GOAL2SKILL: LONG-HORIZON MANIPULATION WITH ADAPTIVE PLANNING AND PROGRESS VERIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Robotic foundation models have advanced manipulation, yet long-horizon tasks remain challenging under partial observability and uncertain execution. Reliable task completion requires retaining relevant information, verifying intermediate outcomes, and adapting plans to execution feedback. We introduce **Goal2Skill**, a dual-system framework that couples adaptive planning, proactive memory, and a self-evolving verifier. A high-level VLM constructs and revises sub-task plans, while a low-level VLA executes action chunks. Periodic verification connects these processes by using current observations and task memory to determine when to continue, advance, or trigger reflective correction. Memory preserves information needed by upcoming steps and records previous attempts, supporting targeted revisions while retaining valid progress. The verifier combines lightweight models, rule-based checks, and VLM reasoning, and refines its assessment procedures through execution feedback, with candidate updates validated before adoption. Experiments on RoboMME and RMBench show that Goal2Skill achieves the highest average success rate among non-oracle methods, with particularly strong gains on tasks requiring persistent memory and multi-stage coordination.

1 INTRODUCTION

Robotic foundation models advance the range of manipulation tasks that robots can perform from natural-language instructions. Vision-language-action (VLA) models connect visual and linguistic representations to motor control, transferring semantic knowledge from pretrained models and manipulation skills from robot demonstrations (Kawaharazuka et al., 2025; Ma et al., 2024; Black et al., 2024; Kim et al., 2024). More recently, world-action models (WAMs) and video-based policies couple action generation with predictions of future observations, leveraging spatiotemporal priors to capture physical dynamics and support anticipatory control (Ye et al., 2026; Kim et al., 2026). These advances improve the generalization of learned behavior. Nevertheless, *long-horizon tasks* remain challenging because they require coordinating interdependent operations: retaining relevant information across interactions, verifying which outcomes have been achieved, and adapting subsequent decisions when execution changes the assumptions underlying an earlier plan.

Long-horizon manipulation requires coordinating planning, memory, and verification under partial observability and uncertain execution. An initial plan may need revision as conditions change, information required by later steps may become unavailable, and executed actions may fail during execution. As shown in Figure 1, a robot asked to scan a Coke can must first remember which side carries the barcode. After the can is moved to the scanner, verification reveals that the scan has failed because the barcode is not visible. Reflection attributes the failure to a wrong orientation rather than a failed grasp, so the system keeps the completed pick and move steps and rewrites only the remaining plan to rotate, align, and rescan the can. Such adaptations must combine current evidence with task history to revise the remaining plan while preserving valid progress. This example illustrates the central challenge: deciding whether to continue, advance to the next sub-task, or revise the plan.

Recent dual-system architectures and related embodied frameworks offer a promising foundation by combining high-level reasoning with low-level execution, where a high-level model interprets goals and guides execution through plans, language subgoals, or skill invocations, while a low-level policy or controller realizes these decisions through physical interaction (Team et al., 2025; Li et al., 2026b; Zhang et al., 2026; Chen et al., 2026b). However, separating planning from execution does

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

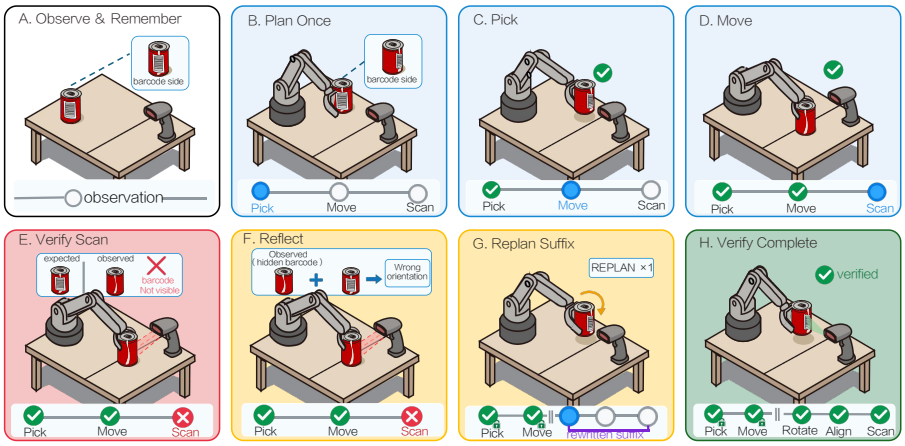

Figure 1: A long-horizon manipulation example. The robot moves a Coke can to a scanner, but the first scan fails because the barcode is not visible. Reflection diagnoses a wrong orientation, and the remaining plan is rewritten to rotate and align the can before rescanning.

not ensure reliable adaptation as task conditions change. The system must retain context for future decisions and verify execution outcomes to determine when and how to revise the plan. Crucially, the evidence needed to establish success varies across tasks and scenes, motivating a *self-evolving verifier* that refines its assessment criteria and procedures through validated execution feedback. This motivates coupling planning and proactive memory with evolving execution verification, so that improved progress assessment supports more reliable plan advancement and targeted correction.

We introduce **Goal2Skill**, a dual-system framework that couples *adaptive planning*, *proactive memory*, and *execution verification* for long-horizon manipulation. A high-level VLM brain decomposes the goal into sub-tasks, while a low-level VLA executes them through continuous visuomotor control. Periodic verification assesses task progress and determines whether to continue execution, advance to the next sub-task, or trigger reflection for local correction and replanning. Task memory supports these decisions by preserving information needed by upcoming steps and updating task context from execution outcomes. Together, these mechanisms enable the brain to adapt its plan while retaining valid progress and accounting for previous attempts.

Central to this coordination is a *self-evolving verifier* that governs task progression by determining when to continue execution, advance to the next step, or trigger correction. It combines lightweight tools, rule-based checks, and VLM reasoning to balance reliability and computational cost. Execution feedback informs iterative refinements to its assessment criteria and checking procedures, with candidate updates validated before adoption. Operating between frequent VLA execution and sparse planning, the verifier provides timely feedback for reflection and plan revision while adapting progress assessment to different tasks and scenes.

We evaluate Goal2Skill on RoboMME and RMBench tasks spanning diverse manipulation settings. It achieves the highest average success rate with particularly strong gains on tasks requiring persistent memory and multi-stage coordination. Qualitative analyses illustrate how execution feedback refines verification procedures and how reflection guides replanning when execution invalidates the initial plan. Our contributions are threefold:

- We introduce an adaptive planning mechanism that combines initial task decomposition with evidence-guided retries, local adjustments, and revision of the remaining plan.
- We develop proactive task memory with a self-evolving verifier to guide task progression and refine verification through execution feedback.
- Experiments on RoboMME and RMBench show that Goal2Skill achieves the highest average success rate with strong gains on tasks requiring persistent memory and multi-stage coordination.

2 Related Work

Long-Horizon VLA and Agentic Manipulation. VLA models unify visual perception, language understanding, and motor control. Systems such as RT-2, OpenVLA, π_0, and RDT-1B demonstrate

instruction-following and generalization capabilities through multimodal and robot-data pretraining (Zitkovich et al., 2023; Kim et al., 2024; Black et al., 2024; Liu et al., 2024). World-action models (WAMs) such as Cosmos Policy couple action generation with predicted future observations and values to support anticipatory visuomotor control (Ye et al., 2026; Kim et al., 2026). Recent work also tackles long-horizon control. For example, Long-VLA introduces phase-aware modeling to improve multi-step manipulation (Fan et al., 2025).

Complementary approaches coordinate high-level reasoning with low-level execution (Chen et al., 2026b; Ding et al., 2026). Gemini Robotics and RoboClaw combine embodied reasoning with manipulation policies (Team et al., 2025; Li et al., 2026b); LoHo-Manip updates progress-aware remaining plans (Liu et al., 2026a); and Harness VLA composes VLA behaviors and analytic primitives through memory-guided orchestration (Zhang et al., 2026). Our work focuses on memory-grounded task progression, coupling adaptive planning and proactive memory with a self-evolving verifier that uses execution evidence to guide plan advancement and targeted recovery.

Memory, Execution Verification, and Reflective Recovery. Memory-augmented policies retain information beyond the current observation. MemoryVLA combines perceptual and semantic memory (Shi et al., 2025), MEM integrates both short-term and long-term contexts (Torne et al., 2026), and ReMem-VLA carries recurrent representations across observation windows (Li et al., 2026a). RMBench further highlights the challenges of memory-dependent manipulation and the limitations of simply extending observation history (Chen et al., 2026a).

Recent work also uses execution feedback to assess progress and support recovery. ProgressVLA guides action generation with learned progress estimates (Yan et al., 2026); Phoenix and PhysReflect-VLA connect reflection with motion correction and physical consistency assessment (Xia et al., 2025; Yang et al., 2026b); and CheckVLA detects deviations through action-conditioned world-model predictions (Liu et al., 2026b). Goal2Skill couples these concerns at the task level: proactive memory preserves evidence needed by upcoming steps, while verification guides plan advancement and reflective correction. Its verifier further refines its checking criteria and procedures through validated execution feedback, supporting adaptation across tasks and scenes.

3 System Architecture

We study long-horizon robotic manipulation specified by a natural-language goal $\mathcal{G}$. At each environment step t, the robot receives an observation o_t comprising visual input and proprioceptive state, and executes a continuous motor action a_t. The objective is to maximize the probability of achieving $\mathcal{G}$ within a global execution budget of T steps.

