BENCHMARKING AND ENHANCING SKILL-LEVEL MEMORY FOR PARTIALLY OBSERVABLE ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in robot learning have enabled manipulation policies to perform increasingly diverse tasks and generalize across environments. However, reliable execution often depends on hidden task states that cannot be determined from current observations alone, making interaction history essential. We introduce **HIDE**, a benchmark for evaluating manipulation memory under partial observability. HIDE comprises 15 tasks covering repetition counting, historical-state recall, and execution-progress tracking, with randomized initial configurations and decision points where similar observations require different actions depending on prior events. We further propose **SEEK**, a framework combining three complementary memory mechanisms to retain historical evidence and track execution state. Evaluations reveal substantial limitations in existing policies on HIDE, while memory augmentation improves task success in both simulation and real-world experiments. Individual mechanisms benefit some tasks but can degrade others; their combination achieves the highest average success rate on HIDE among the evaluated configurations. These findings highlight the importance of maintaining internal representations of hidden task states and matching memory design to task-specific information requirements.

1 INTRODUCTION

Visual robotic manipulation is rapidly progressing toward general-purpose control. Recent robot foundation models, including π_0 (Black et al., 2024), OpenVLA (Kim et al., 2024), and GR00T N1 (NVIDIA et al., 2025), leverage large-scale robot data to achieve diverse manipulation capabilities, while world-action models explore predictive action generation through future environment modeling (Ye et al., 2026; Wang et al., 2026). However, most manipulation policies still primarily rely on current observations or limited temporal context. In real-world execution, critical information for decision making may not be directly observable and must instead be inferred from interaction history.

Recent systems separate high-level reasoning from low-level execution through language subgoals (Shi et al., 2025b; Physical Intelligence et al., 2025) or learned semantic interfaces (Figure AI, 2025; NVIDIA et al., 2025). However, specifying a task objective does not necessarily provide the execution state required by a manipulation policy. For example, an instruction to repeat an operation twice does not indicate how many executions have already been completed. Similarly, object references and task progress may become unavailable during execution. Therefore, policies must maintain relevant internal states rather than relying only on externally specified goals.

We define these decision-relevant variables as *hidden task states*: information that cannot be determined from current visual and proprioceptive observations alone but can be inferred from interaction history. Different histories may lead to visually similar observations while requiring different actions, corresponding to observation aliasing in a partially observable Markov decision process (POMDP) (Kaelbling et al., 1998). Reliable manipulation therefore requires maintaining latent states beyond visible physical configurations, including execution-dependent variables such as completed repetitions, historical references, and procedural progress.

To study this problem, we introduce **HIDE**, a benchmark for hidden-state memory in robotic manipulation. HIDE constructs tasks based on the latent information required for correct decisions: at critical points, the current observation is insufficient, while interaction history provides the missing evidence. It contains 15 tasks organized into three categories: *repetition counting*, *historical-state recall*, and *execution-progress tracking*. The first requires tracking completed operations under visually similar repetitions; the second preserves previously observed attributes or configurations; and the third identifies the current stage of multi-step procedures under ambiguous observations. Built upon RLBench (James et al., 2020), HIDE supports randomized initial configurations, appearance variations, and automated demonstration generation, while providing structured hidden-state annotations for memory analysis.

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.

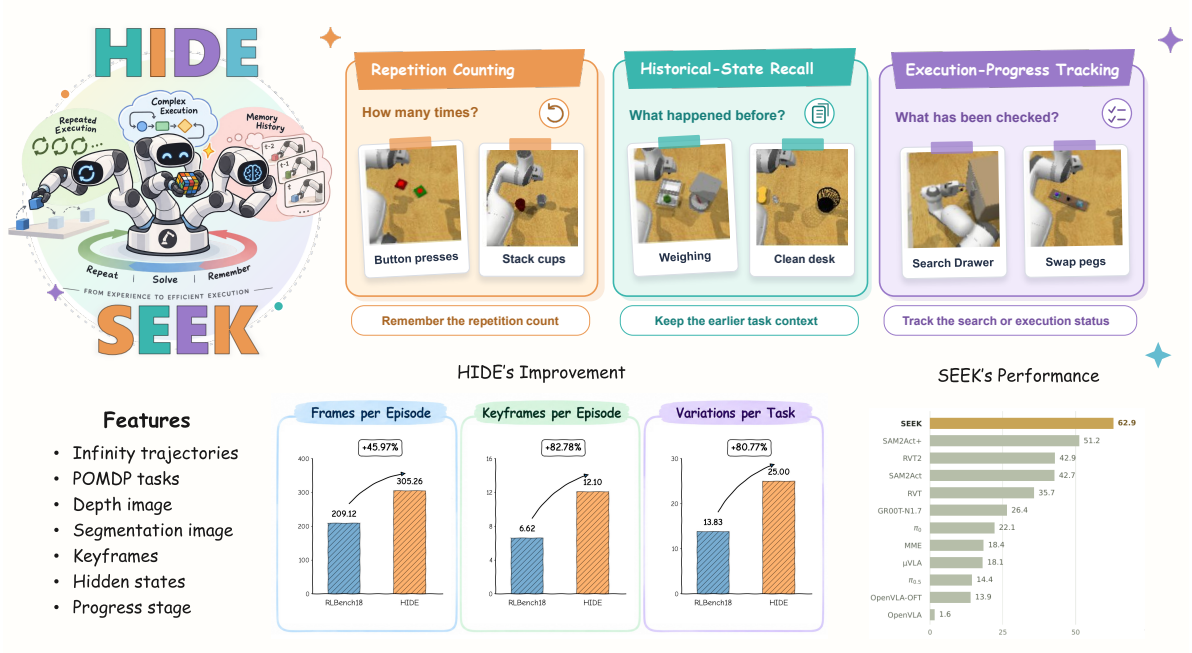


Figure 1: Illustration of our HIDE benchmark, which contains three type of skill-level memory tasks, 5 specific tasks for each type.

Existing memory-based policies retain history through recurrent states, memory banks, or retrieval mechanisms. Methods including SAM2Act+ (Fang et al., 2025), MemoryVLA (Shi et al., 2025a), and Embodied-SlotSSM (Chung et al., 2026) demonstrate the value of historical representations. However, retaining history alone does not determine whether the required hidden states can be recovered. RoboMME (Dai et al., 2026) further shows that memory effectiveness depends on task characteristics and integration strategies. This motivates studying memory according to the hidden-state requirements of manipulation tasks.

To address this challenge, we introduce **SEEK**, a memory-augmented manipulation framework that maintains three complementary forms of historical information: recent interaction context, persistent historical references, and execution progress. Evaluations on HIDE show substantial limitations of existing policies, while SEEK improves performance in both simulation and real-world experiments. Individual mechanisms exhibit distinct capability profiles, benefiting specific hidden-state requirements while potentially introducing interference in others. Combining them achieves the strongest overall performance.

Our contributions are threefold:

- **Hidden-state benchmark.** We formulate manipulation memory from decision-relevant hidden states and introduce HIDE, a 15-task benchmark covering repetition counting, historical-state recall, and execution-progress tracking.
- **Memory framework and analysis.** We introduce SEEK with three complementary memory mechanisms and systematically analyze their individual and combined effects through ablations and cross-category evaluation.
- **Performance gains and insights.** SEEK improves manipulation performance in simulation and real-world experiments, while HIDE reveals how different memory mechanisms correspond to distinct hidden-state requirements.

2 Related Work

2.1 Memory-Dependent Robotic Benchmarks

RLBench (James et al., 2020), CALVIN (Mees et al., 2022), LIBERO (Liu et al., 2023), and RoboCasa (Nasiriany et al., 2024) provide diverse manipulation tasks for evaluating generalization and long-horizon execution. However, these benchmarks primarily focus on task completion and generalization rather than explicitly isolating the historical information required for decision making. Memory-oriented benchmarks such as MemoryBench (Fang et al., 2025) and MIKASA-Robo (Cherepanov et al., 2025) further investigate memory-dependent behaviors under partial

observability. More recent benchmarks explore different aspects of manipulation memory: RoboMME (Dai et al., 2026) organizes tasks around temporal, spatial, object, and procedural memory; RoboMemArena (Lei et al., 2026) studies counting, occlusion, object transfer, and sequential execution; RMBench (Chen et al., 2026) characterizes memory complexity through decision-critical historical observations; and LIBERO-Mem (Chung et al., 2026) focuses on object-level interaction histories.

HIDE complements these benchmarks by organizing tasks according to the hidden variables required for correct manipulation decisions, including repetition count, historical scene state, and execution progress. Instead of measuring memory demand only through task length, scene complexity, or the amount of retained context, HIDE explicitly constructs decision points where the current observation is insufficient but interaction history provides the necessary evidence. Built upon RLBench, HIDE supports configurable initializations, appearance variations, and task-specific execution conditions through automated demonstration collection, enabling controlled evaluation of different hidden-state requirements and memory mechanisms.

2.2 Memory-Augmented Robotic Policies

Existing manipulation policies incorporate historical information through temporal windows, recurrent states, or explicit memory representations. HistRISE (Chen et al., 2025) models object dynamics using point trajectories, ContextVLA (Jang et al., 2025) compresses multi-frame visual context, MemoryVLA (Shi et al., 2025a) maintains perceptual and cognitive memories, and SAM2Act+ (Fang et al., 2025) integrates a visual memory bank. Retrieval-based approaches such as MemER (Sridhar et al., 2026) and PrediMem in RoboMemArena (Lei et al., 2026) select relevant historical information for long-horizon control. These methods demonstrate that historical representations can improve manipulation, while also introducing trade-offs between temporal coverage, information retention, and online memory maintenance.

However, retaining historical observations alone does not guarantee that a policy can recover the hidden state required for a specific decision. Recent studies also investigate structured memory combinations: Mem-0 in RMBench (Chen et al., 2026) combines anchor and sliding memories with subtask termination prediction, while RoboMME (Dai et al., 2026) analyzes the effect of different memory representations and integration strategies. Different from these works, we study memory according to explicit hidden-state requirements in manipulation execution. SEEK introduces complementary mechanisms for recent context, persistent historical references, and execution progress, and evaluates their individual benefits, limitations, and interactions through controlled cross-category experiments.

3 HIDE

3.1 Hidden-State-Dependent Manipulation

We formulate robotic manipulation as a partially observable decision process. At timestep t, an agent receives a task instruction g, the current observation o_t, and optionally its interaction history $h_t = (o_{1:t}, a_{1:t-1})$. We decompose the underlying task state as $s_t = (x_t, z_t)$, where x_t denotes the currently observable physical state and z_t denotes a task-relevant latent state that cannot be identified from the current observation alone.

The latent state summarizes the historical information required for optimal decision making:

$$\pi^*(a_t \mid h_t, g) = \pi^*(a_t \mid o_t, z_t, g). \tag{1}$$

We define a task as *hidden-state-dependent* if there exist two interaction histories h_t and h'_t that lead to visually equivalent current observations but require different optimal actions:

$$\begin{aligned} d(o_t, o'_t) &\le \epsilon, \\ z_t &\neq z'_t, \\ \pi^*(\cdot \mid h_t, g) &\neq \pi^*(\cdot \mid h'_t, g). \end{aligned} \tag{2}$$

Thus, long horizons or visual occlusion alone do not make a task memory-dependent; historical information must causally determine the correct subsequent behavior.

