weight backbones. Our main contributions are summarized as follows:

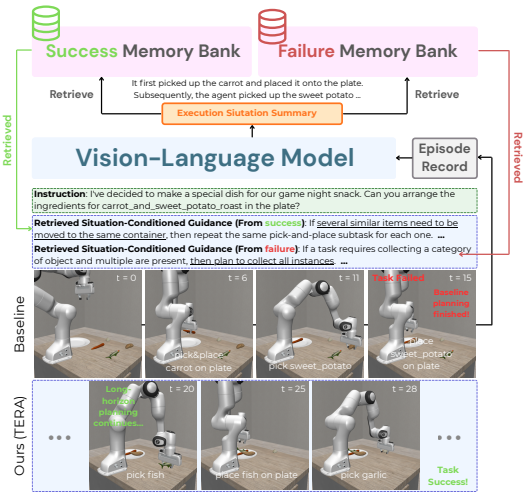
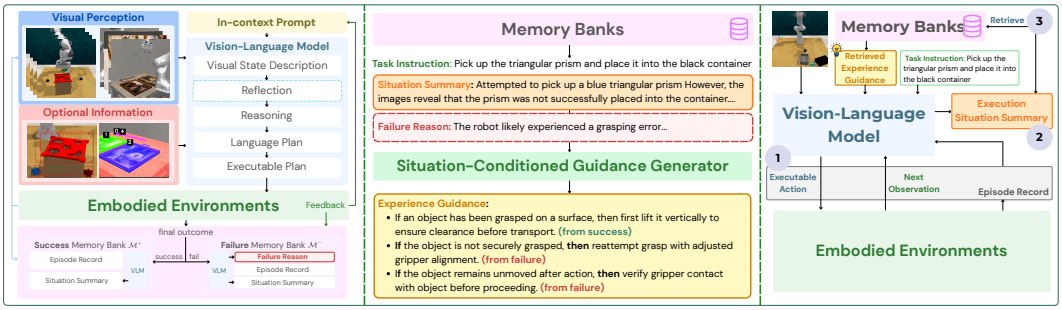


Figure 1: Qualitative comparison on VLABench. The baseline terminates prematurely, while our method summarizes the failed trajectory, retrieves relevant guidance from both success and failure memories, and generates a longer-horizon plan that successfully completes the task.

- We introduce a trajectory-aware experience retrieval framework for embodied agents that retrieves relevant situation-conditioned guidance using summaries of the agent's current multimodal execution trajectory, extending retrieval from instruction- or observation-level queries to trajectory-level context. Our analysis shows that trajectory-aware retrieval enables more situation-aware retrieval and outperforms both instruction-only queries and queries based on the instruction and current observation.
- We preserve successful and failed experiences in separate memory banks and retrieve from both, allowing the planner to contrast successful execution strategies with failure patterns and corrective guidance. This design supports retrieval from trajectories collected across different executions and VLMs, without requiring matched success–failure pairs from the same task, episode, scene, or agent execution.
- Extensive experiments across 16 VLM backbones on EB-Manipulation and VLABench demonstrate the effectiveness of TERA, with improvements of up to 7.9% in mean success rate on EB-Manipulation and 13.2%, 12.9%, and 9.4% in *Skill Match*, *Entity Match*, and *Exact Match* on VLABench, respectively.



(a) Experience Collection & Summarization (b) Situation-Conditioned Guidance Generation (c) Execution

Figure 2: Overview of TERA. **(a)** A VLM agent executes a task, and each completed episode record is assigned to the success bank $\mathcal{M}^+$ or the failure bank $\mathcal{M}^-$ by its outcome and summarized by a VLM, with a failure reason additionally inferred for failed episodes. **(b)** Situation-conditioned guidance generator converts each summary into generalized *if–then* guidance pairing an execution condition with a recommended response. **(c)** During execution, the agent acts (1), appends the resulting observation to the episode record (2), and retrieves guidance from both banks (3) using an execution situation summary combined with the task instruction; the two rule sets are supplied to the planner. Dashed boxes denote benchmark-dependent components.

2 Related Work

Memory-Augmented Embodied Agents. Memory-augmented embodied agents retain information beyond the current observation to support unseen tasks and long-horizon decision-making. Existing methods differ in whether memory is confined to the current trajectory (Ji et al., 2026; Shah et al., 2026; Sun et al., 2026) or persists across completed executions (Li et al., 2024; Sridhar et al., 2025; Jang et al., 2026). For persistent memory, retrieval is typically conditioned on the task instruction and the current observation. OptimusVLA (Li et al., 2026b), for example, retrieves task-level action trajectories from the instruction and observation, while using executed action history separately to estimate task progress. Such queries capture the task goal and current scene, but not what the agent has already attempted. TERA instead retrieves experience using a summary of the agent's current multimodal execution trajectory, extending retrieval from instruction- or observation-level queries to trajectory-level context that better reflects the current execution situation.

Learning from Successful and Failed Experiences. Experience-based agents supply past executions to the underlying model as prompting context, and they differ primarily in which execution outcomes they draw on. Success-driven methods use expert demonstrations (Jiang et al., 2023; Di Palo & Johns, 2024), retrieved examples (Sarch et al., 2023; Sridhar et al., 2025; Jang et al., 2026), or imperfect executions refined into reusable successful examples (Sarch et al., 2024). Failure-driven methods instead analyze and correct errors in the current task (Liu et al., 2023; Duan et al., 2025), or convert feedback from failed attempts into verbal reflections (Shinn et al., 2023). Recent methods learn from both outcomes: ExpeL (Zhao et al., 2024) compares a failed trajectory with a successful one for the same task and abstracts recurring patterns into a list of natural-language insights, MSI-Agent (Fu et al., 2024a) organizes experiences into successful sets and success–failure pairs to generate insights at task-specific and general scales, and ReasoningBank (Ouyang et al., 2026) distills self-evaluated successful and failed experiences into memory items that are consolidated into a single store, obtaining contrast only by repeatedly rolling out the same query. In all three, the distinction between successful and failed executions is resolved when the memory is built, and what the agent receives at inference is a single undifferentiated body of textual guidance. AutoGuide (Fu et al., 2024b) derives concise context-aware guidelines by contrasting successful and failed trajectories for the same task. At inference, it infers the current context from a partial trajectory, matches it to a stored context, and retrieves the associated guidelines. Similar to ExpeL, MSI-Agent, and ReasoningBank, however, the distinction between successful and failed executions is resolved during memory construction, and the agent retrieves from a single unified pool of textual guidance. TERA differs in both how experiences are organized and how they are retrieved. It preserves successful and failed experiences in separate memory banks and retrieves from both, allowing the planner to contrast successful execution strategies with failure patterns and corrective guidance at inference time. Because the two banks are constructed without requiring matched success–failure pairs, they can incorporate trajectories collected across different tasks, scenes, executions, and VLMs.

3 METHOD

We propose Trajectory-aware Experience Retrieval for embodied Agents (TERA), a training-free framework that retrieves past embodied execution experiences to guide an agent on unseen tasks. As illustrated in Figure 2, TERA proceeds in three stages: successful and failed trajectories are summarized offline (Sec. 3.1), each situation summary is converted into situation-conditioned guidance (Sec. 3.2), and TERA retrieves guidance during execution based on the agent's trajectory and inserts it into the planning prompt (Sec. 3.3).

3.1 EXECUTION EXPERIENCE COLLECTION AND SUMMARIZATION

A VLM agent π_{actor} interacts with the environment and collects both successful and failed executions. Each record m_i consists of the language instruction L_i, the visual observations v_i available at each planning point, the generated visual state descriptions I_i, reasoning r_i, and plans p_i, the executed actions a_i, the environment feedback d_i, and the final outcome o_i, determined by the environment's evaluation score.

Embodied environments expose execution records of different shapes, and TERA distinguishes two cases. When the environment supports replanning, execution proceeds in cycles: the agent observes an image, generates a description, reasoning, and plan, executes the corresponding action chunk, and receives action-success feedback before the next cycle begins. The observation opening cycle $c+1$ therefore serves two roles, recording the state that resulted from the chunk executed in cycle c and providing the visual input for the next plan, so aligning the record temporally lets the summarizer associate each plan and action with the visual state before and after its execution. When the environment instead elicits a complete plan in a single pass, the record comprises the instruction, the observations available at planning time, the agent's reasoning and predicted action sequence, and whatever outcome diagnostics the environment reports. TERA converts either form into a common multimodal format before summarization.

Formally, TERA processes each record offline with a VLM $\mathcal{V}_{\text{summarizer}}$ to generate a situation summary s_i conditioned on the language instruction, the multimodal trajectory across all planning points, and the final outcome:

$$s_i \sim \mathcal{V}_{\text{summarizer}}\left(\cdot \;\middle|\; L_i, \left\{\left(v_{i,t}, I_{i,t}, r_{i,t}, p_{i,t}, a_{i,t}, \tilde{d}_{i,t}\right)\right\}_{t=1}^{N_i}, o_i\right) \tag{1}$$

where N_i is the number of planning points in execution i, equal to 1 in the single-pass setting, and $\tilde{d}_{i,t}$ is the environment feedback at replanning step t, included only when the benchmark provides it. For successful and failed executions, we use separate summarization prompts, P_{success} and P_{fail}, respectively. For successful executions, the summary highlights decisions and state transitions that contributed to task completion. For failed executions, the summarizer additionally identifies where the execution went wrong and produces a failure cause f_i alongside s_i in the same pass. This separation allows successful experiences to emphasize effective execution patterns, while failed experiences retain diagnostic information about failure-prone situations and their causes. The complete success- and failure-specific summarization prompts are provided in Appendix A.1.

3.2 SITUATION-CONDITIONED GUIDANCE GENERATION

Situation summaries describe what occurred during an execution but do not specify how the agent should behave in a new one. TERA therefore converts each summary into situation-conditioned guidance that connects an execution condition to a recommended response, using an LLM with separate prompts for successful and failed experiences and supplying f_i in the latter case. Each experience yields a set Q_i of two to four guidance rules in *if–then* form, where the *if* clause states a relevant execution condition and the *then* clause a concrete planning or action guideline, as in "If the target object remains unmoved after a grasp attempt, then verify the gripper alignment and reattempt the grasp before proceeding to placement." The guidance rules capture patterns such as an effective action order, an object-verification strategy, or a subgoal progression, and object-specific names are generalized during generation to improve transfer across tasks and episodes.The complete success and failure specific guidance-generation prompts are provided in Appendix A.2 A memory item is represented as

$$M_i = (L_i, T_i, o_i, s_i, f_i, Q_i, e_i), \tag{2}$$

where T_i is the task type and f_i is empty for successful executions. The retrieval embedding is

$$e_i = \phi([L_i; s_i]), \tag{3}$$

with $\phi(\cdot)$ an embedding function applied to the concatenated instruction and situation summary. Items are stored in a success bank $\mathcal{M}^+$ and a failure bank $\mathcal{M}^-$ according to o_i.