The environment is partially observable: information needed for a decision may be available only in earlier observations or interaction outcomes. Long-horizon execution therefore requires combining current observations with retained context to coordinate interdependent stages and adapt subsequent decisions to observed outcomes.

3.1 Framework Overview

As illustrated in Figure 2, Goal2Skill adopts a dual-system architecture comprising a high-level VLM reasoning system and a low-level VLA executor. The **high-level system** coordinates adaptive planning, proactive memory, execution verification, and reflective recovery. The **low-level executor** carries out the active sub-task through visual grounding and successive action chunks conditioned on updated observations.

A **memory-grounded verifier** links low-level execution to high-level task progression. At periodic checkpoints, it assesses progress on the active sub-task: verified completion advances execution to the next sub-task; detected failures or an exhausted sub-task budget trigger reflection. A reflection module diagnoses the discrepancy and informs a retry, a local adjustment, or a revision of the remaining plan. After the final planned sub-task, the verifier checks whether the global goal has been achieved and, if necessary, initiates further recovery within the remaining execution budget. Throughout this process, memory preserves information needed by future steps and is updated with verified outcomes and previous recovery attempts.

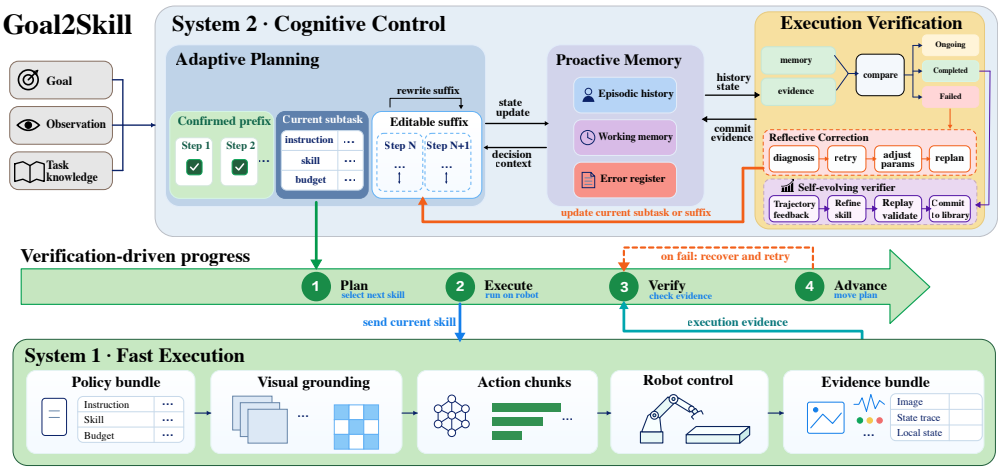

Figure 2: Overview of Goal2Skill. The high-level system integrates adaptive planning, proactive memory, progress verification, and reflective recovery. The low-level executor performs task-conditioned visual grounding and generates action chunks, returning execution evidence for verification and memory updates. The verifier guides task progression by determining when to continue execution, advance to the next sub-task, or trigger corrective reasoning. This closed loop supports targeted plan correction during long-horizon manipulation.

The framework separates three temporal scales: frequent action-chunk execution, periodic verification, and sparse planning or replanning, with reflection triggered by execution discrepancies. To balance responsiveness and computational cost, the verifier uses lightweight models or rule-based checks when sufficient and invokes VLM reasoning when needed. Through validated execution feedback, the verifier iteratively refines its skills, enabling a self-evolving verification mechanism that adapts to new tasks and supports more generalizable task progression.

3.2 High-Level Planning and Coordination

The high-level system leverages a pretrained VLM Φ as brain and coordinates adaptive planning (§ 3.2.1), proactive memory management (§ 3.2.2), progress verification (§ 3.2.3), and reflective correction (§ 3.2.4). These components share structured task memory, allowing decisions to account for the global goal, observed progress, and previous attempts. Planning occurs initially and when revision is needed, verification runs periodically during execution, and reflection is triggered when corrective intervention is required.

3.2.1 Task Decomposition and Adaptive Planning

Given a natural-language goal $\mathcal{G}$ and an initial observation o_0, the planner generates an ordered sequence of sub-tasks:

$$\mathcal{P} = \langle \tau_1, \tau_2, \ldots, \tau_K \rangle = \Phi_{\text{plan}}(\mathcal{G}, o_0). \tag{1}$$

Each sub-task is represented as $\tau_k = (\ell_k, \delta_k, j_k)$, where ℓ_k is its language instruction, δ_k sets the execution budget per attempt, and j_k selects a low-level manipulation skill.

The initial plan captures dependencies between task stages and guides subsequent execution. Verified sub-task completion advances to the next sub-task, provided its prerequisites are satisfied, without regenerating the plan. When execution of the sub-task deviates from expectations, verification and reflection guide corrections at three scopes: retrying the active sub-task, adjusting its local specification, or asking the planner to revise the remaining sub-task sequence. This allows the system to address local failures while retaining parts of the plan that remain valid.

3.2.2 Proactive Memory Management

Long-horizon manipulation tasks require retaining history contexts, information that may disappear from view, together with the outcomes of earlier interactions. We maintain an episode-level memory:

$$\mathcal{M}_t = (\mathcal{H}_t, \mathcal{W}_t, \mathcal{E}_t), \tag{2}$$

where the three components serve distinct roles.

Episodic history $\mathcal{H}_t$ maintains a compact, chronological account of task progress across sub-tasks. It summarizes previous interactions, observed changes in robot and world state, and verified outcomes. Information that remains relevant to later stages, such as a previously observed target ordering, is retained even after it leaves the current view.

Working memory $\mathcal{W}_t$ tracks the active sub-task at t-th step. It records the state observed at the start of the current sub-task, recent execution evidence, and progress toward the expected outcome of this step. Comparing the current state with this local reference helps the verifier determine what has changed and whether the intended sub-task execution has succeeded.

The error register $\mathcal{E}_t$ records failures and recovery attempts throughout the episode. Each entry includes the observed discrepancy, hypothesized cause, proposed correction, and its outcome when available. These records support reflection on both the current failure and similar previous attempts.

Memory updates follow the execution cycle. When a sub-task or a new attempt begins, working memory is initialized from the current observation and relevant episodic context. At each verification checkpoint, recent evidence and the verification result update its local progress record. When a sub-task completes, or an attempt ends in failure, its relevant information is summarized into episodic history before working memory is refreshed for the next attempt or sub-task. Recovery decisions are recorded in the error register and updated when their outcomes become available:

$$\mathcal{M}_t^+ = \Phi_{\text{mem}}\left(\mathcal{M}_t, \mathcal{P}_{t:T}, \mathcal{Z}_t, v_t\right), \tag{3}$$

where $\mathcal{M}_t^+$ denotes memory after the update, $\mathcal{P}_{t:T}$ is the remaining plan, $\mathcal{Z}_t$ contains recent observations and executed actions, and v_t is the verification output defined below. Importantly, the remaining plan makes memory maintenance proactive: it identifies information required by subsequent steps, guiding what should be retained when observed.

To bound context length, recent execution evidence is retained in a limited buffer, while older interactions are compressed into episodic summaries. Compression preserves information relevant to upcoming steps, unresolved discrepancies, and previous recoveries. This keeps working memory focused on the active sub-task without losing the global context needed for planning and reflection.

3.2.3 Memory-Grounded Progress Verification

The verifier assesses the active sub-task after every n executed action chunks, where n controls the verification interval. At the start of a sub-task, it retrieves a task-relevant verification skill from the verification skill library using the sub-task specification, current scene, and task memory. Each verification skill specifies its trigger condition, required evidence, checking procedure, and decision criterion. Each time, the verifier produces

$$v_t = (e_t, c_t) = V_{\theta_k}(\mathcal{Z}_t, \tau_k, \mathcal{M}_t), \tag{4}$$

$$c_t \in \{\text{COMPLETED}, \text{ONGOING}, \text{FAILED}\}, \tag{5}$$

where θ_k denotes the verification rule and c_t is the assessed status with reason e_t. A sub-task is COMPLETED when the evidence supports its expected outcome. It is FAILED when the evidence establishes a discrepancy requiring corrective intervention. Otherwise, it returns ONGOING when the execution is still in progress or the evidence is insufficient to confirm completion.

To balance reliability and efficiency, the verification combines rule-based checks, lightweight models, and VLM. Inexpensive checks handle directly measurable conditions, while ambiguous, history-dependent, or situationally complex judgments invoke VLM reasoning. Where necessary, evidence across multiple checkpoints is used to distinguish persistent failures from transient observations. A COMPLETED assessment advances the existing plan, while ONGOING retains the active sub-task. A FAILED assessment triggers reflection. After the final planned sub-task, the verifier also checks the global goal; completing the planned sequence alone is insufficient to declare task success.

Notably, we introduce a *self-evolving verifier mechanism* that improves its verification procedures from prior task trajectories and environment feedback. The verifier retrieves task-relevant verification skills during online execution, while accumulated trajectories are used to identify gaps and update existing skills or introduce new ones. The updated skills are validated through trajectory replay before being committed to the verification skill library. This allows the verifier to adapt its progress assessment from execution experience.