3.2 Task Categories and Design

HIDE organizes manipulation tasks into three categories according to their primary memory requirements: repetition counting, historical-state recall, and execution-progress tracking. Each category introduces decision points where the correct behavior depends on information that cannot be recovered from the current observation alone.

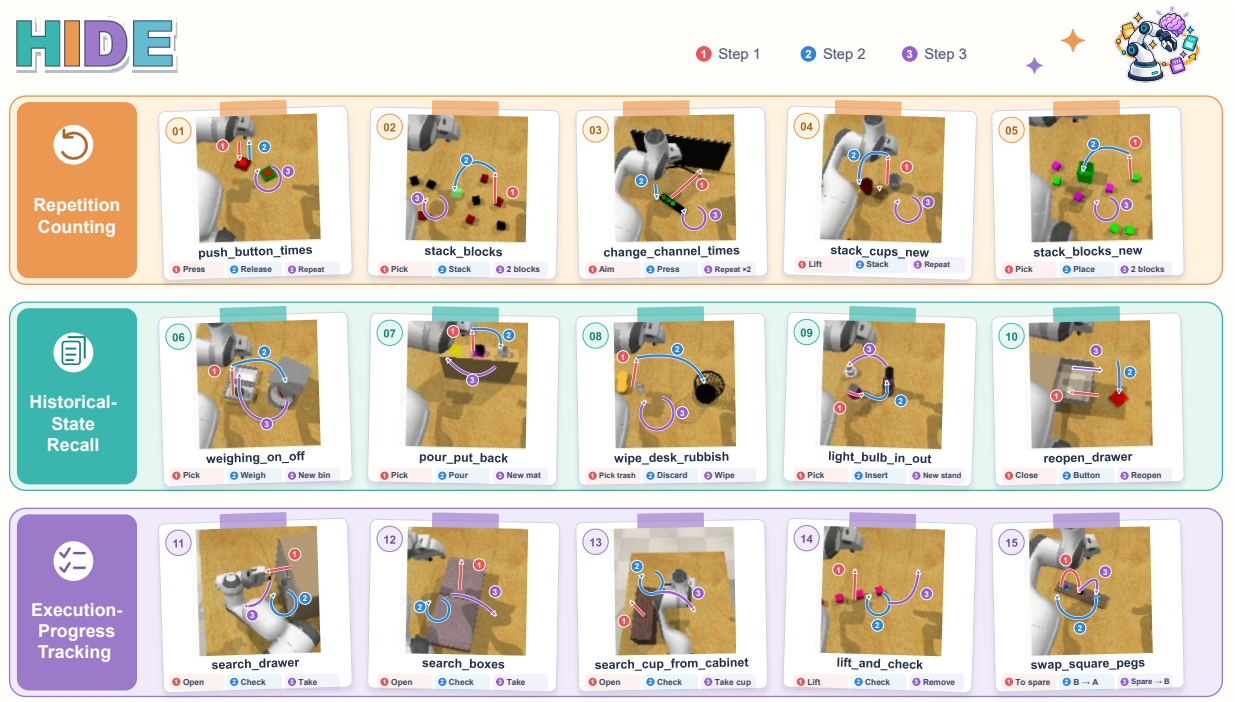


Figure 2: The sequence of steps and repeated and checking operations are indicated by red, blue, purple, and label annotations. Detailed trajectory of three kinds of tasks in HIDE benchmark. The sequence of steps and repeated and checking operations are indicated by red, blue, purple, and label annotations.

Repetition Counting (RS). The robot must repeat an operation a specified number of times. Since different repetitions can produce visually similar observations, the completed count is ambiguous from the current scene. The robot must therefore track completed repetitions and decide when to continue, stop, or switch. This category evaluates whether a policy can maintain an accurate count throughout repetitive interactions rather than merely reproduce a recurring action pattern.

Historical-State Recall (HSR). The robot must act on previously observed information, such as an object's color, identity, location, or an earlier scene configuration. At the relevant decision point, this information is no longer directly accessible due to occlusion, scene changes, or visually confusable alternatives. This category evaluates history-conditioned decisions rather than recognition from the current scene alone.

Execution-Progress Tracking (EPT). These tasks involve multiple manipulation substeps whose intermediate observations may not uniquely reveal which steps have been completed. The robot must use its execution history to determine current progress and select the appropriate next action, avoiding unnecessary repetition, skipped steps, or premature termination. The emphasis is on tracking progress under observational ambiguity rather than task length alone.

Tasks are grouped by their primary memory requirement, though individual tasks may involve additional demands. Across all categories, occlusion, visual similarity, and repeated interactions serve to create history dependence rather than as separate task categories. Detailed task specifications are provided in Figure 2 and Appendix A.1.

3.3 Benchmark Construction

We build HIDE upon RLBench (James et al., 2020), which provides a flexible simulation framework with reusable robot, object, and scene assets. Each task is extended from the standard RLBench task-generation pipeline by specifying object initialization ranges, randomized scene configurations, and a sequence of manipulation keypoints. During demonstration generation, the simulator samples object poses within predefined regions and instantiates diverse visual appearances, including colors and materials. The robot then executes a continuous trajectory by following the corresponding manipulation keypoints, enabling a large number of task variations to be generated from the same task template.

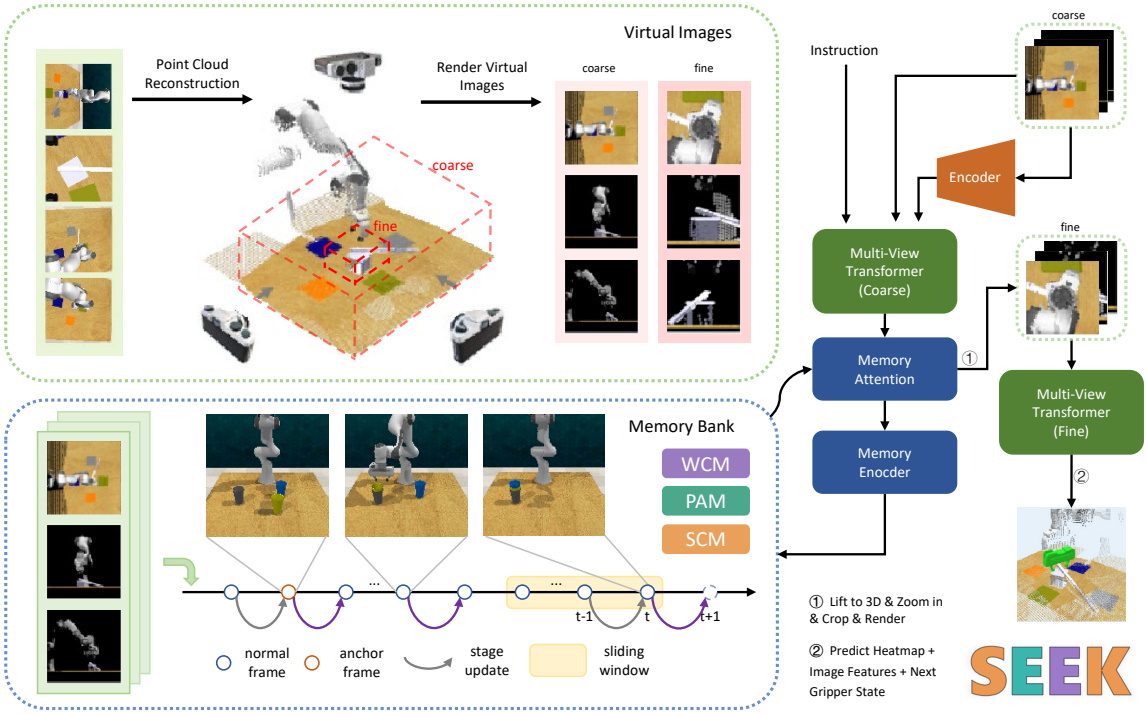


Figure 3: Overview of SEEK. Windowed Context Memory (WCM) retains recent interactions, Persistent Anchor Memory (PAM) retrieves relevant evidence beyond the recent window, and Stage-Counter Memory (SCM) represents execution progress. The three components jointly condition the coarse-to-fine manipulation policy.

Compared with conventional RLBench tasks, HIDE introduces longer and more memory-dependent manipulation sequences. We modify existing assets and interaction patterns to create repeated operations, historically dependent object choices, and multi-stage procedures whose correct execution cannot always be determined from the current observation alone. Task difficulty can be systematically varied through factors such as the number of repetitions, the number and similarity of distractor objects, the duration between informative observations and subsequent decisions, and the number of manipulation stages.

Demonstrations are collected using the standard RLBench scripted expert pipeline. After data collection, we automatically segment each trajectory into task stages according to the predefined manipulation keypoints and task-specific success conditions. These stage annotations provide execution-progress information for both training and evaluation, while avoiding additional manual annotation.

4 SEEK

Under partial observability, a manipulation policy must distinguish states that look similar but require different actions because of their history. Recent interactions reveal local changes, earlier observations may contain now-hidden evidence, and execution progress determines whether an action should be repeated or the policy should move on. SEEK organizes these dependencies into three complementary components: **Windowed Context Memory (WCM)**, **Persistent Anchor Memory (PAM)**, and **Stage-Counter Memory (SCM)**. Together, they provide recent context, a selectively retrieved long-range reference, and an explicit progress state within a compact attention input.

4.1 Hidden-State Memory Modeling

Windowed Context Memory. WCM retains the latest K interaction memories in temporal order. Each new entry replaces the oldest once the window is full, preserving recent object changes and manipulation outcomes. This local context helps the policy track what has just happened, but cannot retain evidence indefinitely as an episode unfolds.

In sequential exploration, it can indicate which locations were recently inspected and how their contents changed, providing context for the next interaction without requiring the entire history at every decision.

Persistent Anchor Memory. PAM preserves access to evidence after it leaves the recent window. For example, an early observation may reveal an object's identity or location before subsequent interactions occlude it. Rather than fixing the first frame as a reference, PAM selects a historical entry according to the current observation. Let $\mathcal{I}_t^W$ denote the indices in WCM. The eligible archive is $\mathcal{A}_t = \{i : 0 \le i < t,\ i \notin \mathcal{I}_t^W\}$, and retrieval is

$$a_t = \underset{i \in \mathcal{A}_t}{\operatorname{argmax}} \cos(\mathbf{k}_t, \mathbf{k}_i), \tag{3}$$

where $\mathbf{k}_t$ and $\mathbf{k}_i$ are pooled visual descriptors computed before memory fusion. Each descriptor is stored alongside its interaction memory, separating the compact key used for selection from the spatial features used for action prediction. Temporal exclusion first removes entries already covered by WCM; content similarity then selects a reference from the remaining history. The selected entry retains its own encoded interaction content, so PAM contributes information from outside the local window rather than another copy of a recent feature. Selection is recomputed at each decision, allowing the reference to change with the observed state. Persistence refers to the availability of earlier evidence, not to a permanently fixed anchor. When no older entry is available, the anchor is omitted. An entry leaving WCM remains in the episode archive and becomes eligible for PAM retrieval. This transfer separates recency from continued availability: intervening actions can displace an observation from local context without making its evidence inaccessible to later decisions.

