MOMENTMEM: MEMORIZING KEY MOMENTS FOR ROBOT MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Many manipulation tasks require recalling visual evidence that has left the robot's view. Yet most vision-language-action (VLA) policies see only the current observation or a short history. Such tasks call for **visual moment memory**, which captures the scene at the moments that matter. To evaluate this capability, we introduce VMBench, whose tasks require recalling patterns, locations, or both. We then propose MomentMem, which instantiates this memory with keyframes. A VLM selects keyframes, and a VLA executor predicts actions from them and the current observation. Both are trained with segment-level annotations and representative-frame sampling. MomentMem achieves state-of-the-art results on VMBench and RMBench, and improves success on a real robot. Its memory also transfers zero-shot to unseen visual appearances and adapts to a held-out task with few annotations.

1 INTRODUCTION

Many manipulation tasks require a robot to use what it saw earlier. For example, the robot may need to recall where a cube is hidden, where a demonstrator placed an object, or which cue flashed for a moment. Such evidence often leaves the view before it is needed. However, most vision-language-action (VLA) policies act on the current observation or a short temporal window (Kim et al., 2025; Black et al., 2024). They therefore cannot use evidence that is no longer visible. Recent work on memory-augmented VLAs shows that historical context improves long-horizon manipulation (Shi et al., 2026b; Cherepanov et al., 2026b). Memory is thus a key capability for robot policies.

But what form should this memory take? As shown in Figure 1, text summaries are compact and track task progress well (Chen et al., 2026a). However, many visual cues are hard to put into words, such as the pattern on a card or the exact position of a cube. Compressing the history into latent features may also lose such details (Shi et al., 2026b; Cherepanov et al., 2026b). Keeping every past frame

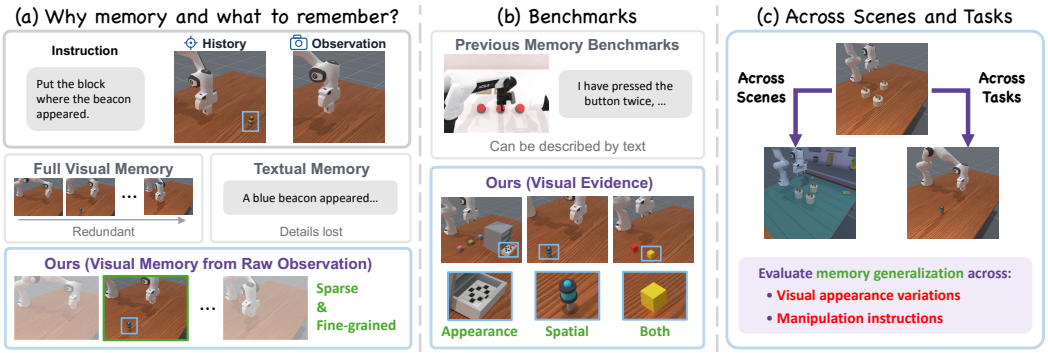


Figure 1: **Overview of MomentMem and VMBench.** **(a)** Robot policies need memory when task-relevant visual evidence is no longer observable. Text memory loses visual details, while full visual memory is redundant. MomentMem stores visual moment memory as keyframes, a sparse set of selected raw observations. **(b)** Existing memory benchmarks mostly involve information that can be described in text. VMBench instead requires recalling visual evidence about appearance, spatial location, or both. **(c)** We further evaluate whether the learned memory selection generalizes across visual appearance variations and manipulation tasks.

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.

preserves them, but makes the context long and redundant. We therefore advocate **visual moment memory**, which captures the scene at the moments that matter. The next question is which moments to remember. Hand-crafted rules are task-specific and transfer poorly. Learning the selection is more flexible, but it needs memory labels. Robot demonstrations rarely provide them, and manual annotation is costly. The learned selection should therefore generalize to new scenes and tasks with few or no additional labels.

To evaluate this capability, we introduce VMBench, a benchmark for visual moment memory. Many existing memory benchmarks involve information that text can summarize, such as how many times a button has been pressed. In contrast, each of the nine tasks in VMBench requires the robot to recall a pattern, a location, or both after it has left the view. Such evidence is hard to describe precisely in language. Each task also has a visual out-of-distribution variant, which changes the scene's appearance but keeps the task mechanics. It tests whether memory selection generalizes to new scenes.