3.2.4 REFLECTIVE CORRECTION AND REPLANNING

Reflection is triggered by a detected execution failure, an exhausted sub-task budget, or an unmet global goal after the planned sequence ends. The reflection module examines the discrepancy between the intended outcome and observed execution:

$$(d_t, \rho_t, \tau_t') = \Phi_{\text{reflect}}(\mathcal{F}_t, \mathcal{M}_t), \tag{6}$$

where $\mathcal{F}_t$ contains the relevant sub-task evidence. The output comprises a diagnosis d_t, a recovery decision ρ_t, and an optional proposed modification τ_t'. The diagnosis distinguishes observed discrepancies from hypothesized causes and explains the evidence supporting the proposed correction.

The recovery decision selects among three scopes. RETRY repeats the active sub-task when another attempt is justified without changing its specification. ADJUST revises the active sub-task's instruction, grounding, or supported execution parameters (τ_t') while retaining the remaining sequence. REPLAN invokes the planner with the global goal to revise unfinished sub-tasks. Recovery generally escalates from a justified retry to local adjustment and then to broader replanning when previous corrections fail. It can proceed directly to replanning when evidence shows that the remaining sequence is no longer appropriate.

Diagnoses and proposed corrections are recorded in $\mathcal{E}_t$ when issued, then supplemented with their observed outcomes. These records help subsequent reflection avoid repeating ineffective recovery actions. Recovery actions, including retries, local adjustments, and replanning, are bounded, and all execution remains subject to the global task budget.

3.3 LOW-LEVEL EXECUTOR

The low-level executor carries out the current sub-task through closed-loop visuomotor control. At each action-generation step, it receives an observation $o_t = (I_t, s_t)$, consisting of RGB images I_t and proprioceptive state s_t, together with the sub-task instruction ℓ_k. The skill index j_k selects a manipulation policy, which generates a chunk of continuous actions.

3.3.1 TASK-CONDITIONED VISUAL FILTERING

In cluttered scenes, visually salient but task-irrelevant objects can distract the low-level policy from the target object. Following Liu et al. (2026c), the executor suppresses such regions to focus visuomotor control on the relevant objects. When a sub-task begins, a spatial grounding step uses the current image, sub-task instruction, and relevant task context to identify distractor bounding boxes $\mathcal{B}_{k,t}$. A segmentation model $\mathcal{S}$ converts these boxes into a binary mask Q_t, where one denotes a distractor pixel. A lightweight tracking module then propagates the mask across subsequent observations. The filtered image is $\hat{I}_t = I_t \odot (1 - Q_t)$, where $\odot$ denotes element-wise multiplication. This operation preserves the coordinates and pixel values of unmasked regions. Grounding is refreshed when the sub-task changes, a corrective adjustment modifies the relevant regions, or mask tracking becomes unreliable. Notably, only the action policy receives the filtered image. Original observations remain available to high-level modules, preserving access to evidence outside the executor's immediate task focus.

3.3.2 ACTION GENERATION

The executor exposes a library of manipulation policies $\Pi = \{\pi_1, \ldots, \pi_J\}$ through a common execution interface. The skill index j_k selects the policy for the active sub-task. Conditioned on the filtered image, proprioceptive state, and instruction, the policy model generates an action chunk: $\mathbf{A}_t = (a_t, \ldots, a_{t+H-1})$, where H is the prediction horizon. This interface allows task-specific policies (e.g., VLA models) to be invoked through the same sub-task representation.

Execution follows a receding-horizon procedure: the policy model generates an action chunk, obtains an updated observation, and generates the next chunk. The active sub-task is retained while its verification status remains ONGOING. Verified completion ends the sub-task, while a corrective intervention or exhaustion of the per-attempt budget δ_k interrupts execution and returns control to the high-level coordination system. Execution observations and actions are recorded for periodic verification and memory updates. This separation allows the policy to respond to local scene changes while reserving high-level planning for task decomposition and necessary revisions.

4 EXPERIMENTAL SETUP

We evaluate whether Goal2Skill improves long-horizon manipulation by maintaining task-relevant information, verifying execution progress, and adapting subsequent decisions from execution feedback. Experiments are conducted on two memory-dependent manipulation benchmarks: RoboMME (Dai et al., 2026) and RMBench (Chen et al., 2026a).

RoboMME. RoboMME benchmark comprises 16 tasks across four suites: *Counting*, *Permanence*, *Reference*, and *Imitation*. These suites assess progress tracking, object permanence, referential reasoning, and reproduction of previously observed behaviors. We evaluate all 16 tasks and highlight six representative tasks in the main paper. *PickXtimes* requires tracking repeated interactions, while *ButtonUnmaskSwap* requires retaining state changes from earlier interactions. *VideoRepick* tests recall of previously observed targets, and *VideoPlaceOrder* additionally requires preserving their temporal order. *PickHighlight* requires incorporating information revealed during interaction, while *VideoPlaceButton* requires verifying an intermediate manipulation before proceeding. Following the benchmark protocol, we use 100 training demonstrations and 50 test episodes per task, yielding 800 test episodes across the full benchmark. Complete results are provided in Appendix C.

RMBench. We include five representative tasks: two M(1) tasks, *Observe and Pick Up* and *Rearrange Blocks*, and three M(n) tasks, *Battery Try*, *Blocks Ranking Try*, and *Press Button*. The M(1) tasks emphasize relatively local memory dependencies, whereas the M(n) tasks require retaining context across multiple stages. We report per-task success rates and the unweighted mean.

Training Details. For RoboMME, we follow the original setting. The low-level policy is initialized from the $\pi_{0.5}$ with an action horizon of 20. For RMBench, we use the official $\pi_{0.5}$ fine-tuning configuration. Demonstrations are segmented into sub-tasks to provide supervision for the hierarchical execution pipeline. More implementation details are provided in Appendix A.

Baselines. For RoboMME, we compare against $\pi_{0.5}$ with and without past-action context, historical-frame sampling via MME-VLA (FrameSamp) (Dai et al., 2026), keyframe retrieval via MemER (Sridhar et al., 2025), and visual memory via SAM2Act+ (Fang et al., 2025). We also include GroundSG+QwenVL, which predicts subgoals from current observations and subgoal history, and GroundSG+Oracle, which uses simulator-provided ground-truth subgoals as an oracle reference (Dai et al., 2026). For RMBench, we include Diffusion Policy (DP) (Chi et al., 2025), ACT (Zhao et al., 2023), $\pi_{0.5}$ (Intelligence et al., 2025) and X-VLA (Zheng et al., 2025). We additionally consider memory-augmented methods including MemER (Sridhar et al., 2025).

5 RESULTS AND ANALYSIS

5.1 MAIN RESULTS ON ROBOMME

As in Table 1, Goal2Skill reaches 70.0% on the six representative tasks, outperforming the strongest non-oracle baseline by over 25 percentage points. Goal2Skill ranks first on four of these tasks and second on the remaining two. Moreover, Goal2Skill achieves a higher average success rate across all 16 RoboMME tasks than all other baselines (Table 7), demonstrating its strong performance.

The largest gains occur on *ButtonUnmaskSwap*, *VideoRepick*, and *VideoPlaceOrder*, which require retaining earlier state changes, recalling previously observed targets, and preserving temporal order, respectively. These results are consistent with its emphasis on preserving task-relevant context and using execution feedback to guide subsequent decisions.

Comparisons with baselines further clarify this pattern. Adding past actions to $\pi_{0.5}$ improves its six-task average only from 19.7% to 22.7%, suggesting that this form of temporal context alone is insufficient. GroundSG+QwenVL nearly matches Goal2Skill on *PickXtimes*, but remains substantially behind on the three tasks above. Although MemER performs better on *PickHighlight* and FrameSamp slightly outperforms Goal2Skill on *VideoPlaceButton*, Goal2Skill remains competitive on both while providing larger gains on tasks with extended state and ordering dependencies.

We further show that the overall advantage spans all three benchmark-defined difficulty levels in Figure 3. Across all 16 tasks, Goal2Skill achieves the best results when averaged over all episodes, yielding the highest non-oracle average at each level. Specifically, it retains a substan-

Table 1: Success rates (%) on RoboMME tasks. Average denotes the average over the six selected tasks. Best and second-best results among non-oracle methods are shown in **bold** and underline, respectively. Complete results on all 16 tasks are reported in Appendix Table 7. All RoboMME baselines are reproduced using the officially released checkpoints.

Method	Pick Xtimes	Button UnmaskSwap	Video Repick	Video PlaceOrder	Pick Highlight	Video PlaceButton	**Average**
• *Base Policies:*							
$\pi_{0.5}$	42.0	6.0	0.0	26.0	12.0	32.0	19.7
$\pi_{0.5}$ w/ past actions	58.0	16.0	8.0	18.0	12.0	24.0	22.7
• *Memory-Augmented VLA Policies:*							
MME-VLA (FrameSamp)	82.0	12.0	32.0	28.0	24.0	**54.0**	38.7
MemER	78.0	24.0	32.0	36.0	**82.0**	16.0	44.7
SAM2Act+	76.0	26.0	6.0	20.0	18.0	24.0	28.3
• *Dual Systems:*							
GroundSG+QwenVL	90.0	10.0	34.0	36.0	24.0	48.0	40.3
Goal2Skill	**92.0**	**50.0**	**60.0**	**90.0**	76.0	52.0	**70.0**
• *Oracle Reference:*							
GroundSG+Oracle	100.0	86.0	98.0	98.0	52.0	100.0	89.0

tial advantage on Hard *VideoPlaceOrder*, and leads on *ButtonUnmaskSwap* at all levels. These findings highlight its effectiveness in carrying historical context into later execution stages, including under more demanding conditions. Detailed results are provided in Appendix Tables 8 and 9.