Stage-Counter Memory. SCM represents progress using a discrete counter and a learned stage embedding. The counter starts at zero and advances when the policy predicts a stage boundary. Whereas WCM and PAM describe previous interactions, SCM indicates how far the execution has progressed. This distinction is useful when repeated actions return the scene to a similar appearance: visual similarity alone may not distinguish an intermediate repetition from the final one. The counter changes at predicted interaction boundaries rather than at every control step, allowing progress to remain stable while an action unfolds.

4.2 Memory Encoding and Retrieval

Multi-camera RGB-D observations are rendered into three virtual views. For each view v, a shared memory encoder combines visual features $\mathbf{X}_t^v$ with the predicted coarse translation heatmap $\mathbf{H}_t^v$ to produce interaction memory $\mathbf{m}_t^v$. This representation associates scene content with the predicted manipulation target. It is written after action prediction and becomes available only to subsequent decisions. WCM and PAM select from these same encoded entries; SCM supplies a separate learned representation $\mathbf{S}_t$. Retaining spatial feature maps also preserves where interaction evidence appeared in each view. Retrieval therefore supplies localized information for manipulation, rather than only a global summary of the past.

The attention input combines the three components:

$$\mathcal{M}_t^v = [\mathcal{W}_t^v; \mathbf{m}_{a_t}^v; \mathbf{S}_t], \qquad \widetilde{\mathbf{X}}_t^v = \mathrm{MemoryAttn}(\mathbf{X}_t^v, \mathcal{M}_t^v). \tag{4}$$

The anchor term is absent when the eligible archive is empty. Spatial position encodings preserve within-view structure, while temporal slot encodings distinguish window entries and the anchor; SCM has its own position embedding. Following SAM 2 (Ravi et al., 2024), current visual features query the memory through attention, producing features conditioned on past interactions and execution progress. Joint attention lets the current observation determine which stored cues are relevant to the decision. The components therefore need not be assigned to separate task-specific policies. The read budget is at most K recent entries, one historical anchor, and one stage representation per view. The archive can grow over the episode, but the number of entries passed to attention remains bounded.

4.3 Memory-Augmented Manipulation Policy

SEEK builds on a language-conditioned, coarse-to-fine multi-view policy. The coarse branch uses memory-conditioned features to predict translation heatmaps over the workspace. The resulting target defines a local region that is re-rendered for fine action prediction, including end-effector position, rotation, and gripper state. Memory thus influences target selection before local refinement; the fine branch does not independently retrieve the episode archive. For visually similar candidate objects, historical context can inform which target the coarse branch selects, while the fine branch uses current local geometry to refine the selected interaction. After prediction, the interaction memory is stored for future WCM and PAM reads, and the stage-transition prediction updates SCM. All memories and the counter are reset between episodes.

Table 1: **Performance on HIDE.** Success rates (%) on individual tasks and their category averages. The benchmark contains three categories: repetition counting, historical-state recall, and execution-progress tracking. The best and second-best results in each column are **bold** and underlined, respectively, including ties.

		Repetition Counting (RC)						Historical-State Recall (HSR)						Execution-Progress Tracking (EPT)					
Method	**Avg.**	Push Button	Stack Blocks	Change Channel	Stack Cups	Stack Blocks-N	**Avg.**	Weigh On/Off	Pour Back	Wipe Desk	Light Bulb	Reopen Drawer	**Avg.**	Search Drawer	Search Boxes	Search Cabinet	Lift & Check	Swap Pegs	**Avg.**
OpenVLA (Kim et al., 2024)	1.6	8	0	4	0	4	3.2	8	0	0	0	0	1.6	0	0	0	0	0	0.0
OpenVLA-OFT Kim et al. (2025)	13.9	12	0	28	0	0	8.0	36	0	68	0	32	27.2	0	4	0	28	0	6.4
π_0 (Black et al., 2024)	22.1	40	0	12	0	0	10.4	40	24	92	0	24	36.0	12	28	36	24	0	20.0
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	14.4	16	0	4	4	4	5.6	20	8	76	0	20	24.8	4	8	20	32	0	12.8
GR00T-N1.7 (NVIDIA et al., 2025)	26.4	12	0	12	4	4	6.4	96	0	84	0	68	49.6	28	16	40	24	8	23.2
RVT (Goyal et al., 2023)	35.7	68	12	20	40	48	37.6	28	24	80	0	60	38.4	40	12	44	60	0	31.2
RVT2 (Goyal et al., 2024)	42.9	12	**56**	24	**48**	76	43.2	36	**36**	84	8	48	42.4	48	16	56	88	8	43.2
SAM2Act (Fang et al., 2025)	42.7	16	**56**	12	**48**	76	41.6	8	32	**100**	4	52	39.2	48	24	64	84	16	47.2
Memory-based Methods																			
μVLA (Cherepanov et al., 2026)	18.1	60	0	4	0	0	12.8	48	8	60	0	28	28.8	0	0	40	20	4	12.8
MME (Dai et al., 2026)	18.4	0	0	4	0	8	2.4	**100**	4	56	0	16	35.2	16	16	28	28	0	17.6
SAM2Act+ (Fang et al., 2025)	51.2	32	52	32	**48**	68	46.4	8	28	96	28	72	46.4	52	8	**72**	**100**	72	60.8
SEEK	**62.9**	**92**	48	**52**	36	**80**	**61.6**	24	32	**100**	**40**	**100**	**59.2**	**60**	**32**	68	88	**92**	**68.0**

4.4 Training Objective

SEEK is trained by behavior cloning on expert demonstrations. The action loss supervises coarse and fine translation, rotation, gripper state, and collision-related predictions. A stage-boundary loss trains the progress predictor, giving $\mathcal{L} = \mathcal{L}_{\text{act}} + \lambda_{\text{stage}}\mathcal{L}_{\text{stage}}$. Training sequences are processed in temporal order with annotated stage states; inference updates memory and the counter online using the policy's predictions. Only preceding observations are eligible for retrieval.

5 Experiments

We organize our experiments around five research questions:

- **Q1:** What limitations do existing manipulation policies exhibit across the hidden-state requirements of HIDE?
- **Q2:** How much does our method improve performance, and are the gains consistent across task categories?
- **Q3:** How do memory content, length, and component design affect overall and category-specific performance?
- **Q4:** Does the method remain effective on standard tasks, under perturbations, and in real-world execution?

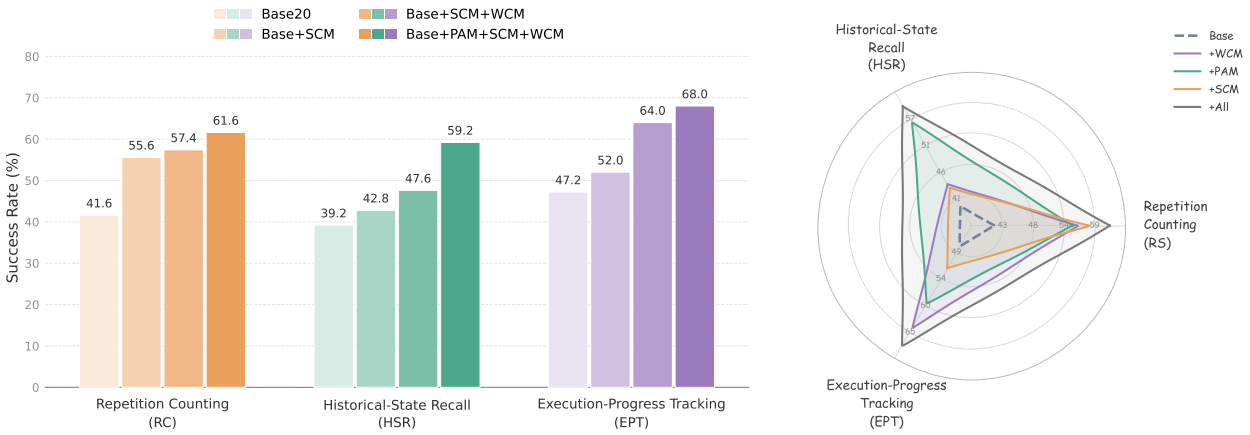
5.1 Experimental Setup

We primarily evaluate SEEK on HIDE, which comprises 15 tasks across Repetition Counting (RC), Historical-State Recall (HSR), and Execution-Progress Tracking (EPT). We additionally evaluate standard manipulation on the 18-task RLBench suite, perturbation robustness on `The Colosseum`, and physical execution on four real-robot tasks. Baselines cover vision-language-action models, 3D manipulation policies, and memory-based methods, with SAM2Act as the backbone baseline.

Following RLBench's demonstration-generation and data-splitting protocol, we use 100 training demonstrations and 25 held-out test episodes per HIDE task. A single policy is jointly trained across all 15 tasks. We report per-task success rates, category averages, and the overall average, and ablate memory content, length, and components. Simulation settings, baseline configurations, training procedures, and real-robot protocols are detailed in Appendix B.

5.2 Q1: Do Existing Policies Exhibit a Hidden-State Memory Gap?

Table 1 shows that HIDE remains challenging even for memory-based policies. SAM2Act+ improves average success from 42.7% for SAM2Act (Fang et al., 2025) to 51.2%, while μVLA (Cherepanov et al., 2026) improves from 13.9% for OpenVLA-OFT (Kim et al., 2025) to 18.1%. Both improve across all three categories, but substantial gaps remain. SAM2Act+ reaches 60.8% on EPT versus 46.4% on both RC and HSR; MME (Dai et al., 2026) achieves 18.4% overall, with 35.2% on HSR but only 2.4% on RC. The strongest baseline varies by category: SAM2Act+ leads RC and EPT, whereas GR00T-N1.7 (NVIDIA et al., 2025) leads HSR at 49.6%. These uneven results suggest that adding memory does not by itself resolve the distinct demands of hidden-state manipulation, motivating category-level evaluation rather than reliance on aggregate performance alone.



(a) Progressive component integration.

(b) Individual and combined components.

Figure 4: **Memory component analysis.** Category success rates (%) for progressive integration (a) and individual components versus their combination (b). The components show complementary strengths across hidden-state requirements. Radar axes use category-specific scales.

Table 2: **Memory content and length ablation.** Success rates (%). Content variants share the same encoding dimensionality and memory architecture.

	Variant	Avg.	RC	HSR	EPT
Memory Content	Proprioception	42.5	31.2	41.6	54.6
	Visual Features	**62.9**	**61.6**	**59.2**	**68.0**
Memory Length	0	46.7	46.4	42.0	51.6
	2	53.7	44.2	51.8	65.0
	4	57.2	48.6	55.8	67.2
	6	**62.9**	**61.6**	**59.2**	**68.0**

Table 3: **RLBench and `The Colosseum` results.** Success rates (%); parentheses give relative drops from Clean. ** denotes our reproduction. Details: Appendix C.