3.3 Trajectory-aware Experience Retrieval

At each planning point j, TERA summarizes the trajectory accumulated so far, comprising the descriptions and plans generated in previous cycles, the attempted actions, and the corresponding visual observations, into an execution situation summary s_j^{exec}, and embeds it with the same function as $e_j^{\text{exec}} = \phi([L_j; s_j^{\text{exec}}])$. Unlike instruction-based retrieval, which ranks memories by similarity to the initial task instruction alone, this query reflects the agent's actual execution context. We score each stored memory by cosine similarity,

$$\text{Sim}(e_j^{\text{exec}}, e_i) = \frac{e_j^{\text{exec}} \cdot e_i}{\|e_j^{\text{exec}}\| \, \|e_i\|}, \tag{4}$$

and retrieve from the two banks independently:

$$\mathcal{R}_j^{\pm} = \text{Top}\, k_{M_i \in \mathcal{M}^{\pm}} \text{Sim}(e_j^{\text{exec}}, e_i). \tag{5}$$

Memories originating from the target episode are excluded from the candidate set to prevent episode-level leakage. The guidance Q_i of the retrieved memories are then inserted into the planning prompt as two labeled groups, one from the success memory bank $\mathcal{M}^+$ and one from the failure memory bank $\mathcal{M}^-$, so that the planner receives what has previously worked alongside what has previously failed and resolves the contrast when the plan is formed rather than when the memory is built.

The point at which retrieval can occur depends on the environment. Where replanning is supported, the agent may generate several plans within an episode, so the summary and the retrieved set are updated at every replanning stage; as the partial trajectory evolves, TERA may retrieve different experiences even though the instruction is unchanged. Where the agent instead produces a complete action sequence in a single planning process, no intermediate point is available for intervention, and TERA retrieves at the beginning of each attempt using a truncated record of the preceding one, so that memory selection still reflects prior execution behavior rather than instruction similarity alone.

4 Experiments

4.1 Experimental Setups

Benchmarks. We evaluate TERA on EB-Manipulation from EmbodiedBench (Yang et al., 2025) for low-level manipulation and VLABench (Zhang et al., 2025) for task planning. In EB-Manipulation, VLM agents control a robotic arm using a 7-dimensional action vector consisting of position, orientation, and gripper commands. EmbodiedBench facilitates direct low-level manipulation by discretizing the position and orientation components of the action space (Yin et al., 2025), supporting replanning during execution, and providing auxiliary detection and pose information for indexed objects. Within EB-Manipulation, we consider all five capability categories: *Base*, *Common Sense*, *Complex Instruction*, *Spatial Awareness*, and *Visual Appearance*. These categories encompass diverse challenges in action sequencing, commonsense reasoning, intent understanding, spatial grounding, and visual grounding.

For VLABench, we select five tasks from the *Complex* task set: *book rearrange*, *cook dishes*, *hammer nail and hang picture*, *chemistry experiments*, and *Texas hold'em*. The selected *Complex* setting is non-interactable and does not provide intermediate trajectories for online replanning, but its two-to-ten-action plans allow us to examine whether experience-based memory meaningfully changes

subsequent planning. We adapt the evaluation protocol by truncating the trajectory from the preceding evaluation attempt on the same task and using it only to construct the situation-summary query for experience retrieval. The previous trajectory itself is not provided to the planner as context. Memories with the same instruction are excluded from retrieval, and no retrieved memory shares an identical scene setting with the evaluation example. Thus, the planner receives only situation-conditioned guidance retrieved from distinct prior experiences, preserving TERA's core trajectory-aware retrieval mechanism without directly reusing the preceding attempt. We provide the detailed protocol in Appendix B.

Memory Bank Construction. For each benchmark, we construct separate success and failure memory banks from trajectories collected using multiple VLM backbones. To collect a diverse set of successful and failed experiences, we construct the memory banks from trajectories generated by six VLM agents in total. We select models to provide a sufficient mixture of successful and failed executions on each benchmark while covering both proprietary and open-weight VLMs. For EB-Manipulation, we use GPT-4o, InternVL3-8B, and InternVL3-78B-AWQ, yielding 684 experiences (150 successful and 534 failed). For VLABench, we use Claude Sonnet-4.6, Gemini-2.5-Flash, and Qwen2.5-VL-7B, yielding 1,427 experiences (212 successful and 1,215 failed). Appendix A.3 provides the detailed dataset composition. During retrieval, memories originating from the target episode are excluded from the candidate set to prevent reuse of the target episode. TERA retrieves two experiences per bank by default.

Baselines and Compared Methods. *Baseline* denotes the reproduced VLM agent without retrieved experience. Since ExpeL (Zhao et al., 2024), ReasoningBank (Ouyang et al., 2026), and Reflexion (Shinn et al., 2023) were originally designed for text-based agents, we adapt them to the same embodied agent by replacing their text-environment interfaces with our observation and action interfaces while preserving their core memory or reflection procedures. Variants that use only successful or only failed experiences retrieve two entries from the corresponding bank, while ExpeL and ReasoningBank follow their original retrieval settings. Reflexion is evaluated separately with one additional attempt on each task failed by the corresponding baseline. We provide implementation details for each method in Appendix C.

Models. We benchmark 16 models, including eight proprietary models and eight open-weight models. The proprietary models include Gemini-2.5-Flash and Gemini-2.5-Pro (Comanici et al., 2025), Gemini-3.1-Pro, Gemini-3.5-Flash, Claude Sonnet-4.6 (Anthropic, 2026), GPT-4o, GPT-4.1, and GPT-5.5 (OpenAI, 2024b; 2025; 2026). The open-weight models include InternVL3 (8B, 38B, and 78B-4bit) (Zhu et al., 2025), InternVL2-8B (OpenGVLab Team, 2024), Ovis2-16B (Lu et al., 2024), Qwen2-VL-7B (Wang et al., 2024b), Qwen2.5-VL-7B (Bai et al., 2025b), and Qwen3-VL-8B (Bai et al., 2025a). We follow the default evaluation settings provided by each benchmark.

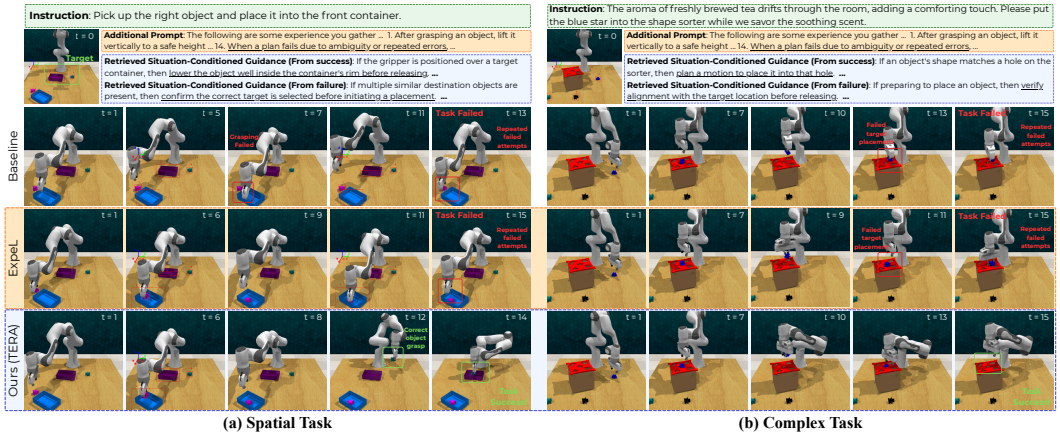
Metrics. We report task success rate for EB-Manipulation, defined as the percentage of episodes successfully completed. For VLABench, we report *Skill Match*, *Entity Match*, and *Exact Match*, which measure agreement with the reference skills, entities, and complete skill–entity sequence, respectively. We set the temperature to 0 for deterministic decoding, reporting results from a single evaluation run.

4.2 Main Results

Results on EB-Manipulation. Table 1 shows that TERA (*Ours*) achieves absolute gains of up to 7.9% in mean success rate on EB-Manipulation. It consistently improves proprietary and large open-weight models with at least 38B parameters, including gains of 6.6% and 7.1% for InternVL3-78B-4bit and InternVL3-38B, respectively. The largest category-level gain is observed on Gemini-2.5-Flash for *Base*, increasing from 35.4% to 52.1% (+16.7%), with the strongest overall improvements on *Complex* and *Base*. As illustrated in Figure 3, retrieved success- and failure-derived guidance helps correct both target selection and manipulation execution. Gains are less consistent for smaller open-weight models: Qwen3-VL-8B improves by 2.2%, whereas InternVL3-8B and Ovis2-16B decrease by 9.2% and 3.6%. This suggests that additional retrieved context may be harder for smaller models to integrate with EmbodiedBench's existing few-shot prompt. Qwen2.5-VL-7B and Qwen2-VL-7B also frequently fail to produce valid structured actions, as reported in EmbodiedBench, while InternVL2-8B exhibits the same failure in our evaluation; all three therefore achieve zero success under both Baseline and TERA. Representative failures are provided in Appendix D.

Table 1: Task success rate (%) on EB-Manipulation. *Baseline* denotes our reproduced base agent without memory. The better result within each backbone and task is shown in **bold**.

Model	EB-Manipulation											
	Base		Common		Complex		Visual		Spatial		Mean	
	Baseline	*Ours*	*Baseline*	*Ours*	*Baseline*	*Ours*	*Baseline*	*Ours*	*Baseline*	*Ours*	*Baseline*	*Ours*
Proprietary models												
Gemini-3.1-Pro	47.9	**54.2** (+6.3)	43.8	**50.0** (+6.2)	41.7	**56.3** (+14.6)	38.9	**50.0** (+11.1)	31.3	**33.3** (+2.0)	40.8	**48.7** (+7.9)
Gemini-3.5-Flash	54.2	**58.3** (+4.1)	**54.2**	50.0 (-4.2)	52.1	**54.2** (+2.1)	**52.8**	50.0 (-2.8)	31.3	**37.5** (+6.2)	48.7	**50.0** (+1.3)
Gemini-2.5-Pro	33.3	**43.8** (+10.5)	**37.5**	33.3 (-4.2)	41.7	41.7 (-)	47.2	**50.0** (+2.8)	**29.2**	18.8 (-10.4)	**37.3**	36.8 (-0.5)
Gemini-2.5-Flash	35.4	**52.1** (+16.7)	27.1	**29.2** (+2.1)	41.7	**50.0** (+8.3)	41.7	**44.4** (+2.7)	27.1	**35.4** (+8.3)	34.2	**42.1** (+7.9)
Claude Sonnet-4.6	37.5	37.5 (-)	27.1	**31.3** (+4.2)	27.1	**41.7** (+14.6)	38.9	**41.7** (+2.8)	35.4	**39.6** (+4.2)	32.9	**38.2** (+5.3)
GPT-5.5	47.9	**50.0** (+2.1)	41.7	41.7 (-)	41.7	**50.0** (+8.3)	36.1	**44.4** (+8.3)	25.0	**37.5** (+12.5)	38.6	**44.7** (+6.1)
GPT-4.1	31.3	**41.7** (+10.4)	25.0	**27.1** (+2.1)	35.4	**41.7** (+6.3)	36.1	**41.7** (+5.6)	31.3	**39.6** (+8.3)	31.6	**38.2** (+6.6)
GPT-4o	31.3	**33.3** (+2.0)	25.0	25.0 (-)	22.9	**29.2** (+6.3)	27.8	**30.6** (+2.8)	31.3	**37.5** (+6.2)	27.6	**31.1** (+3.5)
Open-weight models												
InternVL3-78B-4bit	16.7	**25.0** (+8.3)	20.8	**22.9** (+2.1)	18.8	**25.0** (+6.2)	16.7	**25.0** (+8.3)	29.2	**37.5** (+8.3)	20.6	**27.2** (+6.6)
InternVL3-38B	16.7	**20.8** (+4.1)	18.8	**29.2** (+10.4)	16.7	**22.9** (+6.2)	19.4	**30.6** (+11.2)	27.1	**31.3** (+4.2)	19.7	**26.8** (+7.1)
InternVL3-8B	**14.6**	10.4 (-4.2)	**18.8**	8.3 (-10.5)	**16.7**	8.3 (-8.4)	**13.9**	8.3 (-5.6)	**22.9**	6.3 (-16.6)	**17.5**	8.3 (-9.2)
Ovis2-16B	10.4	**16.7** (+6.3)	**14.6**	6.3 (-8.3)	**16.7**	8.3 (-8.4)	**19.4**	16.7 (-2.7)	**16.7**	12.5 (-4.2)	**15.4**	11.8 (-3.6)
Qwen3-VL-8B	4.2	**12.5** (+8.3)	12.5	12.5 (-)	**10.4**	6.3 (-4.1)	8.3	**13.9** (+5.6)	22.9	**25.0** (+2.1)	11.8	**14.0** (+2.2)