5.2 Main Results on RMBench

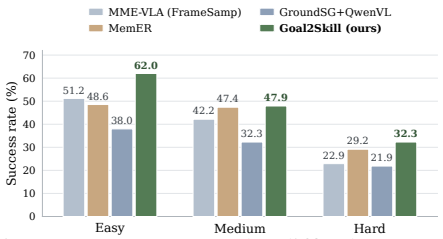

Figure 3: Success rates by difficulty, averaged over all 16 RoboMME tasks.

Table 2 shows that Goal2Skill achieves the highest success rate across the five selected RMBench tasks and the highest success rate on four of the five tasks, supporting its effectiveness beyond RoboMME. The advantage is particularly pronounced on tasks with multi-stage memory dependencies. Goal2Skill reaches 23.0% on the M(1) tasks, compared with 15.0% for the strongest baseline, and 38.7% on the M(n) tasks, compared with 9.0%. On *Battery Try*, which requires using outcomes from previous attempts to guide subsequent actions, Goal2Skill significantly outperforms the baselines. On *Blocks Ranking Try*, which requires maintaining ordering information across interactions, it reaches 60%, compared with 10% for the strongest baseline. These gains are consistent with the framework design on tracking intermediate outcomes and reusing task-relevant information in subsequent decisions.

Goal2Skill also improves *Rearrange Blocks* success rate, and achieves 10% on *Press Button*, where all compared baselines obtain zero success. On *Observe and Pick Up*, it remains close to the best baseline. Together with the RoboMME results, these findings indicate that Goal2Skill's main advantage lies in coordinating tasks that require information and execution outcomes to remain available across multiple stages. Ablations in Appendix C.1 further show the effectiveness of each module.

5.3 Qualitative Analysis

Feedback-Driven Verifier Evolution. Figure 4 illustrates how feedback improves progress assessment. The task requires three complete right-to-left swing cycles before pressing a button. The system initially infers progress from transitions in the planned subgoal sequence. These transitions do not reliably establish that the corresponding physical movements have been completed, causing the internal count to diverge from actual execution and potentially triggering an early button press.

To address this discrepancy, Goal2Skill refines the verifier with a dedicated counting procedure. It detects arrivals at the left and right target regions, suppresses repeated detections of the same arrival, and counts a cycle only after observing the required sequence of transitions. In the illustrated rollout, this procedure correctly tracks all three cycles before allowing progression to the button-press subgoal. The example demonstrates how execution failures can inform task-specific verification procedures that ground plan advancement in observable outcomes.

Table 2: Detailed results on five RMBench tasks. We report success rates (%) together with the corresponding group-wise and overall averages. Baseline results are taken from Yang et al. (2026a). To better isolate the contribution of high-level reasoning and memory, we only compare with methods that use standard low-level policies without specific architectural modifications.

Method	M(1) Tasks		M(n) Tasks			Average		
	Observe and Pick Up	Rearrange Blocks	Battery Try	Blocks Ranking Try	Press Button	M(1)	M(n)	**Total**
• *Base Policies:*								
DP	1.0	0.0	10.0	10.0	0.0	0.5	6.7	4.2
ACT	1.0	29.0	19.0	0.0	0.0	15.0	6.3	9.8
$\pi_{0.5}$	**9.0**	13.0	16.0	6.0	0.0	11.0	7.3	8.8
X-VLA	**9.0**	13.0	26.0	1.0	0.0	11.0	9.0	9.8
• *Memory-Augmented VLA Policies:*								
MemER	7.0	17.0	27.0	0.0	0.0	12.0	9.0	10.2
• *Dual Systems:*								
Goal2Skill	8.0	**38.0**	**46.0**	**60.0**	**10.0**	**23.0**	**38.7**	**32.4**

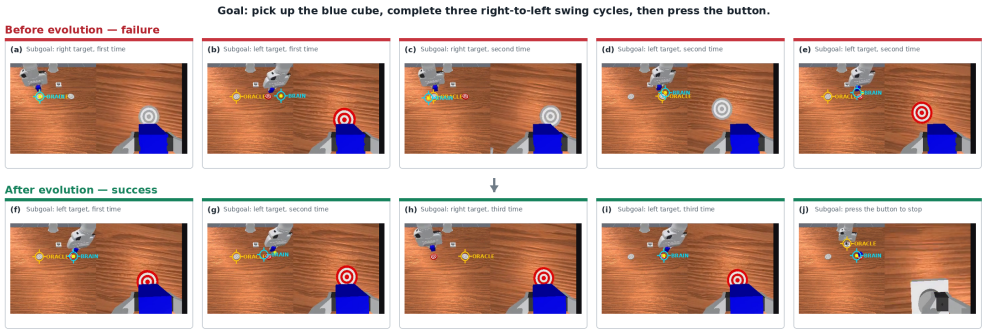

Figure 4: Verification-driven skill evolution for swing counting.

Reflection-Guided Replanning. Figure 5 shows how Goal2Skill adapts when execution invalidates an assumption of the initial plan. Early sub-tasks proceed as expected, but a later verification step reveals that the current state does not satisfy the prerequisites for the next planned action.

The discrepancy triggers reflection over the updated observation, task memory, and execution history. This diagnosis informs a revision of the remaining plan, producing a sub-task appropriate to the current state while retaining previous progress. Execution then resumes from the revised subgoal.

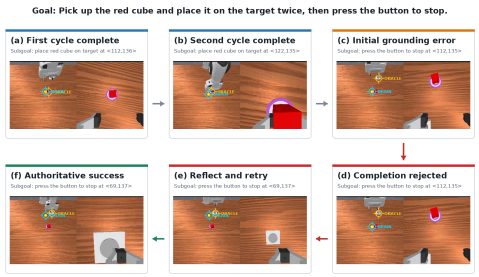

Figure 5: Reflection-based replanning after a verification failure.

This shows the complementary roles of verification and reflection: verification identifies the mismatch between expected and observed state, while reflection guides how the plan should change. The interaction enables replanning as execution unfolds. More samples are in Appendix B.1.

6 Conclusion

We presented Goal2Skill, a dual-system framework that couples adaptive planning, proactive memory, and memory-grounded verification for long-horizon manipulation. By connecting execution evidence to task progression and reflective correction, the framework supports targeted plan revisions while preserving valid progress. Feedback-driven verifier evolution further enables task-specific refinement of progress assessment. Experiments on RoboMME and RMBench demonstrate improved success, particularly when information must be retained and reused across multiple stages. Future work will explore self-improvement across more diverse tasks and embodiments.

AI USE STATEMENT

The Goal2Skill framework uses a pretrained VLM as its high-level reasoning module, and the experiments use publicly available datasets. LLM-based tools are used to assist with language editing. All technical content, experiments, and results are produced and verified by the authors, and we take responsibility for the accuracy and integrity of the final manuscript and reported results. The framework relies on pretrained generative models whose behavior may be unreliable in unfamiliar situations. Deployment in real-world environments therefore requires task-specific validation, appropriate safety constraints, and human oversight.

REFERENCES

K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

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

Yanzhe Chen, Zechen Bai, Zhijun Cao, Wenzheng Zeng, Kevin Qinghong Lin, Yiqi Lin, Guoqiang Liang, Kevin Yuchen Ma, Qiming Huang, and Mike Zheng Shou. Show-harness: Just a vlm agent can play robots. *arXiv preprint arXiv:2609.10522*, 2026b.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Yinpei Dai, Hongze Fu, Jayjun Lee, Yuejiang Liu, Haoran Zhang, Jianing Yang, Chelsea Finn, Nima Fazeli, and Joyce Chai. Robomme: Benchmarking and understanding memory for robotic generalist policies. *arXiv preprint arXiv:2603.04639*, 2026.

Xin Ding, Liang Mi, Mingzhe Huang, Zixuan Wang, Chao Zhang, Zixu Hao, Fu Chen, Xiangyu Li, Yikai Zheng, Yaoyu Guo, et al. Zetta ζ: An efficient closed-loop embodied harness for self-evolving physical intelligence. *arXiv preprint arXiv:2608.16590*, 2026.

Yiguo Fan, Pengxiang Ding, Shuanghao Bai, Xinyang Tong, Yuyang Zhu, Hongchao Lu, Fengqi Dai, Wei Zhao, Yang Liu, Siteng Huang, et al. Long-vla: Unleashing long-horizon capability of vision language action model for robot manipulation. *arXiv preprint arXiv:2508.19958*, 2025.

Haoquan Fang, Markus Grotz, Wilbert Pumacay, Yi Ru Wang, Dieter Fox, Ranjay Krishna, and Jiafei Duan. Sam2act: Integrating visual foundation model with a memory architecture for robotic manipulation. *arXiv preprint arXiv:2501.18564*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Kento Kawaharazuka, Jihoon Oh, Jun Yamada, Ingmar Posner, and Yuke Zhu. Vision-language-action models for robotics: A review towards real-world applications. *IEEE Access*, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

Hang Li, Fengyi Shen, Dong Chen, Liudi Yang, Xudong Wang, Jinkui Shi, Zhenshan Bing, Ziyuan Liu, and Alois Knoll. Remem-vla: Empowering vision-language-action model with memory via dual-level recurrent queries. *arXiv preprint arXiv:2603.12942*, 2026a.

Ruiying Li, Yunlang Zhou, YuYao Zhu, Kylin Chen, Jingyuan Wang, Sukai Wang, Kongtao Hu, Minhui Yu, Bowen Jiang, Zhan Su, et al. Roboclaw: An agentic framework for scalable long-horizon robotic tasks. *arXiv preprint arXiv:2603.11558*, 2026b.