Method	RLBench	`The Colosseum`	
	Avg. Success ↑	Clean ↑	Average ↑
RVT	62.9	43.6	36.3 (↓16.7%)
RVT-2	81.4	67.8	59.5 (↓12.3%)
SAM2Act**	84.1	<u>68.4</u>	61.5 (↓10.1%)
SEEK	**84.7**	**68.6**	**61.9** (↓9.8%)

5.3 Q2: Can Our Method Improve Hidden-State Manipulation?

SEEK achieves 62.9% average success on HIDE (Table 1), exceeding the strongest baseline, SAM2Act+, by 11.7 percentage points. It leads all three categories at 61.6% on RC, 59.2% on HSR, and 68.0% on EPT, improving over the strongest baseline in each category by 15.2, 9.6, and 7.2 percentage points, respectively. The gains therefore span repetition counting, historical-state recall, and execution-progress tracking rather than a single task family. These improvements over existing memory-based policies support SEEK's memory design for using task-relevant history under visual ambiguity.

5.4 Q3: How Do Different Memory Components Contribute?

We examine what information to store, how much history to retain, and how individual memory components contribute to performance.

What to store? With encoding dimensionality and memory architecture fixed, storing visual features raises average success from 42.5% to 62.9% (Table 2). Gains span all three categories and are largest on RC (+30.4 percentage points), followed by HSR (+17.6) and EPT (+13.4). This suggests that visual histories retain more task-relevant information than proprioceptive histories for resolving hidden-state ambiguity in this setting.

How much history to retain? Average success increases with memory length, rising from 46.7% at length 0 to 62.9% at length 6 (Table 2). Length 6 achieves the best overall performance and the highest scores on all three categories. Notably, RC initially drops from 46.4% to 44.2% at length 2, suggesting that a short memory horizon is

Table 4: **Real-world results.** Success rates (%); Avg. is the unweighted mean across tasks.

Method	Avg.	Push button × N	Stack N cups (M total)	Clean desk	Search chip
$\pi_{0.5}$	13	0	20	0	32
SAM2Act+	47	32	48	44	64
SEEK	**89**	**76**	**100**	**80**	**100**

still insufficient for repetition-based tasks. With longer context, RC improves substantially to 61.6%. EPT shows a more gradual saturation, increasing from 67.2% at length 4 to 68.0% at length 6. Overall, these results indicate that sufficiently long temporal context is important for reliably resolving hidden states, especially for repetition counting.

Memory components. Progressively adding SCM, WCM, and PAM improves success across all three categories (Figure 4a). The largest gain at each step occurs on RC with SCM (+14.0 percentage points), EPT with WCM (+12.0), and HSR with PAM (+11.6). The single-component comparison (Figure 4b) also shows distinct strengths: SCM leads the individual variants on RC, PAM on HSR, and WCM on EPT, while their combination leads all three categories. This pattern is consistent with complementary roles in all category of tasks.

5.5 Q4: Does the Method Remain Effective Beyond HIDE?

We further evaluate SEEK on standard manipulation tasks, under environmental perturbations, and in real-world execution to assess whether its memory-based design remains effective beyond HIDE.

Standard manipulation. SEEK achieves 84.7% average success on RLBench (Table 3), slightly improving over our SAM2Act (Fang et al., 2025) reproduction at 84.1%, and outperforming RVT-2 and RVT at 81.4% and 62.9%, respectively. This indicates that introducing memory preserves strong performance on standard manipulation tasks rather than trading it off for hidden-state reasoning.

Perturbation robustness. On `The Colosseum`, SEEK achieves 68.6% on Clean and 61.9% averaged over perturbations, compared with 68.4% and 61.5% for our SAM2Act reproduction (Table 3). The relative drop from Clean is also slightly smaller (9.8% vs. 10.1%). Together, these results show that SEEK maintains competitive robustness under environmental perturbations while retaining its gains on hidden-state tasks.

Real-world execution. We evaluate four real-world tasks: pressing a button a specified number of times, stacking a specified number of cups on the middle cup, cleaning a desk, and lifting a block to search for a white piece. Each task is instantiated with multiple randomized initial configurations, differing in object counts, positions, and scene layouts, so that the same task can be presented under diverse visual conditions. Task details are provided in Appendix A.2. Each task contains visually similar observations associated with different latent states and required actions, making task history necessary for successful execution. SEEK outperforms both baselines on all four tasks (Table 4), averaging 89% success versus 47% for SAM2Act+ (Fang et al., 2025) and 13% for $\pi_{0.5}$ (Physical Intelligence et al., 2025). These results support using memory to retain the initial state, task progress, and recent scene context, helping the policy infer task-relevant latent states under visual ambiguity.

6 Conclusion

We introduced HIDE, a benchmark for evaluating hidden-state reasoning in robotic manipulation, and SEEK, a memory framework that maintains task-relevant information over interaction history. HIDE decomposes hidden-state manipulation into three complementary requirements—repetition counting, historical-state recall, and execution-progress tracking—and exposes clear limitations in existing policies, including methods equipped with memory. SEEK consistently improves performance across these categories, while retaining strong performance on standard manipulation tasks, under environmental perturbations, and in real-world execution. Our ablations further show that effective memory depends not only on retaining history, but also on what information is stored, how long it is preserved, and how different memory mechanisms contribute to distinct hidden-state requirements. Together, these results highlight the importance of explicitly modeling latent task state for long-horizon robotic decision making. HIDE currently focuses on controlled and interpretable forms of hidden state; extending it to richer real-world settings, broader sources of partial observability, and more general memory mechanisms remains an important direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools for (i) literature search and summarization of related work, (ii) scripting and queueing the batch execution of experiments, and (iii) drafting the initial skeleton of parts of the manuscript and figure layouts. We have not used generative AI tools for research ideation, experimental design, code implementation and debugging, data construction and analysis, or the final writing and revision of the manuscript, and AI-assisted data collection and AI-assisted peer review are not applicable to this work. Additionally, we used generative AI tools for language polishing and LaTeX formatting. We have reviewed all AI-assisted work: all references suggested by AI-assisted literature summarization were manually verified against the original papers; experiment execution scripts were inspected by the authors and all reported results were independently checked; all AI-drafted text and figures were reviewed, rewritten, and finalized by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects and required no IRB approval. All simulation experiments are conducted on publicly available, open-source benchmarks (RLBench (James et al., 2020) and The Colosseum (Pumacay et al., 2024)), used in accordance with their licenses; these benchmarks consist of synthetic scenes and contain no personally identifiable information. Real-robot evaluations are conducted in controlled indoor laboratory environments under human supervision and standard safety protocols, and no personal data of bystanders is collected or stored. We do not foresee direct societal harms, dual-use risks, or fairness concerns beyond those common to robotic manipulation research in general.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Jingjing Chen, Hongjie Fang, Chenxi Wang, Shiquan Wang, and Cewu Lu. History-aware visuomotor policy learning via point tracking. *arXiv preprint arXiv:2509.17141*, 2025. Accepted to ICRA 2026.

Shizhe Chen, Ricardo Garcia, Cordelia Schmid, and Ivan Laptev. PolarNet: 3D point clouds for language-guided robotic manipulation. *arXiv preprint arXiv:2309.15596*, 2023.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, Hongcheng Wang, Junjie Wang, Tianhang Yang, Renjing Xu, Ruihai Wu, Yao Mu, Yaodong Yang, Hao Dong, and Ping Luo. RMBench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv preprint arXiv:2603.01229*, 2026.

Egor Cherepanov, Nikita Kachaev, Alexey K. Kovalev, and Aleksandr I. Panov. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning. *arXiv preprint arXiv:2502.10550*, 2025.

Egor Cherepanov, Nikita Kachaev, Daniil Zelezetsky, Aydar Bulatov, Artem Pshenitsyn, Yuri Kuratov, Alexey Skrynnik, Aleksandr I Panov, and Alexey K Kovalev. μvla: On recurrent memory for partially observable manipulation in vla models. *arXiv preprint arXiv:2606.12497*, 2026.

Nhat Chung, Taisei Hanyu, Toan Nguyen, Huy Le, Frederick Bumgarner, Duy Minh Ho Nguyen, Khoa Vo, Kashu Yamazaki, Chase Rainwater, Tung Kieu, Anh Nguyen, and Ngan Le. Rethinking progression of memory state in robotic manipulation: An object-centric perspective. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(5):3407–3415, 2026. doi: 10.1609/aaai.v40i5.37337.

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.

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. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 15925–15942. PMLR, 2025. URL https://proceedings.mlr.press/v267/fang25c.html.

Figure AI. Helix: A vision-language-action model for generalist humanoid control. Technical blog, February 2025. URL https://www.figure.ai/news/helix.

Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3D: 3D feature field transformers for multi-task robotic manipulation. *arXiv preprint arXiv:2306.17817*, 2023.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. RVT: Robotic view transformer for 3D object manipulation. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 694–710. PMLR, 2023. URL https://proceedings.mlr.press/v229/goyal23a.html.

Ankit Goyal, Valts Blukis, Jie Xu, Yijie Guo, Yu-Wei Chao, and Dieter Fox. RVT-2: Learning precise manipulation from few demonstrations. *arXiv preprint arXiv:2406.08545*, 2024.

Pierre-Louis Guhur, Shizhe Chen, Ricardo Garcia Pinel, Makarand Tapaswi, Ivan Laptev, and Cordelia Schmid. Instruction-driven history-aware policies for robotic manipulations. *arXiv preprint arXiv:2209.04899*, 2022.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020. doi: 10.1109/LRA.2020.2974707.

Stephen James, Kentaro Wada, Tristan Laidlow, and Andrew J. Davison. Coarse-to-fine Q-attention: Efficient learning for visual robotic manipulation via discretisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13729–13738, 2022. URL https://arxiv.org/abs/2106.12534.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-Z: Zero-shot task generalization with robotic imitation learning. *arXiv preprint arXiv:2202.02005*, 2022.

Huiwon Jang, Sihyun Yu, Heeseung Kwon, Hojin Jeon, Younggyo Seo, and Jinwoo Shin. ContextVLA: Vision-language-action model with amortized multi-frame context. *arXiv preprint arXiv:2510.04246*, 2025.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X.

Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3D Diffuser Actor: Policy diffusion with 3D scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Huashuo Lei, Wenxuan Song, Huarui Zhang, Jieyuan Pei, Jiayi Chen, Haodong Yan, Han Zhao, Pengxiang Ding, Zhipeng Zhang, Lida Huang, Donglin Wang, Yan Wang, and Haoang Li. RoboMemArena: A comprehensive and challenging robotic memory benchmark. *arXiv preprint arXiv:2605.10921*, 2026.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html`.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022. doi: 10.1109/LRA.2022.3180108.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of household tasks for generalist robots. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.050.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. The Colosseum: A benchmark for evaluating generalization for robotic manipulation. *arXiv preprint arXiv:2402.08191*, 2024.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. SAM 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 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*, 2025a.