(a) Spatial Task (b) Complex Task

Figure 3: Qualitative comparison of Baseline, ExpeL, and TERA (*Ours*) on EB-Manipulation. All examples are conditioned on the task instruction, and colored dashed boxes indicate additional information used during execution. (a) Baseline fails to grasp the target, and ExpeL repeats the same unsuccessful replan, whereas TERA verifies and grasps the correct object and places it in the intended container. (b) TERA matches the star-shaped object to the corresponding slot, refines its placement pose, verifies alignment, and inserts it successfully, whereas the baselines fail.

Results on VLABench. Table 2 shows that TERA (*Ours*) achieves absolute gains of up to 13.2%, 12.9%, and 9.4% in *Skill Match*, *Entity Match*, and *Exact Match* on the VLABench *Complex* task, respectively. TERA improves 15 of the 16 evaluated backbones, with average gains of 5.3%, 5.3%, and 3.0% across the three metrics. As illustrated in Figure 1, TERA continues planning and repeats the pick-and-place procedure until all required objects are collected, whereas the baseline terminates prematurely in the bottom example. These behaviors improve entity grounding and the completion of multi-step action sequences, leading to gains across all three metrics.

4.3 Analyses

Unless otherwise specified, we conduct our analyses on Gemini-2.5-Flash and InternVL3-38B, one proprietary and one open-weight backbone with mid-range performance among the evaluated VLMs, to assess the contribution of individual components in our framework across different model families.

Comparison of Experience Designs. Table 3 compares experience designs under a fixed backbone and memory bank. Beyond ExpeL and ReasoningBank, we evaluate three settings that isolate a design choice shared by a family of prior methods rather than reproducing any single system: success-only memory (Jiang et al., 2023; Sarch et al., 2024), failure-only mem-

Table 2: Average VLABench task-planning performance (%). *Baseline*: reproduced agent without memory. Best results per backbone and metric in **bold**.

Model	VLABench					
	Skill Match		Entity Match		Exact Match	
	Baseline	*Ours*	*Baseline*	*Ours*	*Baseline*	*Ours*
Proprietary models						
Gemini-3.1-Pro	73.5	**76.2** (+2.7)	75.2	**78.1** (+2.9)	42.4	**46.9** (+4.5)
Gemini-3.5-Flash	67.2	**72.5** (+5.3)	70.2	**74.9** (+4.7)	36.8	**39.9** (+3.1)
Gemini-2.5-Pro	64.0	**71.1** (+7.1)	63.1	**69.4** (+6.3)	32.0	**35.8** (+3.8)
Gemini-2.5-Flash	57.4	**69.0** (+11.6)	54.6	**66.1** (+11.5)	26.7	**32.8** (+6.1)
Claude-Sonnet-4.6	64.3	**72.1** (+7.8)	64.4	**72.6** (+8.2)	29.2	**35.5** (+6.3)
GPT-5.5	61.6	**74.8** (+13.2)	62.7	**75.6** (+12.9)	33.5	**42.9** (+9.4)
GPT-4.1	56.2	**63.9** (+7.7)	60.1	**66.1** (+6.0)	22.8	**25.9** (+3.1)
GPT-4o	59.7	**64.9** (+5.2)	55.5	**60.7** (+5.2)	21.0	**24.1** (+3.1)
Open-weight models						
InternVL3-78B-4bit	57.1	**58.1** (+1.0)	51.4	**51.8** (+0.4)	**17.2**	16.8 (-0.4)
InternVL3-38B	60.4	**64.1** (+3.7)	59.4	**62.9** (+3.5)	23.0	**25.2** (+2.2)
InternVL3-8B	52.6	**53.8** (+1.2)	41.5	**43.1** (+1.6)	9.9	**11.4** (+1.5)
InternVL2-8B	48.8	**53.7** (+4.9)	38.2	**41.6** (+3.4)	11.4	**12.9** (+1.5)
Ovis2-16B	58.5	**62.2** (+3.7)	41.4	**43.1** (+1.7)	8.9	**9.3** (+0.4)
Qwen3-VL-8B	61.8	61.8 (-)	55.3	**57.7** (+2.4)	20.5	**21.4** (+0.9)
Qwen2.5-VL-7B	**58.2**	54.8 (-3.4)	**42.7**	38.4 (-4.3)	**13.0**	8.2 (-4.8)
Qwen2-VL-7B	44.5	**57.6** (+13.1)	31.6	**49.2** (+17.6)	5.7	**13.4** (+7.7)

ory (Liu et al., 2023), and descriptive situation summaries (Sarch et al., 2023; Fu et al., 2024a) in place of generalized guidance. Relative to the no-memory baselines of 34.2% for Gemini-2.5-Flash and 19.7% for InternVL3-38B, success-only and failure-only memory achieve 35.5% and 29.8%, respectively, on Gemini-2.5-Flash, but 20.2% and 22.4% on InternVL3-38B. Thus, success-only memory outperforms failure-only memory on Gemini-2.5-Flash, whereas the opposite holds on InternVL3-38B. Consolidating successful and failed experiences into a single memory yields method-dependent effects: ExpeL improves both backbones, to 37.7% and 21.1%, whereas ReasoningBank improves InternVL3-38B to 23.2% while reducing Gemini-2.5-Flash to 33.3%. Representation contributes independently. Replacing the guidance with the situation summaries they are derived from gives 38.6% and 20.2%, the latter close to its no-memory baseline, indicating that most of the gain comes from the rule form rather than the retrieved content: a summary records what occurred, whereas a rule states the triggering condition and the recommended response. No prior design is uniformly dominated, as ExpeL and ReasoningBank each lead on individual categories, but TERA attains the highest mean on both backbones, by 3.5 and 3.6 points over the strongest alternative.

Table 3: Comparison of experience designs on EB-Manipulation using Gemini-2.5-Flash and InternVL3-38B, measured by task success rate (%). "+" denotes the baseline augmented with the corresponding method. Best results are shown in **bold** and second-best results are underlined.

Method	EB-Manipulation					
	Base	Common	Complex	Visual	Spatial	Mean
Gemini-2.5-Flash	35.4	27.1	41.7	41.7	27.1	34.2
+ *Success-only*	33.3	27.1	41.7	**44.4**	33.3	35.5
+ *Failure-only*	33.3	27.1	29.2	30.6	29.2	29.8
+ *ExpeL*	39.6	**37.5**	43.8	**44.4**	25.0	37.7
+ *ReasoningBank*	35.4	35.4	39.6	38.9	18.8	33.3
+ *Situation Summary*	39.6	**37.5**	41.7	38.9	**35.4**	38.6
+ *Ours*	**52.1**	29.2	**50.0**	**44.4**	**35.4**	**42.1**
InternVL3-38B	16.7	18.8	16.7	19.4	27.1	19.7
+ *Success-only*	14.6	18.8	14.6	22.2	**31.3**	20.2
+ *Failure-only*	18.8	16.7	**22.9**	25.0	29.2	22.4
+ *ExpeL*	16.7	8.3	18.8	**33.3**	**31.3**	21.1
+ *ReasoningBank*	**20.8**	25.0	**22.9**	16.7	29.2	23.2
+ *Situation Summary*	16.7	22.9	14.6	25.0	22.9	20.2
+ *Ours*	**20.8**	**29.2**	**22.9**	30.6	**31.3**	**26.8**

Comparison of Retrieval Strategies. All settings in Figure 4 access the same memory bank and retrieve two successful and two failed experiences, differing only in the retrieval query. Random and instruction-only retrieval leave Gemini-2.5-Flash at or below its no-memory baseline, showing that task-level similarity alone does not distinguish different execution situations. Adding the current observation, a query design similar to that of OptimusVLA (Li et al., 2026b), improves Gemini-2.5-Flash but yields performance on InternVL3-38B comparable to random and instruction-only retrieval, suggesting that a single frame may not identify memories relevant to the current execution state. Random retrieval nevertheless remains relatively competitive, likely because the few-shot examples provided by EmbodiedBench and the models' pretrained knowledge already supply sub-

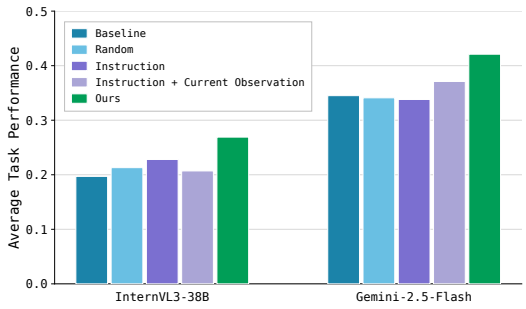


Figure 4: Comparison of retrieval strategies on EB-Manipulation. The baseline performs no memory retrieval.

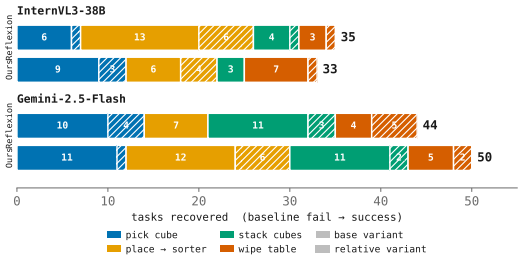


Figure 5: Comparison of task recovery between Reflexion and TERA on EB-Manipulation for Gemini-2.5-Flash and InternVL3-38B.

stantial task guidance. In contrast, TERA summarizes the observations and actions accumulated up to the current step and is the only strategy that improves both backbones, reaching 42.1% and 26.8%.