Isabella Liu, An-Chieh Cheng, Rui Yan, Geng Chen, Ri-Zhao Qiu, Xueyan Zou, Sha Yi, Hongxu Yin, Xiaolong Wang, and Sifei Liu. Long-horizon manipulation via trace-conditioned vla planning. *arXiv preprint arXiv:2604.21924*, 2026a.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

Yushan Liu, Peibo Sun, Xintao Chao, Zhenyang Yang, Yifan Xie, Lingfeng Zhang, Shoujie Li, Chenyu Tang, Fang Chen, Xiao-Ping Zhang, et al. Checkvla: Execution-time verification with action-conditioned world model for long-horizon mobile manipulation. *arXiv preprint arXiv:2607.26789*, 2026b.

Zhen Liu, Xinyu Ning, Zhe Hu, XinXin Xie, Yitong Liu, and Zhongzhu Pu. Hsc-vla: Hierarchical scene-clearing for robust bimanual manipulation in dense clutter. *arXiv preprint arXiv:2603.07484*, 2026c.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied ai. *arXiv preprint arXiv:2405.14093*, 2024.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Memer: Scaling up memory for robot control via experience retrieval. *arXiv preprint arXiv:2510.20328*, 2025.

Gemini Robotics Team, Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, et al. Mem: Multi-scale embodied memory for vision language action models. *arXiv preprint arXiv:2603.03596*, 2026.

Wenke Xia, Ruoxuan Feng, Dong Wang, and Di Hu. Phoenix: A motion-based self-reflection framework for fine-grained robotic action correction. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6981–6990. IEEE, 2025.

Hongyu Yan, Qiwei Li, Jiaolong Yang, and Yadong Mu. Progressvla: Progress-guided diffusion policy for vision-language robotic manipulation. *arXiv preprint arXiv:2603.27670*, 2026.

Ganlin Yang, Zhangzheng Tu, Yuqiang Yang, Sitong Mao, Junyi Dong, Tianxing Chen, Jiaqi Peng, Jing Xiong, Jiafei Cao, Jifeng Dai, et al. Eventvla: Event-driven visual evidence memory for long-horizon vision-language-action policies. *arXiv preprint arXiv:2606.20092*, 2026a.

Jiayu Yang, Tao Yang, Weijun Li, Xiang Chang, Fei Chao, Changjing Shang, and Qiang Shen. Physreflect-vla: Physical feasibility and self-reflective regulation for reliable vision-language-action policies. *arXiv preprint arXiv:2606.27146*, 2026b.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, et al. Harness vla: Steering frozen vlas into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

A Training Details

A.1 RoboMME Training Details

For RoboMME, we follow the low-level policy training protocol of RoboMME (Dai et al., 2026). We initialize the low-level policy from the $\pi_{0.5}$ backbone with the released LIBERO configuration and train a single policy jointly over all 16 RoboMME tasks. Unless stated otherwise, the optimization and policy configurations follow the original RoboMME implementation. The complete training configuration is summarized in Table 3. For RoboMME, we implement our method and reproduce all baselines using the officially released checkpoints.

Table 3: Training configuration of the low-level policy on RoboMME. We follow the original RoboMME training recipe and incorporate task-conditioned masked demonstrations into the joint multi-task training data.

Hyperparameter	Value
Base policy	$\pi_{0.5}$
Policy configuration	LIBERO
Number of training tasks	16
Demonstrations per task	100
Masked demonstrations per task	20
Unmasked demonstrations per task	80
Training scheme	Joint multi-task training
Action horizon	20
Batch size	64
Optimizer	AdamW
β_1	0.9
β_2	0.95
Weight decay	0
Gradient clip norm	1.0
Learning-rate schedule	Constant
Warmup ratio	0.05
Learning rate	5×10^{-5}
EMA decay	0.999
Training steps	80,000
Checkpoint interval	10,000
Frozen module	SigLIP2 vision encoder
Training hardware	$8\times$ NVIDIA A800 GPUs

Masked training data. To expose the low-level policy to the task-conditioned visual filtering used by Goal2Skill, we apply the masking procedure to 20 of the 100 training demonstrations for each task. The remaining 80 demonstrations retain their original visual observations. Thus, the total number of training demonstrations remains unchanged at 100 per task; the masked and unmasked demonstrations are jointly used to train a single multi-task policy over all 16 tasks. This procedure does not introduce any additional parameters into the base model.

A.2 RMBench Training Details

For RMBench, we follow the official $\pi_{0.5}$ fine-tuning configuration used by the benchmark. The low-level policy is initialized from the $\pi_{0.5}$ base checkpoint and trained using the benchmark training recipe. For each evaluated task, 50 expert demonstrations are used for policy training. The complete configuration is summarized in Table 4. For RMBench, we report the baseline results from Yang et al. (2026a) and follow the same inference settings for a consistent comparison.

Masked training data. For each RMBench task, we apply task-conditioned visual masking to 20% of the training demonstrations. Specifically, 10 of the 50 demonstrations are processed with the masking procedure, while the remaining 40 demonstrations retain their original visual observations. The total number of demonstrations therefore remains unchanged, and the masked and

Table 4: Training configuration of the low-level policy on RMBench. We follow the official $\pi_{0.5}$ training configuration and incorporate task-conditioned masked demonstrations into the training data.

Hyperparameter	Value
Base policy	$\pi_{0.5}$
Initialization	$\pi_{0.5}$ base checkpoint
Demonstrations per task	50
Masked demonstrations per task	10 (20%)
Unmasked demonstrations per task	40 (80%)
Batch size	64
Optimizer	AdamW
β_1	0.9
β_2	0.95
Weight decay	1×10^{-10}
Gradient clip norm	1.0
Learning-rate schedule	Cosine decay
Warmup steps	1,000
Peak learning rate	2.5×10^{-5}
Final learning rate	2.5×10^{-6}
EMA decay	0.99
Training steps	20,000
Precision	bfloat16

unmasked demonstrations are jointly used for policy fine-tuning. This procedure does not introduce any additional parameters into the base model.

A.3 High-Level Inference and Execution Details

VLM backend. All high-level reasoning modules use a single online VLM, `qwen3.8-flash`, accessed through the OpenRouter API. The model is used for task planning and replanning, semantic progress verification, and reflective correction. During execution, the verifier combines recent execution evidence with task memory and the expected outcome of the active sub-task to determine its progress status.

Verification cadence. Following the verification procedure described in Sec. 3.2.3, the active sub-task is verified after every n executed action chunks. A verification result is classified as COMPLETED, ONGOING, or FAILED. A COMPLETED result advances execution to the next sub-task, whereas ONGOING retains the current sub-task and FAILED triggers reflective correction.

Execution budgets. Each episode is subject to a global execution budget of $T = 1300$ environment steps. Each sub-task is assigned a per-attempt execution budget of $\delta_k = 320$ steps. If the active sub-task exhausts this budget without verified completion, low-level execution is interrupted and control returns to the high-level system for reflection and recovery.

Recovery limits. Reflection selects among three recovery scopes: RETRY, ADJUST, and REPLAN. A sub-task may be attempted at most $K_{\text{sub}} = 3$ times. After the planned sub-task sequence is exhausted, the verifier performs a final check of the global goal. If the goal remains unverified, reflection and replanning are allowed for at most three additional rounds, subject to the remaining global execution budget.

B Case Study and Trajectory Analysis

B.1 Case Study on RoboMME Trajectories

We further conduct qualitative analysis on six representative cases from RoboMME to examine how verification and reflection contribute to long-horizon task execution. The selected cases span

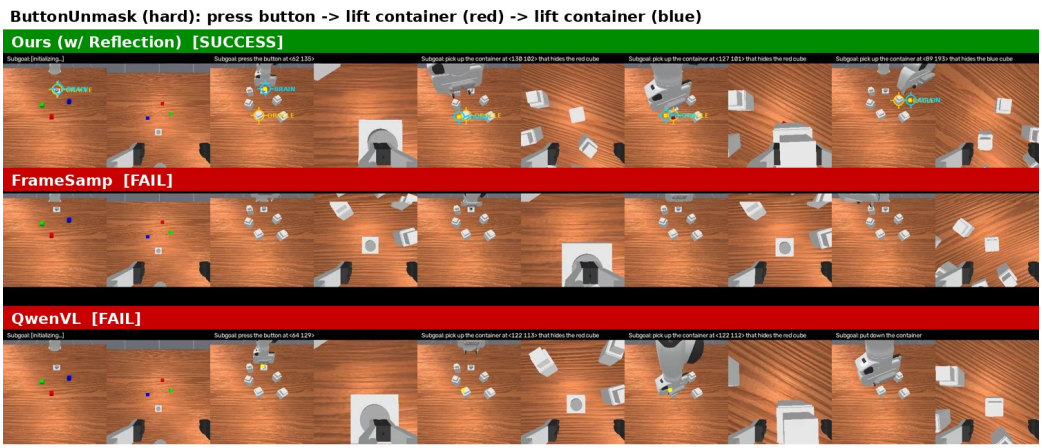

Figure 6: Reflection-driven recovery on BUTTONUNMASK (hard) from RoboMME. The initial plan terminates while the container covering the blue cube remains unopened. Reflection identifies the inconsistent completion record and revises the remaining execution.

six different tasks and multiple difficulty levels, covering both *verification-driven progression* and *reflection-driven recovery*. In all six cases, the full Goal2Skill framework successfully completes the task, whereas the compared baselines fail because of incorrect intermediate execution, premature task progression, or an inability to recover from an incomplete plan. Together, these examples illustrate how verification grounds task transitions in execution evidence, while reflection revises the remaining plan when the recorded progress is inconsistent with the task outcome.