Lucy Xiaoyang Shi, Brian Ichter, Michael Robert Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi Robot: Open-ended instruction following with hierarchical vision-language-action models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 54919–54933. PMLR, 2025b. URL https://proceedings.mlr.press/v267/shi25d.html.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-Actor: A multi-task transformer for robotic manipulation. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 785–799. PMLR, 2023. URL https://proceedings.mlr.press/v205/shridhar23a.html.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. MemER: Scaling up memory for robot control via experience retrieval. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.20328.

Siyin Wang, Junhao Shi, Zhaoyang Fu, Xinzhe He, Feihong Liu, Chenchen Yang, Yikang Zhou, Zhaoye Fei, Jingjing Gong, Jinlan Fu, Mike Zheng Shou, Xuanjing Huang, Xipeng Qiu, and Yu-Gang Jiang. World action models: The next frontier in embodied AI. *arXiv preprint arXiv:2605.12090*, 2026.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, Min Cao, Peng Li, Qiuping Deng, Wenjun Mei, Xiaofeng Wang, Xinze Chen, Xinyu Zhou, Yang Wang, Yifan Chang, Yifan Li, Yukun Zhou, Yun Ye, Zhichao Liu, and Zheng Zhu. GigaWorld-Policy: An efficient action-centered world–action model. *arXiv preprint arXiv:2603.17240*, 2026.

Junjie Zhang, Chenjia Bai, Haoran He, Wenke Xia, Zhigang Wang, Bin Zhao, Xiu Li, and Xuelong Li. SAM-E: Leveraging visual foundation model with sequence imitation for embodied manipulation. *arXiv preprint arXiv:2405.19586*, 2024.

APPENDIX

A TASK DETAILS AND DEMONSTRATION

A.1 SIMULATION TASKS

HIDE contains 15 tasks organized into three categories with five tasks in each: repetition counting, historical-state recall, and execution-progress tracking. Brief statistics are displayed in Table 5.

Table 5: **Task Statistics of HIDE.** We report the language template, average number of frames, average number of extracted keyframes, number of task variations, and variation type. Square brackets denote variable placeholders in the language templates, with `[N]` indicating the requested count.

Task name	Language Template	Frames	Keyframes	# of Var.	Variation Type
push_button_times	"push the `[color]` button `[N]` times"	144.3	7.8	50	repetition count × button color
stack_blocks	"stack `[N]` `[color]` blocks"	372.8	16.1	60	stack size × color
change_channel_times	"turn the channel `[direction]` `[N]` times"	244.5	12.2	6	direction × repetition count
stack_cups_new	"stack `[N]` cup(s) on top of the `[color]` cup"	208.8	8.4	40	base color × cup count
stack_blocks_new	"put `[N]` `[color]` blocks into the rectangular container"	283.5	11.8	60	placement count × color
weighing_on_off	"weigh the pepper and put it in another container"	199.0	11.0	2	starting container
pour_put_back	"pour liquid from the `[color]` cup to the `[color]` cup, then place it on the other coaster"	330.0	7.5	40	coaster side × cup-color pair
wipe_desk_rubbish	"wipe dirt off the desk"	272.9	9.0	1	randomized layout
light_bulb_in_out	"screw in the light bulb and move it to the other holder"	335.9	10.8	20	holder color (implicit)
reopen_drawer	"close the opened drawer, push the button, and open the previous drawer again"	320.2	10.0	3	drawer level (bottom/middle/top)
search_drawer	"take item out of the drawer"	393.8	15.7	3	hidden-item drawer level
search_boxes	"take shoes out of box"	629.1	19.0	2	search path / holder config
search_cup_from_cabinet	"take out a cup from the cabinet"	245.5	9.9	2	cabinet side
lift_and_check	"lift blocks one by one to find the white chip underneath and remove it"	277.3	14.3	80	block color × chip position
swap_square_pegs	"swap the positions of the two rings"	321.3	18.0	6	peg/ring permutation

I. Repetition Counting
Track completed repetitions to know when to stop.

(a) Push button times

Task Description: The robot is instructed to identify the target colored button and press it the requested number of times. Depending on the variation, the button must be pressed once, twice, three times, four times, or five times.

Success Metric: The task is considered successful once the target button has been pressed exactly the requested number of times and the robot arm has withdrawn upward.

Objects: Three colored push buttons and their button mechanisms.

Variation Number: 50 **Keyframes:** 4, 6, 8, 10, or 12

Language Instructions: *"Push the* `<color>` *button* `<count>`*."*

(b) Stack blocks

Task Description: The robot is instructed to identify blocks of the specified color and stack the requested number of them vertically. The requested stack contains two, three, or four blocks.

Success Metric: The task is considered successful once the requested number of target blocks is detected in the stacking region and the robot is no longer holding an object.

Objects: Four target blocks, four distractor blocks, and a stacking region.

Variation Number: 60 **Keyframes:** 11, 17, 22, 23, or 24

Language Instructions: *"Stack* `<2-4>` `<color>` *blocks."*

(c) Change channel times

Task Description: The robot is instructed to pick up the television remote, point it toward the television, and press either the plus or minus channel button the requested number of times.

Success Metric: The task is considered successful once the correct channel button has been pressed exactly one, two, or three times, the remote reaches the required final configuration, and the robot arm has withdrawn upward.

Objects: A television remote, plus and minus buttons, and a television-facing target region.

Variation Number: 6 **Keyframes:** 10, 12, or 14

Language Instructions: *"Turn the channel* `<up or minus>` `<once, twice, or three times>`*."*

(d) Stack cups new

Task Description: The robot is instructed to keep the specified colored cup as the base and place either one or two of the remaining cups on top of it.

Success Metric: The task is considered successful once the requested cup or cups are detected in the stack, the gripper is empty, and the robot arm has withdrawn upward.

Objects: Three colored cups and a cup-stacking success region.

Variation Number: 40 **Keyframes:** 6 or 11

Language Instructions: *"Stack* `<one or two>` *cup(s) on top of the* `<color>` *cup."*

(e) Stack blocks new

Task Description: The robot is instructed to identify blocks of the specified color and place the requested number of them into the open rectangular container at the center of the workspace.

Success Metric: The task is considered successful once the requested number of target blocks is detected inside the container, the gripper is empty, and the robot arm has withdrawn upward.

Objects: Four target blocks, four distractor blocks, and a central open-top container.

Variation Number: 60 **Keyframes:** 6, 11, 16, or 17

Language Instructions: *"Put* `<1–3>` `<color>` *blocks into the rectangular container in the center."*

II. Historical-State Recall

Act on past information no longer visible in the scene.

(f) Weighing on off

Task Description: The robot is instructed to pick up the pepper from one container, place it on the scale long enough to obtain a weight reading, and then move it into the other container.

Success Metric: The task is considered successful once the scale has registered the pepper while the gripper is released and the pepper is subsequently detected in the destination container.

Objects: A pepper, a weighing scale, and two containers.

Variation Number: 2 **Keyframes:** 11

Language Instructions: *"Weigh the pepper and put it in another container."*

(g) Pour put back

Task Description: The robot is instructed to pick up the source cup, pour its liquid into the target cup, and then place the source cup on the other coaster.

Success Metric: The task is considered successful once all liquid particles are detected in the target cup and the source cup is detected on the opposite coaster.

Objects: A source cup, a target cup, liquid particles, and two coasters.

Variation Number: 40 **Keyframes:** 6, 7, or 8

Language Instructions: *"Pour liquid from the* `<source color>` *cup to the* `<target color>` *cup, then place the source cup on the other coaster."*

(h) Wipe desk rubbish

Task Description: The robot is instructed to dispose of the rubbish in the bin and then use the sponge to wipe all visible dirt from the desk.

Success Metric: The task is considered successful once the rubbish is detected in the bin and all generated dirt spots have been removed from the desk.

Objects: A piece of rubbish, a rubbish bin, a sponge, and multiple dirt spots.

Variation Number: 1 **Keyframes:** 8 or 9

Language Instructions: *"Wipe dirt off the desk."*

(i) Light bulb in out

Task Description: The robot is instructed to pick up the light bulb from its initial holder, screw it into the lamp until it lights, remove it, and place it into the other holder.

Success Metric: The task is considered successful once the bulb has been detected in the lamp, has lit up, is subsequently detected in the destination holder, and has been released by the gripper.

Objects: A light bulb, a lamp socket, and two bulb holders.

Variation Number: 20 **Keyframes:** 10 or 11

Language Instructions: *"Screw in the light bulb and move it to the other holder."*

(j) Reopen drawer

Task Description: The robot is instructed to remember the drawer slot that was initially open, close it, press the button on the table, and then find and reopen the previously opened drawer.

Success Metric: The task is considered successful once the initially opened drawer has been closed, the button has been pressed, and the same drawer has been reopened.

Objects: A three-level drawer cabinet and a push button.

Variation Number: 3 **Keyframes:** 10

Language Instructions: *"Close the drawer, then reopen the previously opened drawer while pushing the button in between."*

III. Execution-Progress Tracking

Use history to identify which substeps are done and what comes next.

(k) Search drawer

Task Description: The robot is instructed to search the drawers, locate the hidden item, remove it from the correct drawer, and place it in the target region.

Success Metric: The task is considered successful once the hidden item is detected in the designated success region outside the drawer.

Objects: A three-level drawer cabinet, a hidden item, and a success region.

Variation Number: 3 **Keyframes:** 9, 16, or 23

Language Instructions: *"Take the item out of the drawer."*

(l) Search boxes

Task Description: The robot is instructed to open the shoe box, search for both shoes, remove them, and place them on the table.

Success Metric: The task is considered successful once both shoes are detected outside the box in the target region and the robot is no longer holding either shoe.

Objects: A shoe box and lid, two shoes, and a target region on the table.

Variation Number: 2 **Keyframes:** 15, 16, 23, or 24

Language Instructions: *"Take the shoes out of the box."*

(m) Search cup from cabinet

Task Description: The robot is instructed to open the appropriate side of the cabinet, locate the cup, remove it from the cabinet, and release it outside.

Success Metric: The task is considered successful once the cup is no longer detected inside the cabinet and the robot is no longer holding it.

Objects: A two-sided cabinet, sliding cabinet doors, and a cup.

Variation Number: 2 **Keyframes:** 7, 8, 13, or 14

Language Instructions: *"Take out a cup from the cabinet."*

(n) Lift and check

Task Description: The robot is instructed to lift the blocks one by one, inspect the space underneath each block, locate the hidden white chip, and move the block that covers it away.

Success Metric: The task is considered successful once the block covering the hidden white chip has been identified and moved into the designated success region.

Objects: Four colored blocks, a hidden white chip or marker, and a success region.

Variation Number: 80 **Keyframes:** 6, 11, 16, or 21

Language Instructions: *"Lift blocks one by one to find the white chip underneath and remove it."*

(o) Swap square pegs

Task Description: The robot is instructed to pick up the two square rings and exchange their positions between the pegs.