Analysis of Base-Agent Failures We further investigate the tasks that are failed by the base agent but successfully solved by TERA. To contextualize these gains, we compare TERA with Reflexion (Shinn et al., 2023), which reflects on the agent's initial failed attempt and performs a single retry for each failed task. Figure 5 reports, by task type, the number of base-agent failures on which each method succeeds. On InternVL3-38B, TERA solves 33 such tasks, comparable to Reflexion's 35. On Gemini-2.5-Flash, TERA solves 50 tasks, exceeding Reflexion's 44 while covering all eight task types rather than seven. TERA solves previously failed tasks across diverse categories without retrying the same instance, suggesting that its situation-conditioned success and failure guidance transfers across tasks by combining effective strategies with failure-avoidance guidance.

Effect of the Number of Retrieved Experiences. Figure 6 examines the effect of varying the number of retrieved experiences. Top-2 retrieval achieves the best performance on both InternVL3-38B and Gemini-2.5-Flash. Retrieving a single experience may not cover the diverse procedural and perceptual challenges in manipulation. We observe that moving to Top-3 slightly reduces performance, likely because the additional memory introduces more noise and contextual interference. Thus, Top-2 provides the best balance between experience and contextual complexity in our setting.

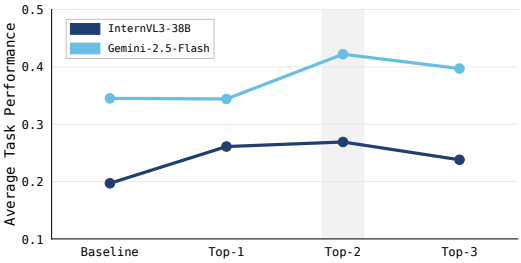


Figure 6: Effect of the number of retrieved experiences on EB-Manipulation across Gemini-2.5-Flash, InternVL3-38B.

5 Conclusion

We presented TERA, a training-free trajectory-aware experience retrieval framework for embodied agents. TERA distills multimodal trajectories into generalized situation-conditioned guidance, stores success- and failure-derived guidance in separate memory banks without matched trajectory pairs, and retrieves relevant guidance using a summary of the current execution trajectory. Across 16 backbones on EB-Manipulation and VLABench, TERA improves most proprietary models and open-weight models. Our analyses attribute these gains to conditioning retrieval on the current execution situation and contrasting success- and failure-derived guidance.

Limitation and Future Works. Despite the overall improvements, smaller open-weight models do not benefit uniformly from retrieved memory, likely because weaker embodied instruction-following and context-integration capabilities make them more sensitive to additional retrieved context. TERA also relies on fixed top-k retrieval from both success and failure banks. We believe that adaptive retrieval could better accommodate these models by selecting more useful experiences and adjusting the amount and balance of retrieved guidance according to the model and current execution state.

AI USE STATEMENT

We used generative AI tools to assist with manuscript editing, including language refinement, restructuring of selected passages, and feedback on the presentation of methodological details. All AI-assisted content was reviewed and revised by the authors, and all reported experiments, results, and claims were verified by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide implementation details of TERA in the main paper and Appendix, including the prompts for situation summarization and guidance generation, memory-bank construction and statistics, retrieval settings, benchmark-specific evaluation protocols, and failure analyses. These details are intended to support reproduction of the reported experiments.

REFERENCES

Anthropic. Introducing claude sonnet 4.6. https://www.anthropic.com/news/claude-sonnet-4-6, February 2026.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

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*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025b. URL https://arxiv.org/abs/2502.13923.

Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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 2025. *arXiv preprint arXiv:2507.06261*, 10, 2025.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28091–28114. Curran Associates, Inc., 2023. doi: 10.52202/075280-1220.

Norman Di Palo and Edward Johns. Keypoint action tokens enable in-context imitation learning in robotics. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.096.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8469–8488. PMLR, 2023.

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. AHA: A vision-language-model for detecting and reasoning over failures in robotic manipulation. In *International Conference on Learning Representations*, 2025.

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 *Advances in Neural Information Processing Systems*, volume 35, 2022.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502. Association for Computational Linguistics, 2026.

Dayuan Fu, Biqing Qi, Yihuai Gao, Che Jiang, Guanting Dong, and Bowen Zhou. MSI-agent: Incorporating multi-scale insight into embodied agents for superior planning and decision-making. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 643–659. Association for Computational Linguistics, November 2024a. doi: 10.18653/v1/2024.emnlp-main.38.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. AutoGuide: Automated generation and selection of context-aware guidelines for large language model agents. In *Advances in Neural Information Processing Systems*, volume 37, 2024b. doi: 10.52202/079017-3811. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/d8efbb5dd415974eb095c3f06bff1f48-Abstract-Conference.html.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D value maps for robotic manipulation with language models. In *Proceedings of The 7th Conference on Robot Learning*, volume 229, pp. 540–562. PMLR, 2023.

Sanghwan Jang, Minjin Jeon, Minsoo Kim, Seong Jin Choi, Dongha Kim, and Hwanjo Yu. RA-VLA: Retrieval-augmented VLA for test-time adaptation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

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. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. VIMA: Robot manipulation with multimodal prompts. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 14975–15022. PMLR, 2023.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

Xiaozhe Li, Tianyi Lyu, Yang Li, Yichuan Ma, Peiji Li, Linyang Li, Qipeng Guo, Dahua Lin, and Kai Chen. What and when to distill: Selective hindsight distillation for multi-turn agents. *arXiv preprint arXiv:2605.19447*, 2026a.

Zaijing Li, Yuquan Xie, Rui Shao, Gongwei Chen, Dongmei Jiang, and Liqiang Nie. Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-1580.

Zaijing Li, Bing Hu, Rui Shao, Gongwei Chen, Dongmei Jiang, Pengwei Xie, Jianye Hao, and Liqiang Nie. Global prior meets local consistency: Dual-memory augmented vision-language-action model for efficient robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35135–35145, June 2026b.

Zeyi Liu, Arpit Bahety, and Shuran Song. REFLECT: Summarizing robot experiences for failure explanation and correction. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3468–3484. PMLR, 2023.

Shiyin Lu, Yang Li, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, and Han-Jia Ye. Ovis: Structural embedding alignment for multimodal large language model. *arXiv preprint arXiv:2405.20797*, 2024.

OpenAI. New embedding models and api updates. https://openai.com/index/new-embedding-models-and-api-updates/, 2024a.

OpenAI. Hello GPT-4o. https://openai.com/index/hello-gpt-4o/, May 2024b.

OpenAI. Introducing GPT-4.1 in the API. https://openai.com/index/gpt-4-1/, April 2025.

OpenAI. Introducing GPT-5.5. https://openai.com/index/introducing-gpt-5-5/, April 2026.

OpenGVLab Team. InternVL2: Better than the best—expanding performance boundaries of open-source multimodal models with the progressive scaling strategy. https://internvl.github.io/blog/2024-07-02-InternVL-2.0/, July 2024.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, 2026.

Gabriel Sarch, Yue Wu, Michael Tarr, and Katerina Fragkiadaki. Open-ended instructable embodied agents with memory-augmented large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3468–3500. Association for Computational Linguistics, December 2023. doi: 10.18653/v1/2023.findings-emnlp.226.

Gabriel Sarch, Lawrence Jang, Michael J. Tarr, William W. Cohen, Kenneth Marino, and Katerina Fragkiadaki. VLM agents generate their own memories: Distilling experience into embodied programs of thought. In *Advances in Neural Information Processing Systems*, volume 37, pp. 75942–75985, 2024. doi: 10.52202/079017-2418.

Rutav Shah, Yisu Li, Femi Bello, Yuke Zhu, and Roberto Mart'in-Mart'in. Memory retrieval in visuomotor policies for long-horizon robot control. In *Robotics: Science and Systems*, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10740–10749, 2020.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021.

Kaustubh Sridhar, Souradeep Dutta, Dinesh Jayaraman, and Insup Lee. REGENT: A retrieval-augmented generalist agent that can act in-context in new environments. In *International Conference on Learning Representations*, 2025.

Hao Sun, Yu Song, Teng Shiyu, Ziwei Niu, and Yen-wei Chen. MIRTH: Mutual-information reasoning with temporal hubs for vision-language-action agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22199–22215. Association for Computational Linguistics, July 2026. doi: 10.18653/v1/2026.acl-long.1016.

Wangtao Sun, Xuanqing Yu, Shizhu He, Jun Zhao, and Kang Liu. ExpNote: Black-box large language models are better task solvers with experience notebook. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 15470–15481, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.1034.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: A large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 809–819. Association for Computational Linguistics, 2018. doi: 10.18653/v1/N18-1074.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024b.

Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, Xiaojian Ma, and Yitao Liang. JARVIS-1: Open-world multi-task agents with memory-augmented multimodal language models. *arXiv preprint arXiv:2311.05997*, 2023.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. EmbodiedBench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757, 2022.

Zewei Ye, Weifeng Lu, Minghao Ye, Tao Lin, Shuo Yang, Junchi Yan, and Bo Zhao. RoboFAC: A comprehensive framework for robotic failure analysis and correction. In *Proceedings of the World Artificial Intelligence Conference Academic (WAICA)*, 2026. Accepted to WAICA 2026 Main Conference.

Yida Yin, Zekai Wang, Yuvan Sharma, Dantong Niu, Trevor Darrell, and Roei Herzig. In-context learning enables robot action prediction in llms. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8972–8979. IEEE, 2025.

Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. VLABench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11142–11152, October 2025.

Tianjun Zhang, Aman Madaan, Luyu Gao, Steven Zheng, Swaroop Mishra, Yiming Yang, Niket Tandon, and Uri Alon. In-context principle learning from mistakes. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 59520–59558. PMLR, 2024.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936.

Jiaming Zhou, Ke Ye, Jiayi Liu, Teli Ma, Zifan Wang, Ronghe Qiu, Kun-Yu Lin, Zhilin Zhao, and Junwei Liang. Exploring the limits of vision-language-action manipulation in cross-task generalization. In *Advances in Neural Information Processing Systems*, 2025.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

APPENDIX

We provide additional methodological details and analyses that could not be included in the main paper due to space limitations. Sec. A presents the prompt templates used to generate situation summaries and situation-conditioned guidance, together with details of the experience dataset construction and memory retrieval procedure. Sec. B describes the evaluation protocol for VLABench. Sec. C provides the implementation details of the compared methods. Finally, Sec. D presents a detailed failure analysis, including performance degradation caused by retrieved guidance, the limitations of small-parameter VLMs on EB-Manipulation, and cases in which retrieved experiences adversely affect task planning.

A PROMPTS AND RETRIEVAL IMPLEMENTATION DETAILS

As described in Secs. 3.1–3.3, we provide the complete input prompts and output formats used for situation summarization, situation-conditioned guidance generation, experience dataset construction, and memory retrieval. All experiments were conducted using one or at most two NVIDIA RTX A6000 48GB GPUs for each open-weight model. Here, situation-conditioned guidance refers to *if–then* experience rules that pair an execution condition with either a recommended actions or planning patterns to avoid. Sec. A.1 provides the benchmark-specific prompts used to generate situation summaries from successful and failed trajectories, as well as execution situation summaries during inference. Sec. A.2 presents the prompts used to convert situation summaries into situation-conditioned guidance, represented as experience rules, for the success and failure memory banks. Sec. A.3 describes the models used to collect experiences for EB-Manipulation and VLABench and reports the number of successful and failed trajectories contributed by each model. Finally, Sec. A.4 details how the execution situation summary generated from the current trajectory is compared with the situation summaries stored in the memory banks to retrieve relevant guidance.