ButtonUnmask (hard). As shown in Figure 6, the task requires the robot to press a button and subsequently uncover the containers hiding the red and blue cubes. Goal2Skill executes the initially planned stages, but the episode remains unsuccessful after the plan is exhausted. Reflection then revisits the recorded execution history and identifies a false completion: the sub-task for uncovering the blue cube had been marked as completed although its intended outcome was not confirmed. The system therefore revises the unfinished execution and repeats the corresponding uncovering operation, eventually completing the task.

In the same case, FrameSamp produces little effective task-directed motion, while QwenVL generates intermediate sub-goals but fails to physically complete the required manipulation, leaving the blue container unopened. This example demonstrates how reflection can recover when a recorded sub-task completion is inconsistent with the actual task state.

PatternLock (easy). As illustrated in Figure 7, the robot must use the attached stick to reproduce a demonstrated trajectory. Although the initial plan reaches its final stage, the global task remains incomplete. Reflection examines the final recorded stage and determines that the "move backward-left" motion connects the red disk from the left-middle region toward the bottom-middle region, but does not complete the target pattern. The missing trajectory segment is subsequently replanned, and Goal2Skill completes the required path.

By comparison, MemER deviates from the target trajectory after reaching the central region and fails to complete the final segment, resulting in failure. QwenVL makes limited effective progress and eventually times out at the step budget. This case highlights the ability of reflection to recover from execution that appears locally plausible but remains globally incomplete.

VideoUnmask (medium). As shown in Figure 8, the task requires the robot to infer the target from a demonstration video and uncover the container hiding the green cube. After the container-lifting sub-task is executed, the initial plan terminates without achieving the task goal. Reflection identifies that the corresponding sub-task had been recorded as successful despite the absence of task-level confirmation. The completion record is therefore treated as invalid, and the unfinished manipulation

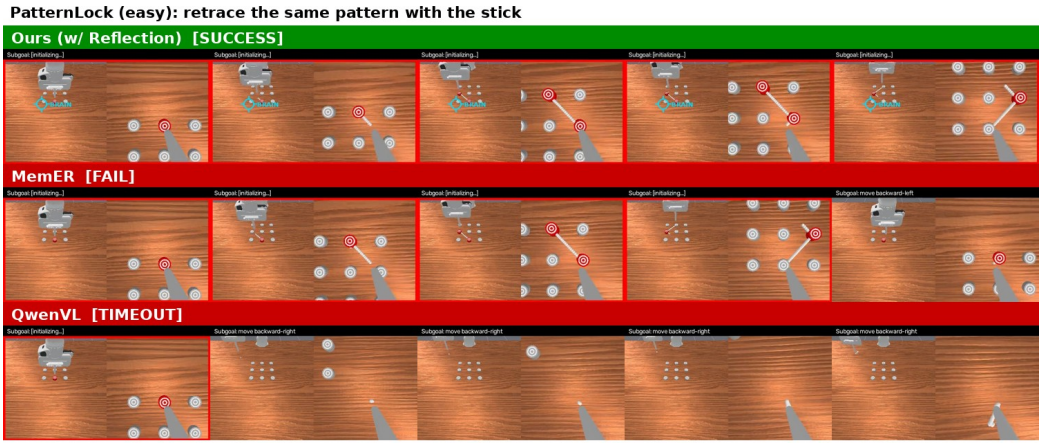

Figure 7: Reflection-driven correction on PATTERNLOCK (easy) from RoboMME. Reflection detects that the demonstrated trajectory remains incomplete after the initial plan terminates and replans the missing motion segment.

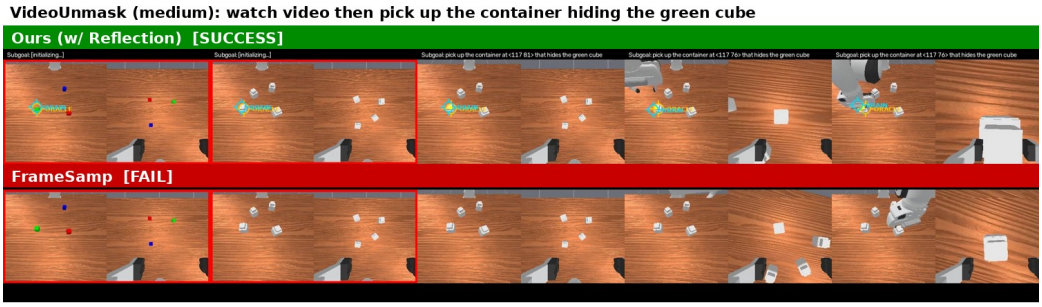

Figure 8: Reflection-driven recovery on VIDEOUNMASK (medium) from RoboMME. A nominally completed container-lifting stage is invalidated by the unsuccessful terminal task state, triggering replanning until the target cube is uncovered.

is replanned. After bounded replanning attempts, the execution is corrected sufficiently to uncover the target cube.

FrameSamp, in contrast, fails to extract an effective target from the demonstration and does not manipulate the correct container, resulting in failure. This example illustrates how terminal task evidence can expose a false intermediate completion, after which reflection can redirect execution toward the unresolved operation.

VideoPlaceOrder (medium). As shown in Figure 9, the task requires the robot to observe a demonstration video and place the blue cube at the second target location used in the demonstration. During execution, the verifier compares the evidence at the beginning and end of the placement stage: the blue cube is initially located on the table and is later observed at the intended target position. This evidence supports the expected postcondition of the placement sub-task, allowing the verifier to mark the stage as COMPLETED and safely advance the plan.

In contrast, FrameSamp and QwenVL fail to correctly resolve the "second target position" in this case, leading to an incorrect placement or stalled execution. The example shows that explicit verification of the physical outcome prevents task progression from relying solely on whether a placement action has been issued.

PickHighlight (easy). As illustrated in Figure 10, the task first requires pressing a button and then picking all cubes marked by newly appearing white highlights. Because the highlighted targets are not available before the button interaction, the initial plan contains only the

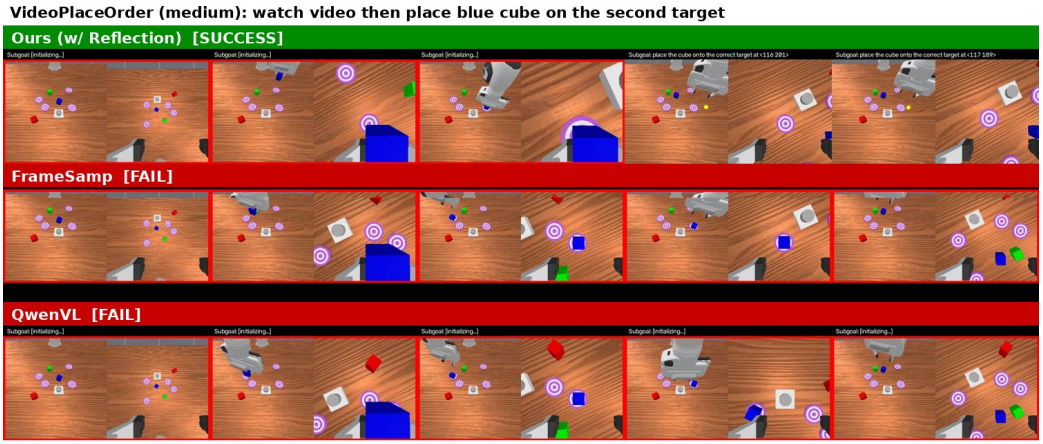

Figure 9: Verification-driven progression on VIDEOPLACEORDER (medium) from RoboMME. The verifier compares stage-level execution evidence and confirms that the blue cube reaches the required target location before advancing the plan.

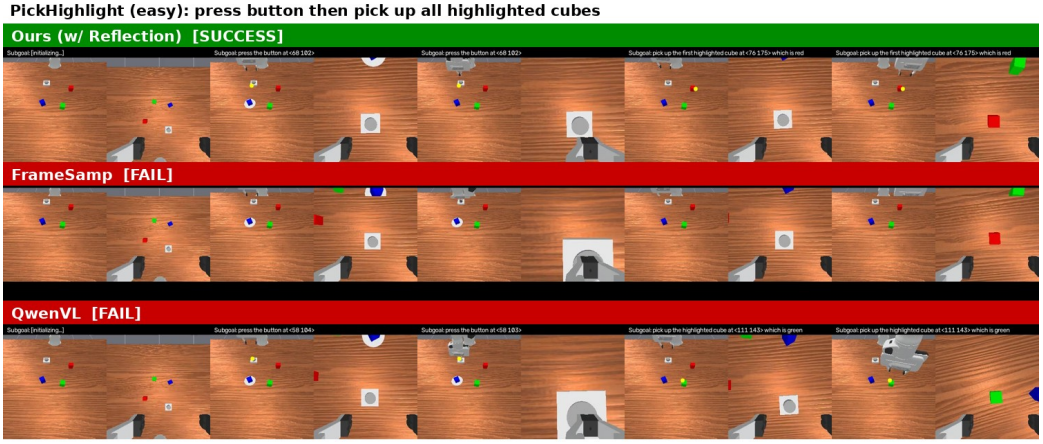

Figure 10: Verification- and reflection-guided plan expansion on PICKHIGHLIGHT (easy) from RoboMME. Verification detects the scene transition induced by the button press, after which reflection extends the plan to manipulate the newly highlighted cubes.

button-press stage. Verification confirms that the button press has produced the expected scene transition and that the highlight regions have appeared. Reflection then triggers replanning (`reason=pick_highlight_post_button`) and introduces two additional manipulation subtasks for the highlighted cubes, after which the task is successfully completed.