Success Metric: The task is considered successful once each square ring is detected on the other ring's original target peg.

Objects: Two square rings and three pegs.

Variation Number: 6 **Keyframes:** 18

Language Instructions: *"Swap the positions of the two rings."*

A.2 REAL-WORLD TASKS

We have designed four tasks that need completed using hidden states, including multiple button presses, stacking paper cups, cleaning tables, inspecting chip, each of which is highly confusing and challenging.

Table 6: Language templates and demonstration statistics for the four tasks.

Task name	Language Template	Frames	Keyframes	# of Var.	Variation Type
push_button_times	"push the `[color]` button `[N]` times"	144.3	7.8	3	repetition count × randomized object placement
stack_cups	"stack `[N]` `[color]` blocks"	372.8	16.1	4	initial count × to stack count
wipe_desk_rubbish	"wipe dirt off the desk"	272.9	9.0	2	initial placement of rubbish × randomized object placement
lift_and_check	"lift blocks and look for the white piece"	277.3	14.3	3	hidden-chip position × randomized object placement

I. Real-world Tasks
Interaction with real-world objects.

(a) Push button times

Task Description: The robot is instructed to identify the target colored button and press it the requested number of times. Depending on the variation, the button must be pressed once, twice, three times, four times, or five times.

Success Metric: The task is considered successful once the target button has been pressed exactly the requested number of times and the robot arm has withdrawn upward.

Objects: Three colored push buttons and their button mechanisms.

Variation Number: 3 **Keyframes:** 4, 6 or 8

Language Instructions: *"Push the button* `<count>`*."*

(b) Stack cups

Task Description: The robot is instructed to identify blocks of the specified color and stack the requested number of them vertically. The requested stack contains two, three, or four blocks.

Success Metric: The task is considered successful once the requested number of target blocks is detected in the stacking region and the robot is no longer holding an object.

Objects: Four target blocks, four distractor blocks, and a stacking region.

Variation Number: 4 **Keyframes:** 11, 17 or 23

Language Instructions: *"Stack* `<1-2>` *cups."*

(c) Wipe desk rubbish

Task Description: The robot must first place the rubbish in the bin, revealing dirt at its former location. It must then pick up the sponge and wipe all of the dirt from the desk.

Success Metric: The task is considered successful once the rubbish is in the bin and all dirt spots have been removed from the desk.

Objects: A piece of rubbish, a bin, a sponge, dirt spots, and a desk.

Variation Number: 2 **Keyframes:** 9

Language Instructions: *"CLean the desk."*

(d) Lift and check

Task Description: The robot is instructed to inspect a row of blocks to find the white chip hidden underneath one of them. It must lift the blocks one by one until the chip is found and then remove it.

Success Metric: The task is considered successful once the hidden white chip has been found and the corresponding object has been moved to the designated success region.

Objects: Four colored blocks, a hidden white chip, and a success region.

Variation Number: 3 **Keyframes:** 6, 11, 16, or 21

Language Instructions: *"Lift blocks one by one to find the white chip underneath and remove it."*

A.3 Task Visualization

Here we provide the complete visual sequences of all HIDE tasks in Table 5–7. Red boxes highlight pairs of observations that are visually similar while corresponding to different hidden task states. Although the current visual appearance provides little information to distinguish these states, the latent state determines the appropriate next action. For example, visually similar scenes may correspond to different repetition counts, different historically observed object properties, or different execution stages, and therefore lead to different action choices. These examples illustrate the central challenge of HIDE: the current observation alone is insufficient to determine the next action without recovering task-relevant information from interaction history.

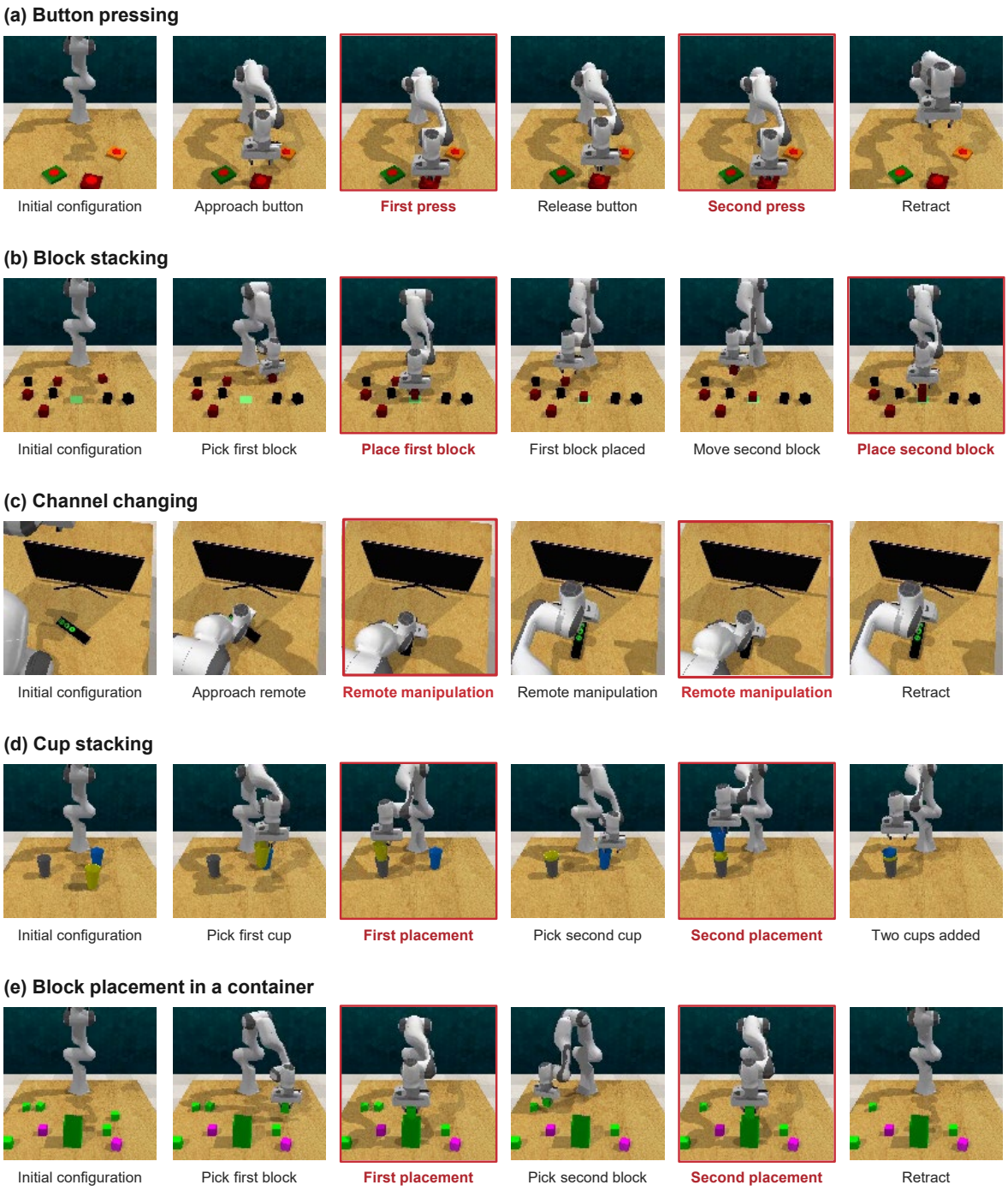


Figure 5: Illustration of Repetition Counting (RS) tasks.

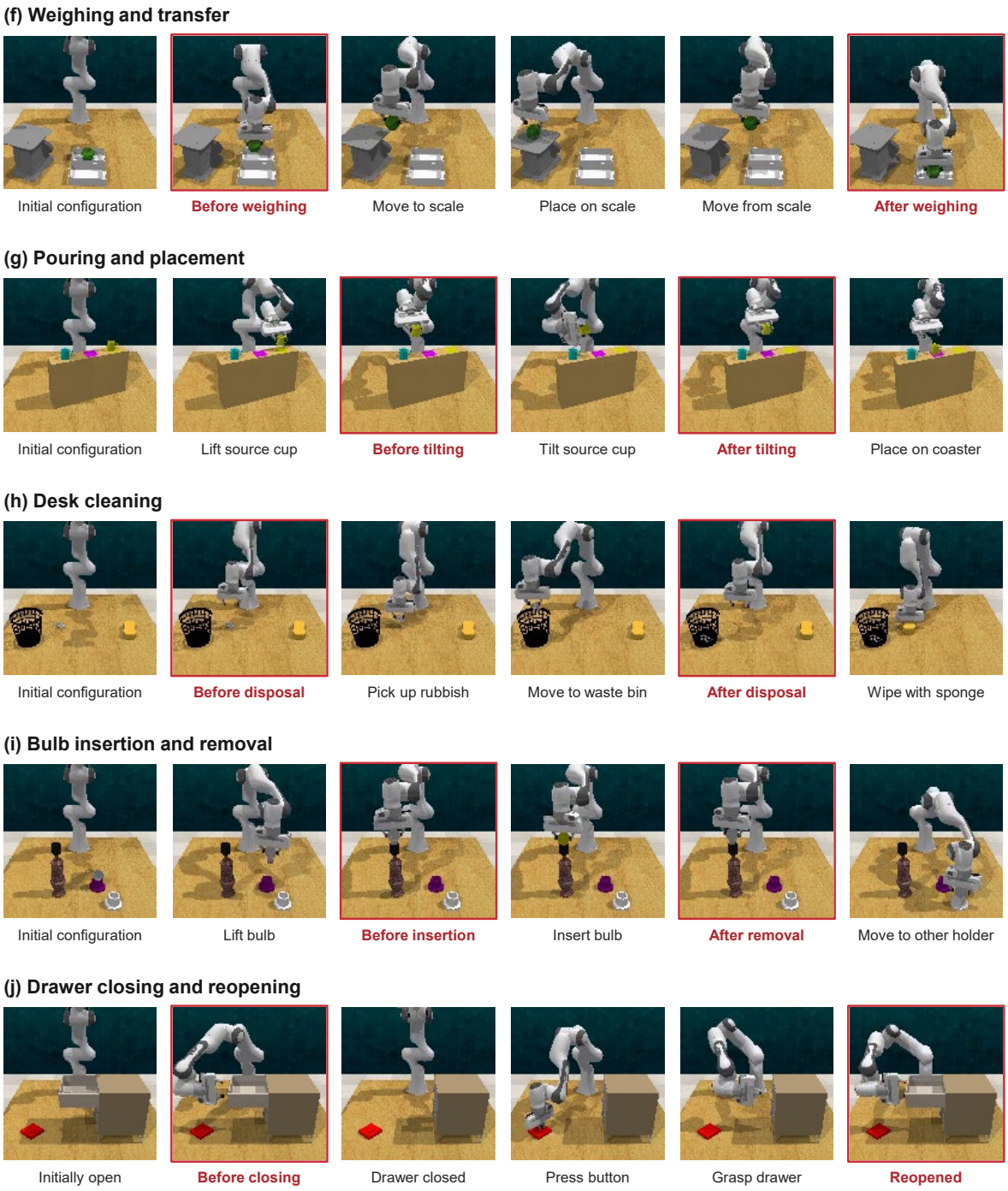


Figure 6: Illustration of Historical-State Recall (HSR) tasks.