A.1 TEMPLATE FOR SITUATION SUMMARY GENERATION

We provide the complete prompt templates and output formats used to generate situation summaries for EB-Manipulation and VLABench. As described in Sec. 3.1, TERA uses a VLM to summarize completed successful and failed trajectories during memory construction and to summarize the trajectory accumulated so far during inference. We use Gemini-2.5-Pro (Comanici et al., 2025), a natively multimodal sparse mixture-of-experts Transformer, for all situation-summary generation. It jointly processes the task instruction, temporally ordered trajectory images, and textual execution records in a single multimodal prompt to produce a structured situation summary without additional training. Because EB-Manipulation and VLABench provide execution records in different forms, we use a separate prompt for each benchmark. In EB-Manipulation, execution proceeds through multiple replanning cycles. At each cycle, the agent observes an image, generates a visual state description, reasoning, and plan, executes the corresponding action chunk, and receives action-success feedback before the next cycle begins. As illustrate in Figure 7, 8 the prompt therefore presents the trajectory in temporal order, allowing the VLM to associate each planned and executed action with the visual states observed before and after its execution.

In contrast, the selected VLABench setting is non-interactable and does not support intermediate replanning. The agent generates a complete action sequence in a single planning pass, and the execution record comprises the task instruction, the observations available at planning time, the agent's reasoning and predicted action sequence, and the outcome diagnostics reported by the environment. As illustrated in Figure 7, and 8. We therefore use a separate prompt that summarizes the planned

```
{
  "trajectory_type": "SUCCESSFUL",

  "instruction":
    "Write a concise 3-5 sentence factual summary of what was actually executed and how
     the scene changed over time.",

  "trajectory_description": [
    "One or more plan cycles. Each plan cycle contains visual_state_description,
     reasoning_and_reflection, language_plan, and executable_plan (the plan the robot
     intended).",
    "Use images as ground truth for what changed.",
    "Do not list raw 7-D coordinates; describe actions as abstract behaviors."
  ],

  "Annotated image timeline": [
    "Images are ordered from earliest to latest state.",
    "The first image is the initial state.",
    "The last image is the latest state right before current replan.",
    "Use executed_actions/action_success as ground truth; use images to verify state
     transitions."
  ],

  "images": [
    "<image>   # Image[0]",
    "<image>   # Image[1]",
    "...",
    "<image>   # Image[last]"
  ],

  "structured_trajectory": {
    "Instruction": "<instruction>",

    "Plan cycle 1": {
      "visual_state_description": "<...>",
      "reasoning_and_reflection": "<...>",
      "language_plan": "<...>",
      "executable_plan": "<...>"
    },

    "...": "...",

    "Executed actions": [
      "[x, y, z, rx, ry, rz, gripper]",
      "..."
    ]
  }
}
```

Figure 7: Template for an EB-Manipulation situation summary of a successful trajectory input prompt.

execution and its outcome without assuming the cycle-level observations and action-success feedback available in EB-Manipulation.

Memory-Bank Situation Summarization. For completed trajectories, the final success or failure outcome is provided to the summarizer. For successful trajectories, the VLM summarizes the execution situation and the decisions and actions that contributed to task completion. For failed trajectories, it additionally infers the likely failure reason from the multimodal execution record.

We provide benchmark-specific diagnostic categories, informed by the failure analyses of EmbodiedBench (Yang et al., 2025) and RoboFAC (Ye et al., 2026) and adapted to the execution and evaluation settings of each benchmark. EB-Manipulation requires low-level robotic control through 7-dimensional actions; therefore, its categories include *step omission*, *wrong target object or object state*, *position deviation*, *orientation deviation*, *grasping error*, and *timing error*. In contrast, VLABench primarily evaluates task-level planning rather than low-level control. Its prompt there-

```
{
  "trajectory_type": "FAILED",

  "Step 1":
    "Write a concise 3-5 sentence factual summary of what the robot tried, what actually
     executed, and where it went wrong.",

  "trajectory_description": [
    "One or more plan cycles. Each plan cycle contains visual_state_description,
    reasoning_and_reflection, language_plan, and executable_plan (the plan the robot
    intended).",
    "Use images as ground truth for what changed.",
    "Do not list raw 7-D coordinates; describe actions as abstract behaviors."
  ],

  "Step 2":
    "First, read the six failure-cause categories below. Then on a NEW LINE write '
    Failure reason:' followed by a 1-2 sentence root-cause analysis.",

  "failure_reason_instruction":
    "Ground the reason in image timeline + executed_actions evidence, and mention scene-
    understanding mismatch when relevant.",

  "failure_cause_categories": [
    "Step omission : a required step was skipped",
    "Wrong target object : Fail to select the correct object to manipulate as specified
    by the language instruction or incorrect object stat, unlike annotated image.",
    "Position deviation : The end-effector fails to reach the correct position",
    "Orientation deviation : The end-effector fails to reach the correct orientation",
    "Grasping error : gripper failed to close properly or the object slipped",
    "Timing error : Executing the subtask at an incorrect timing (e.g. open/close
    mistimed)"
  ],

  "images": [
    "<image> # Image[0]",
    "<image> # Image[1]",
    "...",
    "<image> # Image[last]"
  ],

  "structured_trajectory": {
    "Instruction": "<instruction>",

    "Plan cycle 1": {
      "visual_state_description": "<...>",
      "reasoning_and_reflection": "<...>",
      "language_plan": "<...>",
      "executable_plan": "<...>"
    },

    "...": "...",

    "Executed actions": [
      "[x, y, z, rx, ry, rz, gripper]",
      "..."
    ]
  }
}
```

Figure 8: Template for an EB-Manipulation situation summary of a failed trajectory input prompt.

fore provides the skill-match, entity-match, and exact-match scores as outcome diagnostics and considers *step omission*, *wrong target object or object state*, *wrong action*, and *wrong plan* as reference failure types. These categories guide diagnosis rather than restricting the output to a fixed label set. The inferred failure reason is stored with the situation summary and used to generate situation-conditioned guidance for the failure memory bank.

```
{
  "images": [
    "<image>   # input.png (four views, unlabeled)",
    "<image>   # input_mask.png (four views, numbered)"
  ],

  "trajectory_type":
    "You are summarizing a SUCCESSFUL robotic manipulation trajectory.",

  "instruction":
    "The agent was given the instruction: <instruction>",

  "Image information": [
    "The first image shows four different perspectives of objects (without labels).",
    "The second image contains the same four perspectives of objects, but each view is
     labeled with a number (representing each object's identifier)."
  ],

  "summary_instruction":
    "Summarize the agent's reasoning flow objectively based on what it actually inferred
     :",

  "Agent CoT Output":
    "<origin_output>",

  "output_instruction":
    "Write a concise 2~3 sentence factual summary of what was actually executed plan.",

  "constraints": [
    "Use the agent's CoT output as ground truth for what was planned and executed.",
    "Use the images as ground truth for the starting state of the scene.",
    "Do NOT reference object numbers or exact object name; describe actions as abstract
     behaviors."
  ]
}
```

Figure 9: Template for a VLABench situation summary of a successful trajectory input prompt.

Inference-Time Execution Situation Summarization. During inference, TERA instead summarizes the trajectory accumulated so far as an execution situation summary. Figure 11 and 12 illustrate additional instructions in generating execution situation summary. In EB-Manipulation, this trajectory contains the observations, descriptions, plans, attempted actions, and feedback available up to the current replanning stage. In VLABench, it consists of a truncated record from the preceding execution attempt. Because no final success or failure outcome is provided at this stage, the prompt does not ask the VLM to predict whether the task will ultimately succeed or fail. Instead, it focuses on objectively describing the current environment, the actions attempted so far, their observable consequences, and the remaining task progress. The resulting execution situation summary is used to retrieve memories relevant to the agent's current execution context.

A.2 Template for Situation-Conditioned Guidance Generation

We provide the prompt templates used to convert the situation summaries into situation-conditioned guidance, represented as *if–then* experience rules. We use Gemini-2.5-Pro (Comanici et al., 2025) for all guidance generation and apply separate prompts to successful and failed experiences. For a successful experience, the generator receives only the situation summary and extracts reusable strategies that contributed to task completion. For a failed experience, it receives both the situation summary and the inferred failure reason, allowing it to generate guidance that identifies failure-prone conditions and recommends corrective or preventive actions.

For both outcomes, the generated rules pair an execution condition in the *if* clause with a concrete planning or action response in the *then* clause. The prompts also instruct the model to generalize task- or object-specific details so that the resulting guidance can be transferred across different tasks

```
{
  "images": [
    "<image> # input.png (four views, unlabeled)",
    "<image> # input_mask.png (four views, numbered)"
  ],

  "trajectory_type":
    "You are summarizing a FAILED robotic manipulation trajectory.",

  "instruction":
    "The agent was given the instruction: <instruction>",

  "Image information": [
    "The first image shows four different perspectives of objects (without labels).",
    "The second image contains the same four perspectives of objects, but each view is
     labeled with a number (representing each object's identifier)."
  ],

  "summary_instruction":
    "Summarize the agent's reasoning flow objectively based on what it actually inferred
     and excuted:",

  "Agent CoT Output":
    "<origin_output>",

  "Evaluation scores for this episode": [
    "skill_match_score : <skill_match_score> (how often predicted skill names match the
     reference, ignoring order and object)",
    "entity_match_score : <entity_match_score> (how often predicted target IDs match
     the reference, ignoring order and skill type)",
    "exact_match_score : <exact_match_score> (how well the full operation structure
     matches the reference  final task score)"
  ],

  "For reference, the types of mistakes an agent can make are": [
    "Step omission : a required step was skipped",
    "Wrong target object : failed to select the correct object as specified by the
     instruction, or incorrect object state compared to the annotated image",
    "Wrong action: choose incorrect action",
    "Wrong plan : an incorrect plan was chosen"
  ],

  "Step 1":
    "Write a concise 3-5 sentence factual summary of what the robot tried, what actually
     executed.",

  "Step 1 constraints": [
    "Use the agent skill_sequence in CoT as ground truth for what was planned/executed
     .",
    "Use the images as ground truth for the starting state of the scene.",
    "Use the scores above to understand the degree and type of failure.",
    "Do NOT reference object numbers and exact object name; describe actions as abstract
     behaviors."
  ],

  "Step 2":
    "Read the failure-cause categories listed above. On a NEW LINE write \"Failure
     reason:\" followed by a 1-2 sentence root-cause analysis."
}
```

Figure 10: Template for a VLABench situation summary of a failed trajectory input prompt.

and episodes. The complete prompts and representative outputs for successful and failed experiences are provided in Figure 13, 14.

```
{
  "trajectory_type":
    "You are summarizing an IN-PROGRESS robotic manipulation trajectory before
    replanning.",

  "Inputs you are given (in order)":
    "Write a concise 3-5 sentence factual summary of what was ACTUALLY executed and how
    the scene changed",

  "constraints": [
    "Do NOT assume success. Do NOT claim the object was picked/placed unless the latest
    annotated image supports it.",
    "Using the LATEST annotated image as primary evidence, additionally judge current
    progress state and whether the executed actions are actually succeeding toward the
    task goal.",
    "If planned intent and observed outcome mismatch, state the mismatch explicitly.",
    "Use executed_actions as ground truth for what ran; use images as ground truth for
    what changed."
  ],

  "images": [
    "<image> # Image[0]",
    "<image> # Image[1]",
    "...",
    "<image> # Image[last]"
  ],

  "structured_trajectory": {
    "Instruction": "<instruction>",

    "Plan cycle 1": {
      "visual_state_description": "<...>",
      "reasoning_and_reflection": "<...>",
      "language_plan": "<...>",
      "executable_plan": "<...>"
    },

    "...": "...",

    "Executed actions": [
      "[x, y, z, rx, ry, rz, gripper]",
      "..."
    ]
  }
}
```

Figure 11: Template for an EB-Manipulation execution situation summary input prompt.