FrameSamp and QwenVL, by comparison, terminate their effective execution after the button interaction and do not proceed with the newly revealed manipulation stage. This case demonstrates the complementary roles of verification and reflection: verification establishes that the scene has entered a new task state, while reflection uses the newly available information to extend the unfinished plan.

VideoPlaceButton (easy). As shown in Figure 11, the task requires the robot to interpret a demonstration video and place the green cube at the target location after the button-triggered event. Before transitioning from grasping to placement, the verifier compares the stage-start observation with the current state. The green cube is initially located on the table, whereas the current observation shows that it is securely held by the gripper. This evidence satisfies the expected grasping postcondition, allowing the plan to advance to the placement stage.

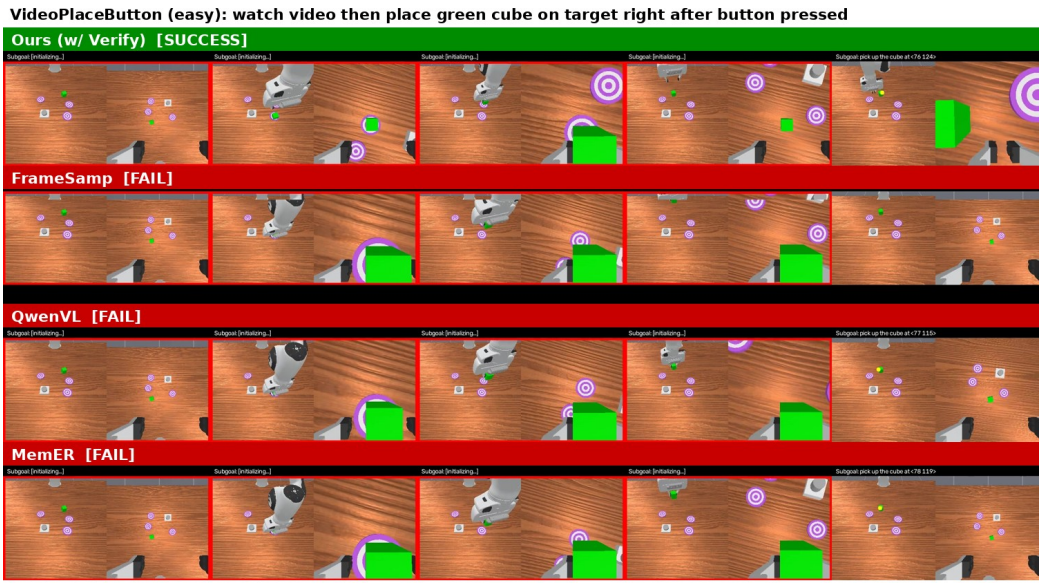

Figure 11: Verification-driven stage transition on VIDEOPLACEBUTTON (easy) from RoboMME. The verifier confirms that the green cube has been securely grasped before execution proceeds to the placement stage.

In the corresponding baseline executions, FrameSamp, QwenVL, and MemER all complete the button interaction but fail during grasping. FrameSamp positions the gripper away from the cube, QwenVL remains above the target without establishing a grasp, and MemER keeps the gripper noticeably separated from the object. These trajectories illustrate why explicit verification of grasp completion is important before transitioning to the subsequent placement stage.

Summary. The six RoboMME cases reveal two complementary benefits of the proposed closed-loop framework. First, in VIDEOPLACEORDER and VIDEOPLACEBUTTON, verification grounds sub-task transitions in observed execution outcomes. By requiring evidence that the expected post-condition has actually been achieved, the system avoids advancing the plan based only on nominal action execution.

Second, in BUTTONUNMASK, PATTERNLOCK, VIDEOUNMASK, and PICKHIGHLIGHT, reflection addresses mismatches between the current plan and the actual task state. It can invalidate an incorrect completion record, recover missing operations, or extend the unfinished plan when new task-relevant information becomes available. Together, verification and reflection connect physical execution outcomes with high-level task progression and targeted recovery, providing a closed-loop mechanism for long-horizon manipulation.

B.2 CASE STUDY ON RMBENCH

Figure 12 shows representative rollouts on the RMBench tasks. The examples show coherent progression across observation, rearrangement, trial-based interaction, ranking, and final goal verification. This visual evidence is consistent with the numerical results in Table 2: the framework improves not only the final success statistic, but also illustrates coherent task progression across stages.



Figure 12: Sample trajectories of our framework on RMBench tasks. From top to bottom, the task sequences correspond to Observe and Pick Up, Rearrange Blocks, Battery Try, Blocks Ranking Try, and Press Button. Each row shows key frames from one rollout and illustrates that the method can complete representative memory-dependent manipulation tasks with different sequential interaction requirements.

Table 5: Memory ablation on RMBench. We progressively remove components of structured memory to evaluate their contributions to tasks requiring target retention, intermediate state tracking, or multi-stage bookkeeping.

Setting	Observe and Pick Up	Rearrange Blocks	Blocks Ranking Try	Avg.
Goal2Skill	8%	38%	60%	35.3%
$-\mathcal{E}_t$	6%	36%	42%	28.0%
$-\mathcal{E}_t - \mathcal{W}_t$	8%	21%	54%	27.7%
$-\mathcal{E}_t - \mathcal{W}_t - \mathcal{H}_t$	4%	10%	6%	6.7%

C Additional Experimental Results and Analysis

C.1 Ablation Study

We conduct targeted ablations on RMBench tasks to examine the contributions of structured memory and reflective correction. The task subsets are chosen to emphasize the capabilities associated with each component, including long-horizon state tracking and recovery from failed execution.

Structured memory. As shown in Table 5, the full Goal2Skill model achieves an average success rate of 35.3%. Removing the error register $\mathcal{E}_t$ reduces the average to 28.0%, with *Blocks Ranking Try* decreasing from 60% to 42%. This suggests that retaining failure and recovery information contributes to robust long-horizon behavior, particularly when earlier execution outcomes are relevant to later decisions. Further removing working memory $\mathcal{W}_t$ results in an average success rate of 27.7%. Although the overall average changes only slightly, the effect varies across tasks: *Rearrange Blocks* decreases from 36% to 21%, while *Blocks Ranking Try* increases from 42% to 54%. This indicates that working memory is particularly useful for tracking evolving intermediate states, but its contribution is task dependent. Finally, removing episodic history $\mathcal{H}_t$ as well reduces the model without any structured memory, causing the average success rate to drop sharply from 27.7% to 6.7%. This setting even falls slightly below the $\pi_{0.5}$ baseline, likely because the planner, without any record of completed sub-tasks, can no longer track progress and tends to repeat finished steps or advance prematurely, issuing sub-task instructions that conflict with the actual scene and disrupt the low-level policy. This further indicates that task-level planning is only beneficial when grounded in persistent memory. The largest decrease occurs on *Blocks Ranking Try*, from 54% to 6%, highlight-

Table 6: Reflection ablation on RMBench.

Setting	Battery Try	Press Button	Avg.
Goal2Skill	46%	10%	28.0%
−Reflection	30%	5%	17.5%

Table 7: Complete per-task success rates (%) on all 16 RoboMME tasks (Dai et al., 2026). Best and second-best results among non-oracle methods are shown in **bold** and underline, respectively. GroundSG+Oracle is included as an oracle reference.

Method	Counting				Permanence				Reference				Imitation				Overall
	Bin Fill	Pick Xtimes	Swing Xtimes	Stop Cube	Video Unmask	Video Unmask Swap	Button Unmask	Button Unmask Swap	Pick Highlight	Video Repick	Video Place Button	Video Place Order	Move Cube	Insert Peg	Pattern Lock	Route Stick	
Base Policies																	
$\pi_{0.5}$	30.0	42.0	36.0	6.0	20.0	18.0	22.0	6.0	12.0	0.0	32.0	26.0	26.0	2.0	2.0	4.0	17.8
$\pi_{0.5}$ w/ past actions	26.0	58.0	26.0	4.0	30.0	20.0	24.0	16.0	12.0	8.0	24.0	18.0	34.0	2.0	4.0	6.0	19.5
Memory-Augmented VLA Policies																	
MME-VLA (FrameSamp)	38.0	82.0	**94.0**	**50.0**	38.0	28.0	26.0	12.0	24.0	32.0	**54.0**	28.0	48.0	2.0	**54.0**	**66.0**	42.3
MemER	**66.0**	78.0	60.0	0.0	86.0	**36.0**	**78.0**	24.0	**82.0**	32.0	16.0	36.0	**82.0**	2.0	14.0	6.0	43.6
SAM2Act+	40.0	76.0	26.0	0.0	28.0	18.0	32.0	26.0	18.0	6.0	24.0	20.0	30.0	0.0	0.0	0.0	21.5
Dual Systems																	
GroundSG+QwenVL	48.0	90.0	6.0	0.0	**88.0**	28.0	26.0	10.0	24.0	34.0	48.0	36.0	68.0	**6.0**	8.0	4.0	32.8
Goal2Skill	58.0	**92.0**	56.0	2.0	84.0	24.0	70.0	**50.0**	76.0	**60.0**	52.0	**90.0**	68.0	2.0	32.0	8.0	**51.5**
Oracle Reference																	
GroundSG+Oracle	90.0	100.0	100.0	44.0	98.0	96.0	98.0	86.0	52.0	98.0	100.0	98.0	90.0	10.0	100.0	64.0	82.8

ing the importance of persistent chronological task history for delayed target recall and multi-stage bookkeeping. Overall, the results show that the three memory components provide complementary functionality, with the complete structured memory achieving the strongest overall performance.