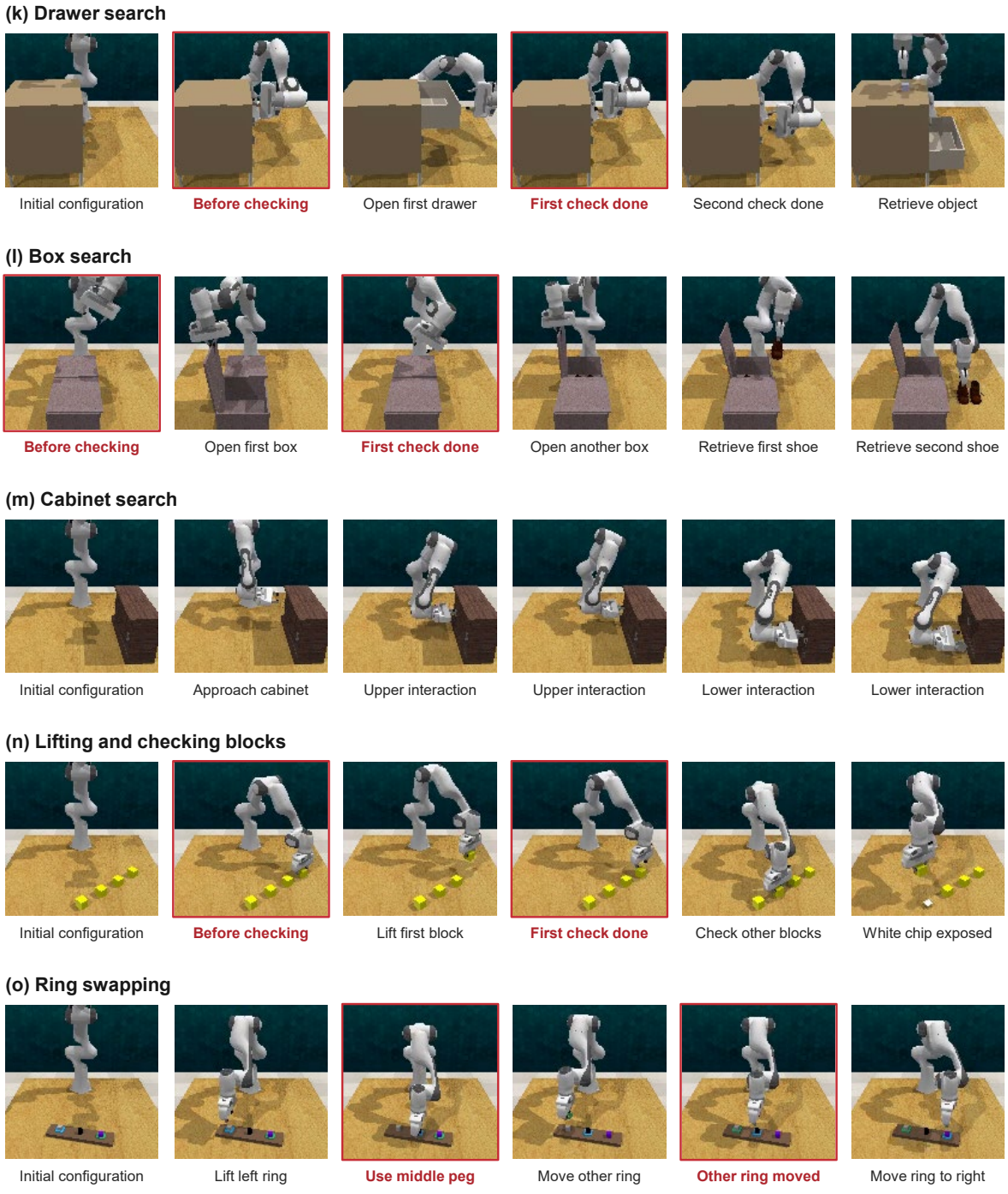


Figure 7: Illustration of Execution-Progress Tracking (EPT) tasks.

B Additional Experimental Details

B.1 Benchmarks and Data

HIDE extends RLBench with 15 tasks organized into three categories—repetition counting, historical-state recall, and execution-progress tracking—with five tasks per category. We follow the RLBench workflow for demonstration generation, language-conditioned task specification, and train–test separation. For HIDE and the standard 18-task RLBench suite, we use 100 training demonstrations per task and 25 held-out test episodes per task. Training and test sets contain separate demonstration episodes with scene and task variations. HIDE serves as our principal benchmark for comparisons and component ablations, while the standard RLBench suite evaluates general manipulation performance. We use `The Colosseum` to assess robustness, reporting performance in clean scenes and under its benchmark-defined perturbations.

B.2 Simulation and Observations

The simulation experiments use CoppeliaSim through PyRep, with a 7-DoF Franka Emika Panda robot operating in a tabletop workspace. Our 3D policy uses RGB-D observations from four cameras: front, left shoulder, right shoulder, and wrist, each at 128×128 resolution. Calibrated observations are combined into a point cloud, together with the language instruction and robot proprioception. The policy predicts keyframe actions comprising a target end-effector position, orientation, and gripper state. An OMPL-based motion planner executes the corresponding motion to the target pose.

Our HIDE comparison covers three families of policies. Vision-language-action models include OpenVLA (Kim et al., 2024), OpenVLA-OFT (Kim et al., 2025), π_0 (Black et al., 2024), $\pi_{0.5}$ (Physical Intelligence et al., 2025), and GR00T-N1.7 (NVIDIA et al., 2025). We also compare against 3D manipulation policies, including RVT (Goyal et al., 2023), RVT-2 (Goyal et al., 2024), and SAM2Act (Fang et al., 2025). Memory-based baselines include μVLA (Cherepanov et al., 2026), MME (Dai et al., 2026), and SAM2Act+. SAM2Act serves as the backbone reference for studying the effect of introducing memory. For controlled ablations, we separately report the common 20-epoch initialization, denoted Base20, and a matched continuation that disables memory reading. These controls are distinguished from the independently reported SAM2Act baseline. For external benchmarks, results from official weights, our reproductions, and published reports are identified separately.

B.3 Training Protocol

We train a single SEEK policy jointly across all 15 HIDE tasks. Temporal training initializes from 20-epoch base model and runs for up to 20 additional epochs. All component variants independently start from this same checkpoint; progressive component additions do not reuse weights from the preceding variant. We evaluate checkpoints after 10, 15, and 20 additional epochs, using epoch 15 as the primary comparison point and the other checkpoints to analyze training-duration effects. During temporal training, the coarse and fine policy branches are trainable, while the SAM 2 image encoder and its LoRA parameters are frozen. Unless explicitly ablated, the historical read budget is six entries, with the stage-conditioning block counted separately. The continuation control retains the common auxiliary stage supervision while disabling memory reading, so it isolates the effect of memory access under the continuation protocol.

B.4 Evaluation Metrics

The primary metric is task success rate. On HIDE, we report per-task success rates, category averages over five tasks, and an overall unweighted average over all 15 tasks. Each reported model uses one checkpoint across all tasks. RLBench and `The Colosseum` follow their respective evaluation protocols, with clean and perturbed success rates reported separately for the latter. Real-robot results are reported as successful trials out of the total trials for each task.

B.5 Real-Robot Evaluation

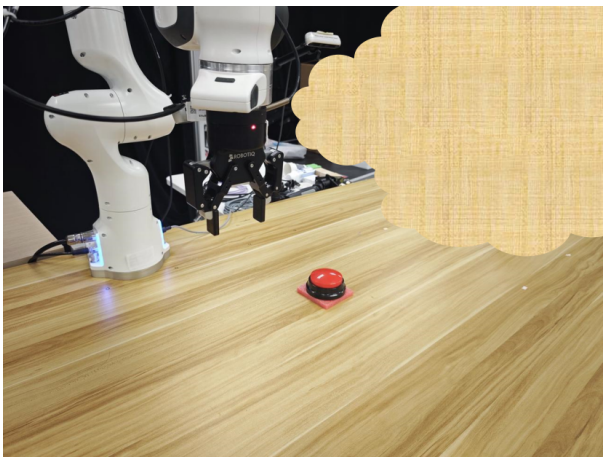
We further evaluate on a Franka Emika Panda robot equipped with a Robotiq gripper and an external Intel RealSense D455 RGB-D camera. For each task, we collect 50 real-robot demonstrations and evaluate the trained policy over 25 test trials, with the predefined task variations uniformly represented during both data collection and evaluation. The four tasks are pressing a button a specified number of times, stacking a specified number of cups on the middle cup, cleaning a desk, and lifting blocks to search for a white piece. We compare SEEK with SAM2Act+ (Fang et al., 2025)



Figure 8: Scene of Real-world experiment.

and $\pi_{0.5}$, and report success rates over the 25 evaluation trials. All methods use separately trained real-robot policies. The scene is shown in Figure 8, 9.

C DETAILED RESULTS



(a) Repeated button pressing.



(b) Cup stacking.



(c) Desk cleaning.



(d) Block lifting and white-piece search.

Figure 9: **Real-world task setups.** The four physical tasks used for real-robot evaluation: repeated button pressing, cup stacking, desk cleaning, and lifting blocks to search for a white piece.

Table 7: **Full Comparisons of Multi-Task Performance on RLBench.** We report success rates (%) on 18 RLBench tasks (James et al., 2020). SAM2Act results reported in the original paper are shown in light gray. An asterisk (*) denotes our evaluation using the officially released SAM2Act weights. Their slightly lower performance under our unified evaluation protocol may reflect differences in evaluation environments, dependency versions, and simulation stochasticity. The last two rows report our method without and with memory, respectively. Avg. Rank is recomputed across all listed rows by averaging per-task ranks, with average ranks assigned to ties. Bold and underlined values indicate the best and second-best results in each column across all rows, respectively; ties are marked equally.