Table 4: Experience dataset statistics for EB-Manipulation by source model.

Models	Success	Failure	Total
GPT-4o	63	165	228
InternVL3-8B	40	188	228
InternVL3-78B-AWQ	47	181	228
Total	150	534	684

Table 5: Experience dataset statistics for EB-Manipulation by task split.

Task split	Success	Failure	Total
Base	30	114	144
Common	31	113	144
Complex	28	116	144
Spatial	40	104	144
Visual	21	87	108
Total	150	534	684

```
{
  "images": [
    "<image> # input.png (four views, unlabeled)",
    "<image> # input_mask.png (four views, numbered)"
  ],

  "trajectory_type":
    "You are summarizing a IN-PROGRESS robotic manipulation task situation for memory
     retrieval.",

  "instruction":
    "The agent was given the instruction: <instruction>",

  "Image information": [
    "The first image shows four different perspectives of objects (without labels).",
    "The second image contains the same four perspectives of objects, but each view is
     labeled with a number (representing each object's identifier)."
  ],

  "summary_instruction":
    "Summarize the agent's reasoning flow and executed skill_sequence objectively,
     without assuming the plan is failed or successful:",

  "Agent CoT Output":
    "<origin_output>",

  "output_instruction":
    "Write a concise 2~3 sentence factual summary of the current task situation, the
     actions actually executed in the skill_sequence, and whether the executed target
     selection or ordering is verified by the instruction and images.",

  "constraints": [
    "Treat the skill_sequence inside the agent output as ground truth for the actions
     actually executed.",
    "Use the images as ground truth for the starting state of the scene.",
    "Carefully match the objects visible in the images with the target entities and
     actions in the skill_sequence.",
    "Do NOT reference object numbers or exact object name; describe actions as abstract
     behaviors."
  ]
}
```

Figure 12: Template for a VLABench execution situation summary input prompt.

Table 6: Experience dataset statistics for VLABench by source model.

Models	Success	Failure	Total
Claude Sonnet-4.6	89	410	499
Gemini-2.5-Flash	83	350	433
Qwen2.5-VL-7B	40	455	495
Total	212	1,215	1,427

Table 7: Experience dataset statistics for VLABench by task.

Task	Success	Failure	Total
Book Rearrange	39	245	284
Cook Dishes	78	211	289
Hammer Nail and Hang Picture	26	253	279
Chemistry Experiment	6	282	288
Texas Hold'em	63	224	287
Total	212	1,215	1,427

A.3 Experience Dataset Construction

We construct the experience datasets from trajectories generated by multiple VLM backbones to cover different model families and capacities. For EB-Manipulation (Table 4 and Table 5), we use GPT-4o (OpenAI, 2024b) as a proprietary backbone and InternVL3-8B and InternVL3-78B-AWQ (Zhu et al., 2025) as relatively small and large open-weight backbones, respectively. For VLABench (Table 6, and 7), we use Claude Sonnet (Anthropic, 2026) and Gemini-2.5-Flash (Comanici et al., 2025) as representative proprietary backbones, together with Qwen2.5-VL-7B (Bai et al., 2025b) as a compact open-weight backbone. We combine proprietary and open-weight models to obtain a broader distribution of execution outcomes: the stronger proprietary models provide a

```
{
  "trajectory_type":
    "You summarize a SUCCESSFUL robotic manipulation trajectory into 2-4 short, reusable
     planning observations.",

  "Output rules": [
    "Output ONLY the rules, nothing else.",
    "Write one rule per line.",
    "Each rule should be a concise English sentence in the style:",
    "If <observable situation>, then <helpful planning behavior>.",
    "Emphasize situations where direct execution, verification, or caution may improve
     reliability.",
    "Refer to observable scene conditions",
    "Use natural, guidance-like wording rather than strict commands."
  ],

  "Successful situation summary":
    "<situation_summary>"
}
```

Figure 13: Template for situation-conditioned guidance of a successful trajectory input prompt.

```
{
  "trajectory_type":
    "You summarize a FAILED robotic manipulation trajectory into 2-4 short, reusable
     planning observations that may help avoid similar failures.",

  "Output rules": [
    "Output ONLY the rules, nothing else.",
    "Write one rule per line.",
    "Each rule should be a concise English sentence in the style:",
    "If <observable situation>, then <more reliable or cautious behavior>.",
    "Avoid strict command-style wording such as MUST or DO NOT.",
    "Refer to observable scene conditions rather than raw coordinates."
  ],

  "Failed situation summary":
    "<situation_summary>",

  "Failure reason":
    "<failure_reason>"
}
```

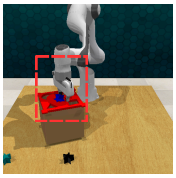
Figure 14: Template for situation-conditioned guidance of a failed trajectory input prompt.

sufficient number of successful trajectories, while the open-weight models contribute a wider range of failure cases and execution behaviors. These models were selected to provide diverse successful and failed experiences rather than to represent an exhaustive comparison of data-generation backbones. For each source model, each benchmark episode is executed once, and we retain the resulting trajectory unless the run fails to produce a valid action or the VLM output does not conform to the required response format. All situation summaries and situation-conditioned guidance were generated using Gemini-2.5-Pro.

A.4 Memory Retrieval Procedure

At each retrieval point, TERA generates an execution situation summary from the trajectory accumulated so far and compares it with the situation summaries stored in the success and failure memory banks. We concatenate each situation summary with its task instruction and encode it using `text-embedding-3-small` (OpenAI, 2024a). The similarity between the current execution situation and each stored episode is computed as

Instruction: The aroma of freshly brewed tea drifts through the room, adding a comforting touch. Please put the blue star into the shape sorter while we savor the soothing scent.



Situation Summary in Replanning step:
The first attempt dropped the blue star **onto the sorter lid**. The second attempt tried to **re-grasp the star from the lid, but this also failed**. The gripper is empty, and the blue star remains on the sorter lid after being moved slightly.

Instruction Only

Success Memory

Instruction: The light breeze from the open window carries the distant sounds of birds chirping. Could you put the green star into the shape sorter while we enjoy the peaceful soundtrack of nature?

Situation Summary: The robot first picked up a dark green star, but the placement failed and the star fell back onto the table. It then re-planned, picked up a second brighter green star, and moved it directly above the star-shaped hole.

Guidance:
- If an object's shape matches a hole on the sorter, then plan a motion to place it into that hole.
- If inserting an object into a tight-fitting hole, then align the object with the hole's orientation before attempting insertion.

Failed Memory

Instruction: The room is filled with the scent of freshly baked cookies. Can you put the red star into the shape sorter while we enjoy this delightful aroma?

Situation Summary: The robot picked up the correct red star but released it on the sorter lid. It then believed the red star was gone, switched to the blue star, and repeated the same incorrect placement. It eventually moved to another incorrect target.

Guidance:
- If an object is released, then verify the gripper is empty before perceiving the scene again.
- If a new object is targeted for grasping, then first confirm the gripper is not already holding an object.
- If an object remains in the gripper after a release attempt, then try re-gripping it before attempting to place it again

15 steps/2 replans
Failed!

Our method

Success Memory

Instruction: The gentle glow of the desk lamp casts warm light across the workspace. Please put the green star into the shape sorter as we bask in this cozy illumination.

Situation Summary: The robot successfully picked up the green star, moved it directly above the hole, lowered it vertically, and released it after the star was correctly inserted.

Guidance:
- If placing an object into a receptacle, then first align the object so it is centered directly above the opening.
- If the held object is aligned over its destination, then perform a straight downward motion for insertion.
- If an object has been successfully lowered into place, then open the gripper to release it.

Failed Memory

Instruction: A soft melody from the radio plays in the background, creating a pleasant atmosphere. Can you put the blue star into the shape sorter as we enjoy the music?

Situation Summary: The robot successfully picked up the blue star but dropped it on the sorter lid instead of into the hole. It then tried twice to re-grasp the star from the lid, but both attempts closed on empty space. The blue star remained on the sorter lid.

Guidance:
- If placing an object into a receptacle with a specific opening, then align the object with the opening before releasing.
- If releasing an object into a container, then visually check that the object is fully inside before moving away.
- If an object is in an unstable position after a failed placement, then consider moving it to a clear area before re-grasping.

13 steps/ 2 replans
Success!

Figure 15: Overview of Memory construction.

$$\mathrm{Sim}(e_j^{\mathrm{exec}}, e_i) = \frac{e_j^{\mathrm{exec}} \cdot e_i}{\|e_j^{\mathrm{exec}}\| \, \|e_i\|}, \tag{6}$$

where $e_j^{\text{exec}} = \phi([L_j; s_j^{\text{exec}}])$ represents the current execution situation and $e_i = \phi([L_i; s_i])$ represents a stored episode. TERA retrieves the two most similar episodes independently from the success and failure memory banks. The situation-conditioned guidance associated with the retrieved episodes is then inserted into the planning prompt to guide the subsequent plan generation.

B VLABench Evaluation Protocol

As described in Sec. 4.1, the selected VLABench *Complex* setting is non-interactable and does not provide intermediate trajectories for online replanning. To evaluate TERA's trajectory-aware retrieval under this constraint, we use the trajectory from the preceding evaluation attempt on the same task to construct a partial-execution query for subsequent planning. For a failed attempt, we retain the trajectory up to its termination. This approximates an execution state in which a failure has been detected and replanning would begin in an interactive environment. However, the final success or failure label and the evaluation scores are excluded from the prompt used to generate the execution situation summary.

For a successful attempt, using the complete trajectory would reveal that the task has already been completed and would include actions unavailable at an intermediate decision point. Even when the memory constructed from the same episode is excluded from the retrieval candidates, a query containing the completed execution would represent a finished solution rather than an execution in progress.