Reflection. Table 6 evaluates the contribution of reflective correction. The full model achieves 28.0% on these two tasks, reaching 46% on *Battery Try* and 10% on *Press Button*. Removing reflection reduces the average to 17.5%, with *Battery Try* decreasing from 46% to 30% and *Press Button* from 10% to 5%. Although failures are still detected, they can only be addressed by re-executing the same sub-task, without diagnosing the cause or revising the plan. This suggests that verification signals are most effective when translated into explicit corrective actions such as adjusting execution or replanning.

C.2 Complete RoboMME Results

Complete RoboMME results. Table 7 shows that the trend in the main-text comparison extends to the full RoboMME benchmark. Goal2Skill achieves the highest overall success rate among non-oracle methods at 51.5%, followed by MemER at 43.6% and MME-VLA (FrameSamp) at 42.3%. Goal2Skill also ranks among the top two non-oracle methods on 12 of the 16 tasks.

The results differ across memory categories. Goal2Skill averages 57.0% on *Permanence* and 69.5% on *Reference*. The gain is most pronounced on *Reference*, where MemER reaches 41.5%. These tasks require information observed earlier in the episode to remain available for later decisions. This pattern is consistent with the role of persistent task memory in Goal2Skill.

On *Counting*, Goal2Skill reaches 52.0%, while FrameSamp achieves 66.0%. On *Imitation*, Goal2Skill reaches 27.5%, compared with 42.5% for FrameSamp. The task-level results show a similar division: Goal2Skill is strong when success depends on maintaining task state over time, while methods with strong perceptual history remain competitive on tasks such as *SwingXtimes*, *PatternLock*, and *RouteStick*. Overall, the full benchmark shows that the main gain of Goal2Skill comes from long-horizon state tracking and coordination.

C.3 Difficulty-Stratified RoboMME Results

Difficulty-stratified RoboMME results. Tables 8 and 9 show how the methods behave as task difficulty increases. Success rates generally decrease from Easy to Hard episodes, but the change varies considerably across tasks.

Table 8: Difficulty-stratified evaluation on the Counting and Permanence tasks of RoboMME (Dai et al., 2026). We report success rates (%) for Easy, Medium, and Hard episodes. Each task contains 50 test episodes, including 26 Easy, 12 Medium, and 12 Hard episodes. GroundSG+Oracle is included as an oracle reference.

Method	Difficulty	Counting				Permanence			
		Bin Fill	Pick Xtimes	Swing Xtimes	Stop Cube	Video Unmask	Video Unmask Swap	Button Unmask	Button Unmask Swap
Base Policies									
$\pi_{0.5}$	Easy	38.5	46.2	34.6	7.7	23.1	19.2	23.1	7.7
	Medium	25.0	41.7	41.7	8.3	16.7	16.7	25.0	8.3
	Hard	16.7	33.3	33.3	0.0	16.7	16.7	16.7	0.0
$\pi_{0.5}$ w/ past actions	Easy	34.6	61.5	26.9	3.8	34.6	19.2	26.9	19.2
	Medium	25.0	66.7	25.0	8.3	25.0	25.0	25.0	16.7
	Hard	8.3	41.7	25.0	0.0	25.0	16.7	16.7	8.3
Memory-Augmented VLA Policies									
MME-VLA (FrameSamp)	Easy	53.8	92.3	96.2	53.8	46.2	34.6	34.6	15.4
	Medium	41.7	100.0	100.0	41.7	25.0	33.3	33.3	16.7
	Hard	0.0	41.7	83.3	50.0	33.3	8.3	0.0	0.0
MemER	Easy	84.6	96.2	53.8	0.0	96.2	34.6	84.6	34.6
	Medium	58.3	83.3	83.3	0.0	91.7	50.0	91.7	25.0
	Hard	33.3	33.3	50.0	0.0	58.3	25.0	50.0	0.0
SAM2Act+	Easy	50.0	80.8	26.9	0.0	30.8	19.2	34.6	30.8
	Medium	33.3	83.3	25.0	0.0	25.0	16.7	33.3	25.0
	Hard	25.0	58.3	25.0	0.0	25.0	16.7	25.0	16.7
Dual Systems									
GroundSG+QwenVL	Easy	76.9	92.3	7.7	0.0	100.0	26.9	38.5	11.5
	Medium	16.7	91.7	0.0	0.0	91.7	33.3	25.0	16.7
	Hard	16.7	83.3	8.3	0.0	58.3	25.0	0.0	0.0
Goal2Skill	Easy	73.1	96.2	73.1	3.8	92.3	34.6	84.6	65.4
	Medium	50.0	91.7	50.0	0.0	83.3	16.7	66.7	41.7
	Hard	33.3	83.3	25.0	0.0	66.7	8.3	41.7	25.0
Oracle Reference									
GroundSG+Oracle	Easy	96.2	100.0	100.0	50.0	100.0	96.2	96.2	92.3
	Medium	91.7	100.0	100.0	41.7	100.0	100.0	100.0	91.7
	Hard	75.0	100.0	100.0	33.3	91.7	91.7	100.0	66.7

Goal2Skill remains relatively stable on several tasks that require persistent progress tracking. On *PickXtimes*, its success rate changes from 96.2% on Easy episodes to 83.3% on Hard episodes. On *VideoUnmask*, it decreases from 92.3% to 66.7%. A similar pattern appears on *VideoPlaceOrder*, where Goal2Skill achieves 96.2%, 91.7%, and 75.0% on Easy, Medium, and Hard episodes, respectively. In comparison, the strongest non-oracle result on the Hard split of *VideoPlaceOrder* is 33.3%.

The breakdown also reveals different strengths across methods. MemER remains strong on *PickHighlight* and *MoveCube*, while FrameSamp performs well on several trajectory- and imitation-oriented tasks. Goal2Skill is strongest when increasing difficulty requires the system to preserve task state over a longer sequence, rather than simply execute a more precise motion. The difficulty results therefore reinforce the task-level pattern observed in the aggregate evaluation.

Table 9: Difficulty-stratified evaluation on the Reference and Imitation tasks of RoboMME (Dai et al., 2026). We report success rates (%) for Easy, Medium, and Hard episodes. Each task contains 50 test episodes, including 26 Easy, 12 Medium, and 12 Hard episodes. GroundSG+Oracle is included as an oracle reference.

Method	Difficulty	Reference				Imitation			
		Pick Highlight	Video Repick	Video Place Button	Video Place Order	Move Cube	Insert Peg	Pattern Lock	Route Stick
Base Policies									
$\pi_{0.5}$	Easy	15.4	0.0	34.6	26.9	26.9	3.8	3.8	7.7
	Medium	8.3	0.0	33.3	25.0	25.0	0.0	0.0	0.0
	Hard	8.3	0.0	25.0	25.0	25.0	0.0	0.0	0.0
$\pi_{0.5}$ w/ past actions	Easy	15.4	7.7	26.9	19.2	34.6	3.8	3.8	7.7
	Medium	8.3	8.3	25.0	16.7	33.3	0.0	8.3	0.0
	Hard	8.3	8.3	16.7	16.7	33.3	0.0	0.0	8.3
Memory-Augmented VLA Policies									
MME-VLA (FrameSamp)	Easy	23.1	46.2	65.4	34.6	50.0	0.0	80.8	92.3
	Medium	33.3	25.0	58.3	25.0	41.7	8.3	50.0	41.7
	Hard	16.7	8.3	25.0	16.7	50.0	0.0	0.0	33.3
MemER	Easy	96.2	26.9	23.1	34.6	73.1	0.0	26.9	11.5
	Medium	83.3	33.3	16.7	41.7	91.7	8.3	0.0	0.0
	Hard	50.0	41.7	0.0	33.3	91.7	0.0	0.0	0.0
SAM2Act+	Easy	23.1	7.7	26.9	23.1	26.9	0.0	0.0	0.0
	Medium	16.7	8.3	25.0	16.7	33.3	0.0	0.0	0.0
	Hard	8.3	0.0	16.7	16.7	33.3	0.0	0.0	0.0
Dual Systems									
GroundSG+QwenVL	Easy	38.5	30.8	50.0	42.3	57.7	11.5	15.4	7.7
	Medium	16.7	50.0	66.7	25.0	83.3	0.0	0.0	0.0
	Hard	0.0	25.0	25.0	33.3	75.0	0.0	0.0	0.0
Goal2Skill	Easy	88.5	73.1	69.2	96.2	80.8	3.8	46.2	11.5
	Medium	75.0	58.3	41.7	91.7	66.7	0.0	25.0	8.3
	Hard	50.0	33.3	25.0	75.0	41.7	0.0	8.3	0.0
Oracle Reference									
GroundSG+Oracle	Easy	96.2	100.0	100.0	100.0	92.3	11.5	100.0	76.9
	Medium	8.3	100.0	100.0	100.0	91.7	8.3	100.0	25.0
	Hard	0.0	91.7	100.0	91.7	83.3	8.3	100.0	75.0