Method	Avg. Success ↑	Avg. Rank ↓	Close Jar	Drag Stick	Insert Peg	Meat off Grill	Open Drawer	Place Cups	Place Wine	Push Buttons
Image-BC (CNN) (Jang et al., 2022)	1.3	14.08	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0
Image-BC (ViT) (Jang et al., 2022)	1.3	14.25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
C2F-ARM-BC (James et al., 2022)	20.1	13.03	24.0	24.0	4.0	20.0	20.0	0.0	8.0	72.0
HiveFormer (Guhur et al., 2022)	45.3	11.06	52.0	76.0	0.0	**100.0**	52.0	0.0	80.0	84.0
PolarNet (Chen et al., 2023)	46.4	10.19	36.0	92.0	4.0	**100.0**	84.0	0.0	40.0	96.0
PerAct (Shridhar et al., 2023)	49.4 ± 4.3	9.83	55.2 ± 4.7	89.6 ± 4.1	5.6 ± 4.1	70.4 ± 2.0	88.0 ± 5.7	2.4 ± 3.2	44.8 ± 7.8	92.8 ± 3.0
Act3D (Gervet et al., 2023)	65.0	7.78	92.0	92.0	27.0	94.0	93.0	3.0	80.0	99.0
RVT (Goyal et al., 2023)	62.9 ± 3.7	8.11	52.0 ± 2.5	99.2 ± 1.6	11.2 ± 3.0	88.0 ± 2.5	71.2 ± 6.9	4.0 ± 2.5	91.0 ± 5.2	**100.0** ± 0.0
RVT-2 (Goyal et al., 2024)	81.4 ± 3.1	4.56	**100.0** ± 0.0	99.0 ± 1.7	40.0 ± 0.0	99.0 ± 1.7	74.0 ± 11.8	38.0 ± 4.5	95.0 ± 3.3	**100.0** ± 0.0
3D Diffuser Actor (Ke et al., 2024)	81.3	4.81	96.0 ± 2.5	**100.0** ± 0.0	65.6 ± 4.1	96.8 ± 1.6	89.6 ± 4.1	24.0 ± 7.6	93.6 ± 4.8	98.4 ± 2.0
SAM-E (Zhang et al., 2024)	70.6 ± 0.7	6.22	82.4 ± 3.6	**100.0** ± 0.0	18.4 ± 4.6	95.2 ± 3.3	**95.2** ± 5.2	0.0 ± 0.0	94.4 ± 4.6	**100.0** ± 0.0
SAM2Act (Fang et al., 2025)	**86.8** ± 0.5	4.03	99.0 ± 2.0	99.0 ± 2.0	84.0 ± 5.7	98.0 ± 2.3	83.0 ± 6.0	**47.0** ± 6.0	93.0 ± 3.8	**100.0** ± 0.0
SAM2Act* (Fang et al., 2025)	84.0 ± 1.0	4.14	**100.0** ± 0.0	**100.0** ± 0.0	**92.0** ± 2.8	99.0 ± 1.7	81.0 ± 3.3	25.0 ± 8.7	91.0 ± 3.3	**100.0** ± 0.0
w/o memory (Ours)	84.1 ± 1.0	4.11	**100.0** ± 0.0	**100.0** ± 0.0	81.0 ± 8.2	98.0 ± 2.0	77.0 ± 5.9	37.0 ± 5.2	**99.0** ± 1.7	**100.0** ± 0.0
w/ memory (Ours)	84.7 ± 0.6	**3.81**	**100.0** ± 0.0	**100.0** ± 0.0	80.0 ± 8.9	98.0 ± 2.0	78.0 ± 4.5	44.0 ± 7.5	94.0 ± 4.5	**100.0** ± 0.0

Method	Put in Cupboard	Put in Drawer	Put in Safe	Screw Bulb	Slide Block	Sort Shape	Stack Blocks	Stack Cups	Sweep to Dustpan	Turn Tap
Image-BC (CNN) (Jang et al., 2022)	0.0	8.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	8.0
Image-BC (ViT) (Jang et al., 2022)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	16.0
C2F-ARM-BC (James et al., 2022)	0.0	4.0	12.0	8.0	16.0	8.0	0.0	0.0	0.0	68.0
HiveFormer (Guhur et al., 2022)	32.0	68.0	76.0	8.0	64.0	8.0	8.0	0.0	28.0	80.0
PolarNet (Chen et al., 2023)	12.0	32.0	84.0	44.0	56.0	12.0	4.0	8.0	52.0	80.0
PerAct (Shridhar et al., 2023)	28.0 ± 4.4	51.2 ± 4.7	84.0 ± 3.6	17.6 ± 2.0	74.0 ± 13.0	16.8 ± 4.7	26.4 ± 3.2	2.4 ± 2.0	52.0 ± 0.0	88.0 ± 4.4
Act3D (Gervet et al., 2023)	51.0	90.0	95.0	47.0	93.0	8.0	12.0	9.0	92.0	94.0
RVT (Goyal et al., 2023)	49.6 ± 3.2	88.0 ± 5.7	91.2 ± 3.0	48.0 ± 5.7	81.6 ± 5.4	36.0 ± 2.5	28.8 ± 3.9	26.4 ± 8.2	72.0 ± 0.0	93.6 ± 4.1
RVT-2 (Goyal et al., 2024)	66.0 ± 4.5	96.0 ± 0.0	96.0 ± 2.8	88.0 ± 4.9	92.0 ± 2.8	35.0 ± 7.1	**80.0** ± 2.8	69.0 ± 5.9	**100.0** ± 0.0	99.0 ± 1.7
3D Diffuser Actor (Ke et al., 2024)	**85.6** ± 4.1	96.0 ± 3.6	97.6 ± 2.0	82.4 ± 2.0	97.6 ± 3.2	44.0 ± 4.4	68.3 ± 3.3	47.2 ± 8.5	84.0 ± 4.4	99.2 ± 1.6
SAM-E (Zhang et al., 2024)	64.0 ± 2.8	92.0 ± 5.7	95.2 ± 3.3	78.4 ± 3.6	95.2 ± 1.8	34.4 ± 6.1	26.4 ± 4.6	0.0 ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0
SAM2Act (Fang et al., 2025)	75.0 ± 3.8	99.0 ± 2.0	**98.0** ± 2.3	89.0 ± 2.0	86.0 ± 4.0	64.0 ± 4.6	76.0 ± 8.6	78.0 ± 4.0	99.0 ± 2.0	96.0 ± 5.7
SAM2Act* (Fang et al., 2025)	73.0 ± 7.1	99.0 ± 1.7	92.0 ± 4.9	**90.0** ± 4.5	85.0 ± 4.4	**67.0** ± 3.3	39.0 ± 4.4	**85.0** ± 5.2	**100.0** ± 0.0	94.0 ± 4.5
w/o memory (Ours)	78.0 ± 2.0	98.0 ± 2.0	95.0 ± 4.4	83.0 ± 7.1	98.0 ± 2.0	66.0 ± 4.5	42.0 ± 6.0	73.0 ± 5.2	**100.0** ± 0.0	88.0 ± 5.7
w/ memory (Ours)	80.0 ± 4.0	**100.0** ± 0.0	97.0 ± 1.7	83.0 ± 3.3	**100.0** ± 0.0	65.0 ± 3.3	40.0 ± 12.0	78.0 ± 6.6	**100.0** ± 0.0	87.0 ± 3.3

Table 8: **`The Colosseum` results.** We compare the selected methods. Entries report success rates (%), with relative changes (%) from the corresponding unperturbed evaluation in parentheses. Non-dagger literature results are aggregated over the perturbation-specific task subsets used in SAM2Act. Relative changes use the Clean means of the same task subsets. Average is the unweighted mean over the 12 individual perturbations, excluding All-Mixed; its relative change is measured against overall Clean. PerAct and RVT are aggregated from the original benchmark paper; RVT-2 is aggregated from the BridgeVLA paper; SAM2Act uses paper-reported results. Both memory variants are our methods. † denotes legacy RVT results with different task subsets, retained for reference and excluded from ranking. The best and second-best displayed success rates among the listed non-dagger methods are bold and underlined, respectively; ties share the same rank.

Method	Clean	Average	MO-Color	RO-Color	MO-Texture	RO-Texture	MO-Size	RO-Size
PerAct	34.5 (0.0)	28.7 (↓ 16.6)	24.0 (↓ 30.3)	31.7 (↓ 14.4)	28.8 (↓ 24.2)	17.7 (↓ 16.2)	33.6 (↓ 14.0)	29.3 (↓ 15.2)
RVT†	43.6 (0.0)	36.3 (↓ 16.7)	26.0 (↓ 40.4)	31.3 (↓ 29.3)	44.8 (↑ 4.7)	41.1 (↑ 4.3)	35.3 (↓ 16.3)	40.5 (↓ 18.2)
RVT-2	67.8 (0.0)	59.5 (↓ 12.3)	53.0 (↓ 21.8)	59.2 (↓ 9.8)	**59.7** (↓ 8.3)	<u>56.7</u> (↓ 0.3)	60.9 (↓ 5.9)	53.4 (↓ 17.2)
SAM2Act	64.7 (0.0)	**62.3** (↓ 3.7)	**65.0** (↑ 0.4)	61.0 (↓ 0.9)	<u>55.4</u> (↓ 10.0)	50.3 (↓ 9.6)	**65.4** (↓ 7.5)	**58.0** (↓ 16.2)
w/o memory (Ours)	<u>68.4</u> (0.0)	61.5 (↓ 10.1)	62.1 (↓ 9.3)	<u>65.2</u> (↓ 1.2)	52.1 (↓ 11.1)	**56.8** (↓ 12.6)	<u>63.1</u> (↓ 10.3)	<u>57.3</u> (↓ 20.6)
w/ memory (Ours)	**68.6** (0.0)	<u>61.9</u> (↓ 9.8)	<u>62.4</u> (↓ 9.0)	**65.7** (↓ 0.9)	52.7 (↓ 11.0)	56.6 (↓ 12.9)	62.4 (↓ 11.4)	<u>57.3</u> (↓ 20.6)

Method	Light Color	Table Color	Table Texture	Distractor	Background Texture	Camera Pose	All-Mixed
PerAct	29.1 (↓ 15.5)	30.4 (↓ 11.9)	23.2 (↓ 32.7)	27.1 (↓ 21.3)	33.5 (↓ 2.8)	36.3 (↑ 5.4)	7.2 (↓ 79.1)
RVT†	34.0 (↓ 22.0)	30.0 (↓ 31.2)	45.2 (↑ 3.7)	18.8 (↓ 56.9)	46.4 (↑ 6.4)	42.2 (↓ 3.2)	6.4 (↓ 85.3)
RVT-2	58.0 (↓ 14.4)	62.6 (↓ 7.8)	56.6 (↓ 16.5)	<u>60.8</u> (↓ 10.5)	**68.7** (↑ 1.3)	<u>64.4</u> (↓ 5.0)	15.6 (↓ 77.1)
SAM2Act	<u>65.2</u> (↑ 0.7)	65.6 (↑ 1.3)	**65.4** (↑ 1.0)	**62.3** (↓ 3.8)	<u>68.6</u> (↑ 6.0)	**65.7** (↑ 1.6)	**26.9** (↓ 58.4)
w/o memory (Ours)	**65.5** (↓ 4.2)	<u>66.4</u> (↓ 2.9)	60.3 (↓ 11.8)	58.4 (↓ 14.6)	67.3 (↓ 1.6)	63.2 (↓ 7.6)	25.3 (↓ 63.0)
w/ memory (Ours)	64.9 (↓ 5.4)	**69.9** (↑ 1.9)	<u>60.7</u> (↓ 11.5)	58.6 (↓ 14.6)	67.6 (↓ 1.5)	63.6 (↓ 7.3)	<u>26.3</u> (↓ 61.7)