We therefore remove the final action from each successful trajectory and use the remaining prefix as a partial-progress state. The full failed trajectory or truncated successful trajectory is used only to construct the retrieval query and is never provided directly to the planner. Retrieval is restricted to other task instances of the same task type, where each task instance corresponds to a distinct instruction and object configuration, excluding the current evaluation instance. In the 500 VLABench task instances used in our evaluation, no two distinct instances within the same task type share an identical combination of manipulated-object identities and positions, although object categories may overlap. The planner receives only the retrieved situation-conditioned *if–then* guidance.

To effectively evaluate whether retrieved memory benefits planning from a partially completed trajectory, we use a protocol that provides the model with the execution history available up to an intermediate point and retrieves relevant experiences for subsequent planning. Related settings also condition decisions on partial execution histories. Mind2Web (Deng et al., 2023) evaluates each action independently while providing the ground-truth history of the preceding actions. Although these settings rely on expert or ground-truth histories rather than trajectories generated by a preceding model attempt, they illustrate the use of an intermediate state or history as the context for the next decision. Selective Hindsight Distillation (Li et al., 2026a) further distinguishes the history available at decision time from training-only hindsight, such as future trajectories and successful reference trajectories that are unavailable to the agent when selecting its current action. That work studies training-time credit assignment rather than evaluation or memory retrieval, but its distinction between decision-time information and privileged hindsight motivates our decision not to use a complete successful trajectory as the retrieval query. Our truncation protocol therefore approximates an in-progress execution state while preserving TERA's trajectory-aware retrieval mechanism.

C Implementation of Compared Methods

As described in Sec. 4.1 of the main paper, we compare the methods on EB-Manipulation, where the agent supports intermediate replanning based on observations and execution feedback. Gemini-2.5-Flash (Comanici et al., 2025) and InternVL3-38B (Zhu et al., 2025) are used as the main acting backbones. To compare TERA with representative methods that improve agent behavior by incorporating prior experiences into the prompt, we reproduce Reflexion, ExpeL, and ReasoningBank using the same EB-Manipulation agent. Since these methods were originally developed for text-based agents, we adapt them to the same VLM-based embodied agent for a fair comparison, while preserving their core reflection or memory procedures and replacing their original environment interfaces with the visual observation, reasoning, planning, and robotic action interfaces used by our agent.

Implementation of Reflexion. Reflexion (Shinn et al., 2023) improves an agent through repeated trials on the same task. After a failed trial, the agent generates a verbal reflection from its previous trajectory and environment feedback and provides this reflection as additional context for the next attempt. The reflection memory is maintained within the same task instance and does not require retrieval or an embedding-based memory bank.

Reflexion was originally evaluated on text-based household interaction in ALFWorld (Shridhar et al., 2021), multi-hop question answering in HotpotQA (Yang et al., 2018), and code generation in HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). We adapt its reflection prompt to robotic manipulation. Each failed trajectory provided to the reflection model contains the task instruction, visual state description, reasoning and reflection, language plan, executable plan, env feedback, task success. For the Gemini-2.5-Flash agent, Gemini-2.5-Pro is used to generate the reflection, while the generative reflection model is otherwise identical to the planning model, following the setup of the paper.

Reflexion performs an additional execution attempt and is therefore evaluated separately from the retrieval-based methods, including TERA, ExpeL, and ReasoningBank. Episodes successfully completed by the baseline are not re-executed. For each failed episode, Reflexion generates a verbal reflection from the failed trajectory and performs one additional attempt conditioned on the generated reflection.

Implementation of ExpeL. ExpeL (Zhao et al., 2024) first collects successful and failed trajectories and extracts a global set of reusable insights by comparing failed executions with successful executions. At evaluation time, the global insights are included in the task prompt, together with successful trajectories retrieved according to task similarity.

ExpeL was originally evaluated on multi-hop question answering in HotpotQA (Yang et al., 2018), household interaction in ALFWorld (Shridhar et al., 2021), and online shopping in WebShop (Yao et al., 2022), with additional transfer evaluation on fact verification in FEVER (Thorne et al., 2018). We preserve its insight-extraction and successful-example retrieval procedures while replacing the text-based ReAct trajectories with embodied trajectories containing the task instruction, visual state description, reasoning and reflection, language plan, executable plan and task success. The insight prompts are modified to refer to robotic manipulation, while the original rule-update procedure is retained. Gemini-2.5-Pro is used to induce a global bank of 20 insights from failure-to-success trajectory pairs.

For each evaluation episode, the current task instruction is used as the retrieval query. ExpeL retrieves the two successful trajectories whose task instructions are most similar to the current instruction from the same experience pool used by TERA. Retrieval uses normalized `all-mpnet-base-v2` embeddings and cosine similarity. The retrieved trajectories and all global insights are inserted once at the beginning of the episode. Experiences with the same episode index or an identical task instruction are excluded, and the retrieved examples are not updated at subsequent replanning stages.

Implementation of ReasoningBank. ReasoningBank (Ouyang et al., 2026) performs test-time learning with a memory bank that is initially empty. It was originally evaluated on web interaction in WebArena (Zhou et al., 2024) and Mind2Web (Deng et al., 2023), and software issue resolution in SWE-Bench-Verified (Jimenez et al., 2024). For each incoming task, it retrieves relevant memory items, executes the task, and extracts new memory items from the completed trajectory according to its final outcome.

We adapt its retrieval and memory-construction procedures to robotic manipulation. Each completed trajectory used for memory construction contains the task instruction, reasoning and reflection, executable plan and task success. The extracted items are appended to the memory bank and become available for subsequent tasks. For each EB-Manipulation split, the memory bank begins empty and is updated after every episode. At inference, the current task instruction is used as the retrieval query and compared with the task instructions associated with stored experiences. ReasoningBank retrieves the memory items associated with the most similar experience using `text-embedding-3-small` embeddings and cosine similarity, following the original top-1 retrieval setting. The acting backbone also serves as the memory-extraction model.

Table 8: Effect of the number of retrieved successful and failed experiences on InternVL3-8B performance across EB-Manipulation task types. Baseline uses no retrieved experiences; Top-1 retrieves one successful and one failed experience, while Top-2 retrieves two from each bank. Values denote success rates (%).

Method	Base	Common	Complex	Visual	Spatial	Mean
Baseline	14.6	18.8	16.7	13.9	22.9	17.5
Top-1	16.7	8.3	16.7	16.7	14.6	14.6
Top-2 (Ours)	10.4	8.3	8.3	8.3	6.3	8.3

Table 9: Effect of the number of retrieved successful and failed experiences on Ovis2-16B performance across EB-Manipulation task types. Baseline uses no retrieved experiences; Top-1 retrieves one successful and one failed experience, while Top-2 retrieves two from each bank. Values denote success rates (%).

Method	Base	Common	Complex	Visual	Spatial	Mean
Baseline	10.4	14.6	16.7	19.4	16.7	15.4
Top-1	10.4	6.3	8.3	25.0	8.3	11.8
Top-2 (Ours)	16.7	6.3	8.3	16.7	12.5	11.8

D Failure Analysis

D.1 Performance Degradation with Retrieved Guidance

As shown in Tables 8 and 9, TERA results in performance drops of 9.2 and 3.6 percentage points relative to the baseline on EB-Manipulation for InternVL3-8B (Zhu et al., 2025) and Ovis2-16B (Lu et al., 2024), respectively. Adding retrieved memories decreases the mean success rates of both models. This degradation is more pronounced for InternVL3-8B: as the retrieved context increases from one success and one failure experience to two from each bank, its mean success rate further decreases from 14.6% to 8.3%. Since InternVL3-8B has fewer parameters than Ovis2-16B, this result suggests that models with smaller parameter counts may be more sensitive to the additional prompt context introduced by retrieved guidance.

We observe three recurring failure patterns. First, although the performance degradation is more pronounced for InternVL3-8B, increasing the retrieved context from one success and one failure experience to two from each bank further reduces the mean success rate from 14.6% to 8.3%. In contrast, Ovis2-16B shows no additional decrease when the number of retrieved experiences is increased from one to two per bank. Since InternVL3-8B has a smaller parameter count, this result suggests that smaller-parameter models may be more sensitive to the longer prompt context and additional guidance introduced by retrieving more memories. As shown in Table 10, the number of format errors increases from 56 to 127 for InternVL3-8B and from 51 to 155 for Ovis2-16B as the retrieved context increases. Second, the models occasionally assign inconsistent object identities to the same visual labels across replanning steps. The attention visualization in Figure 16 suggests that retrieved textual guidance can receive greater attention than the current image, weakening grounding in the current scene. Finally, when partially relevant or conflicting guidance is retrieved, these models often fail to resolve the conflict using the current observation and instead follow an inappropriate recommendation, leading some episodes that succeed under the baseline to fail with TERA (Figure 17).

Table 10: Prompt length and format errors for InternVL3-8B and Ovis2-16B under different memory settings. Prompt lengths are reported in thousands of tokens. Baseline uses no retrieved experiences; Top-1 retrieves one successful and one failed experience, while Top-2 retrieves two from each bank.

Model	Setting	Avg. Prompt	Success Avg.	Failure Avg.	Format Errors
InternVL3-8B	Baseline	28.7K	13.8K	31.9K	56
	Top-1	35.9K	14.3K	39.5K	85
	Top-2 (Ours)	54.0K	17.8K	56.9K	127
Ovis2-16B	Baseline	28.5K	13.1K	31.4K	51
	Top-1	52.7K	12.2K	58.1K	146
	Top-2 (Ours)	59.0K	14.6K	64.9K	155

Table 11: Planning calls and elapsed task time across methods (228 EB-Manipulation tasks per setting). Planning calls count completed action-planning steps; auxiliary calls are excluded. For Reflexion, calls include the reused baseline and up to two retries, and times sum the durations of the executed trials.

Model	Method	Planning Calls	Mean Time (s)	Median Time (s)
InternVL3-38B	Ours	634	141.4	152.7
	ExpeL	665	153.0	128.4
	ReasoningBank	630	120.1	128.1
	Reflexion	1621	303.6	362.0
Gemini-2.5-Flash	Ours	528	129.3	145.1
	ExpeL	553	104.0	116.4
	ReasoningBank	528	160.8	154.2
	Reflexion	1,242	254.5	279.8

D.2 Limitations of Small-Parameter VLMs on EB-Manipulation

InternVL2-8B (OpenGVLab Team, 2024), Qwen2-VL-7B (Wang et al., 2024b), and Qwen2.5-VL-7B (Bai et al., 2025b) achieve zero success under both the baseline and TERA settings. Their primary limitation is the inability to consistently follow the executable action format required by EB-Manipulation. Each entry in `executable_plan` must encode a 7-dimensional numerical action specifying position, orientation, and gripper state. Outputs that violate this schema cannot be parsed and consequently trigger fallback random actions, preventing meaningful task progress.

As shown in Figure 18, the models exhibit different types of format errors: InternVL2-8B generates symbolic action names, Qwen2-VL-7B returns a natural-language manipulation instruction, and Qwen2.5-VL-7B produces an empty executable plan. Since these errors occur with and without retrieved guidance, the zero success rates mainly reflect limitations in generating valid low-level actions rather than an inability to retrieve relevant experiences.

This degradation is specific to the EB-Manipulation setting and is not observed for the same models on VLABench. A likely reason is the greater inference burden imposed by replanning. Averaged over the 13 backbones in Table 1 of the main paper, EB-Manipulation requires approximately 2.0K input tokens for the initial plan and additional history tokens during replanning, resulting in about 2.6K tokens overall, compared with 1.35K tokens for the single-pass VLABench setting. Moreover, VLABench uses a relatively simple skill-based action format, whereas EB-Manipulation requires precise 7-dimensional action vectors. Retrieved guidance is also repeatedly included across replanning stages rather than only once. These combined requirements appear to place a larger burden on models with relatively small parameter counts, increasing both grounding and output-format failures.

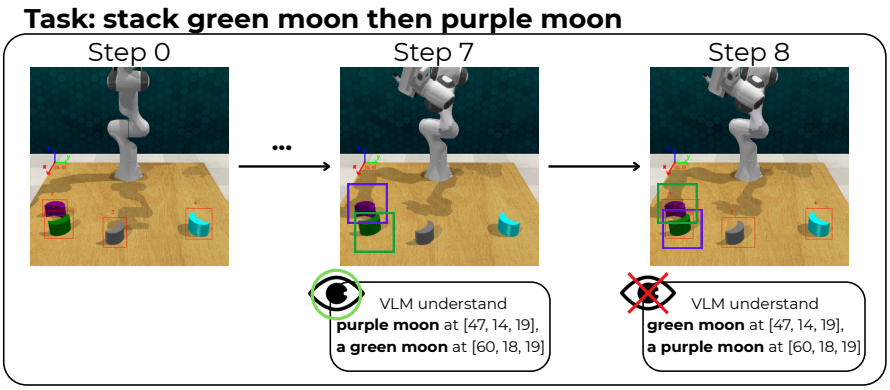


Figure 16: Qualitative result in Ovis2-16B, EB-Manipulation. Consecutive replans assign opposite color labels to the same coordinates despite a nearly identical scene(confusing green and purple).

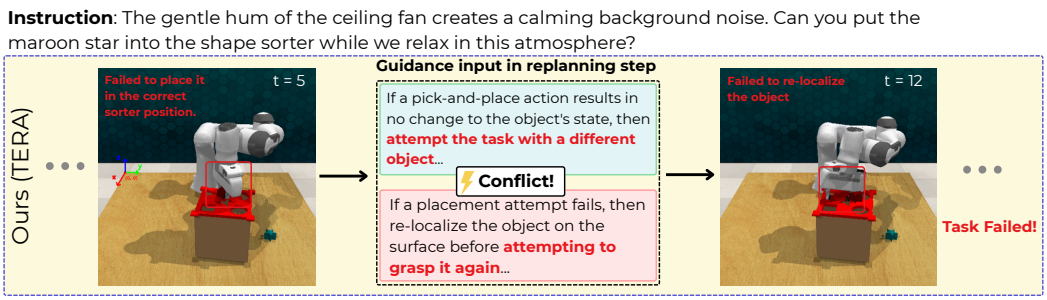


Figure 17: Qualitative result in Ovis2-16B, EB-Manipulation. Success and failure memories give opposite recovery advice (switch object vs. retry the same object), and the conflict leads to task failure.

D.3 When Retrieved Experience Hurts Planning

As reported in Sec. 4.2 and Table 1 of the main paper, Gemini-2.5-Pro decreases by 4.2 percentage points on the Common split and by 10.4 percentage points on the Spatial split. In our evaluation, repeated executions did not always produce identical initial actions even with the decoding temperature set to zero. Consequently, an episode solved by the baseline could follow a different execution path, enter replanning, and subsequently be affected by the retrieved guidance.

The larger degradation on the Spatial split is illustrated in Figure 19. The instruction identifies the objects through relative positions, asking the agent to "Stack the left star on top of the right triangular prism." However, the retrieved guidance only provides general advice about lifting and centering an object and does not help disambiguate which star and triangular prism correspond to the specified spatial relations. After replanning, the model selects the wrong target and fails the task.

A different limitation appears when comparing TERA and ExpeL on the Common split with Gemini-2.5-Flash. Although both methods improve over the baseline, TERA increases success by 2.1 percentage points, whereas ExpeL yields a larger gain of 10.4 percentage points. As shown in Figure 20, ExpeL provides diverse global guidance and a successful few-shot trajectory that support the required grasp-and-sweep execution. In contrast, TERA retrieves success- and failure-derived guidance that suggests misaligned responses. The success guidance directs the model to focus on an incorrect spatial area when identifying the target, whereas the failure guidance recommends updating the object position and replanning the grasp. These guidance cues do not jointly support correct object identification and the required left-to-right sweeping motion, making a successful grasp difficult. These results show that success and failure guidance can be complementary when they provide consistent roles—an execution strategy and a failure precaution, respectively—but misaligned guidance can make it difficult for the model to select an appropriate recovery action.

Correct output

```
"executable_plan": [
 {"action": "[51, 26, 28, 0, 60, 94, 1]"},
 {"action": "[51, 26, 18, 0, 60, 94, 0]"},
 {"action": "[51, 26, 28, 0, 60, 94, 0]"},
 {"action": "[71, 47, 27, 6, 66, 109, 1]"}]
```

Format error

InternVL2-8B

```
"executable_plan": [
 {"action": "move_to_object_position"},
 {"action": "close_gripper"},
 {"action": "move_to_object_position"},
 {"action": "open_gripper"}]
```

Qwen2-VL-7B

```
"executable_plan": [
 {"action": "Pick up the blue cube at [71, 16, 17]
  and place it into the green container at [61, 34,
  18]"}]
```

Qwen2.5-VL-7B

```
"executable_plan": []
```

Figure 18: Representative executable-plan format errors in small-parameter VLMs. EB-Manipulation requires each action to encode a 7-dimensional numerical vector. InternVL2-8B instead generates symbolic actions, Qwen2-VL-7B produces a natural-language instruction, and Qwen2.5-VL-7B returns an empty executable plan.

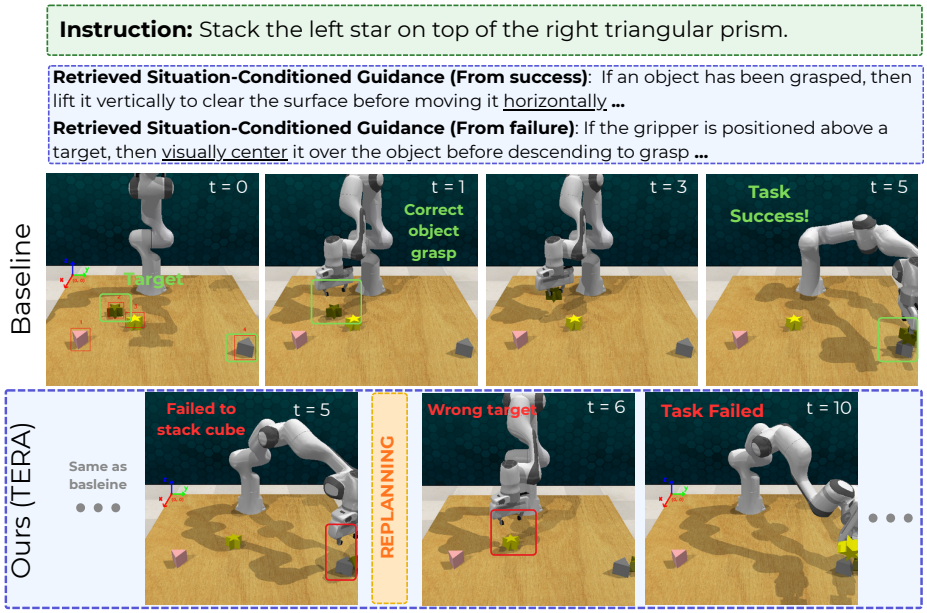


Figure 19: Qualitative comparison on EB-mamipluation. Spatial grounding failure in Gemini-2.5-Pro. The baseline correctly identifies the spatially specified objects and completes the task, whereas the retrieved guidance does not disambiguate the left–right relations. After replanning, TERA selects the wrong target object and fails.

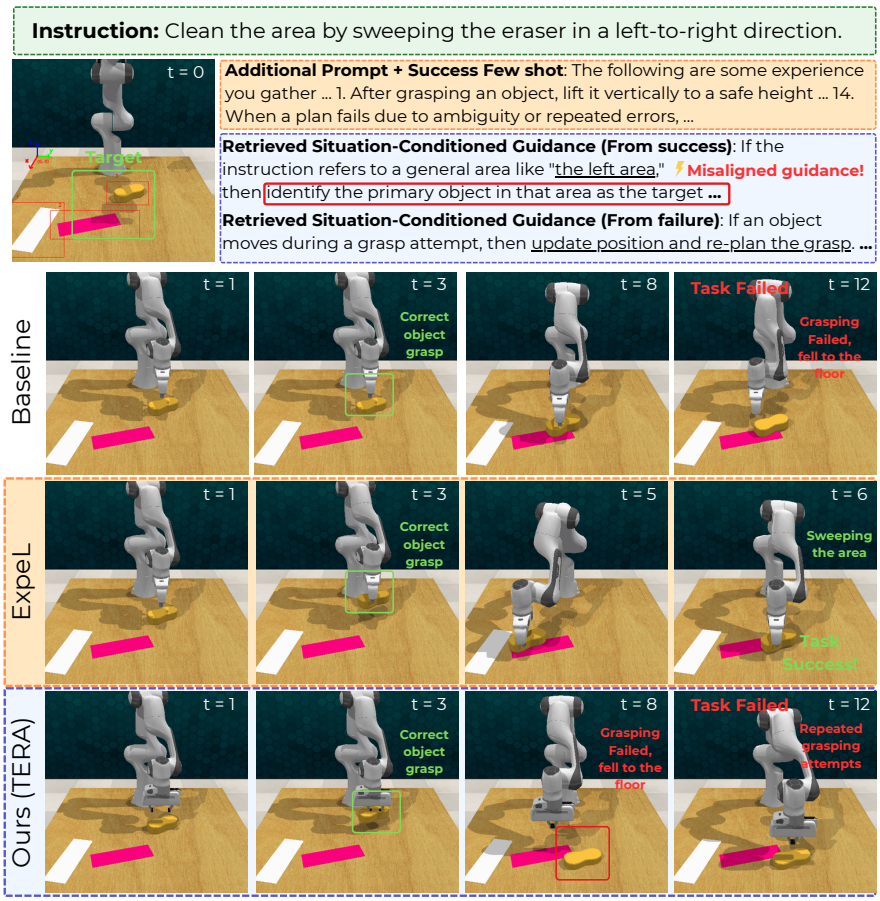


Figure 20: Qualitative comparison on EB-Manipulation. Failure caused by misaligned guidance in Gemini-2.5-Flash on the Common split. ExpeL succeeds using task-relevant global insights and a successful few-shot trajectory. In contrast, TERA retrieves success-derived guidance that fails to support correct object identification and the required left-to-right sweeping motion, making a successful grasp difficult and leading to task failure.