We then propose MomentMem, which instantiates visual moment memory with keyframes. Keyframes are a natural choice. As raw past observations, they preserve the scene exactly as it appeared at key moments. Visual moment memory, however, is not limited to keyframes, which are only one practical instantiation, and other instantiations are possible. In MomentMem, a keyframe identifier, built on a vision-language model (VLM), decides whether the current observation captures a key moment. If so, the observation is stored in memory. An executor, built on a VLA, then predicts actions from the stored keyframes and the current input. The main difficulty is learning what to store. A task-relevant event usually stays visible over several consecutive frames, and any of them is a valid keyframe. Labeling only one of them as correct would penalize equally valid choices. We therefore annotate each event as a segment and sample one representative frame from it during training. The identifier thus learns to store one keyframe per event, and the executor learns to act from any valid one. Scheduled sampling further helps the identifier recover from its own earlier mistakes.

Experiments show that MomentMem provides effective visual moment memory, and this holds across diverse evaluation settings. On VMBench, MomentMem substantially outperforms both the memory-free $\pi_{0.5}$ and existing memory-augmented policies. On the existing RMBench (Chen et al., 2026a), it achieves the highest average success among all compared methods. On a real Franka robot, it also clearly improves task success over $\pi_{0.5}$. The benefit of MomentMem thus extends beyond our own benchmark and from simulation to the real world.

More importantly, the memory learned by MomentMem generalizes across scenes and tasks. Across scenes, the identifier transfers zero-shot to unseen visual appearances. The executor, trained only on these automatic labels, still outperforms all baselines by a large margin. Its performance in the new scenes is on par with that in the original ones. Across tasks, we first train the identifier on source tasks and then adapt it to a held-out task with only a few annotated trajectories. Compared with adapting on the held-out task alone, this substantially improves success and approaches the performance of full memory supervision. A trained identifier can thus label new data by itself, which reduces the cost of memory annotation.

Our contributions are summarized as follows:

- We address long-horizon robotic manipulation that requires visual moment memory. To this end, we introduce VMBench to evaluate visual moment memory, which existing benchmarks do not measure well.
- We develop MomentMem, which equips VLA policies with visual moment memory. Our technical innovations lie in the design of segment-level memory annotations and the sampling of representative frames during training.
- Experiments demonstrate that MomentMem maintains visual moment memory well for robotic manipulation, achieving state-of-the-art results on both our VMBench and the existing RMBench. Further results reveal that its memory capability generalizes across scenes in zero-shot setting and across tasks in few-shot setting.

2 Related Work

2.1 Vision-Language-Action Models

Vision-Language-Action (VLA) models connect pretrained vision-language representations with robot action generation. RT-2 (Zitkovich et al., 2023) demonstrated the transfer of web-scale

semantic knowledge to robotic control, while OpenVLA (Kim et al., 2025) and Octo (Ghosh et al., 2024) developed open generalist policies trained on diverse robot data. Recent models, including π_0 (Black et al., 2024), $\pi_{0.5}$ (Black et al., 2025), GR00T series (NVIDIA et al., 2025), GigaBrain series (GigaBrain Team et al., 2025; 2026a;b), and Gemini Robotics (Gemini Robotics Team et al., 2025), have further improved action generation and cross-task generalization.

Despite these advances, most VLAs condition on the current observation or a short context window. They therefore struggle when relevant evidence has disappeared from view (Black et al., 2024; Shi et al., 2026b; Chen et al., 2026b; Cherepanov et al., 2026b). Our work addresses this limitation by providing VLA models with task-relevant visual observations selected from the full episode history.

2.2 Robotic Manipulation Policies with Memory

Recent studies have recognized the problem of state aliasing in some robot manipulation tasks. Solving such tasks therefore requires information from trajectory history (Fang et al., 2025; Shi et al., 2026b; Chen et al., 2026a). The available information consists of the task instruction, visual observations, actions, and proprioceptive states. From this perspective, existing memory methods mainly differ in how they process and select information from these sources. Some methods maintain recurrent states, memory banks, or compressed tokens to retain information from long histories (Shi et al., 2026b;a; Cherepanov et al., 2026b; Li et al., 2026; Bagaria et al., 2026; Wang et al., 2026; Hu et al., 2026b;a). However, compressing a trajectory into a limited representation creates an information bottleneck and can lose visual moment memory. Other methods summarize the history with language or use textual plans as memory (Chen et al., 2026a; Sridhar et al., 2026; Torne et al., 2026). Such representations are compact, but concise text summaries may omit task-relevant visual details, such as fine-grained appearances. Another line of work directly selects observations from the visual history. This preserves the original visual evidence, but many existing methods rely on predefined rules or heuristics for selection (Chen et al., 2026b; Zeng et al., 2026b;a; Lei et al., 2026). Such heuristics are task-specific and adapt poorly to different tasks and interactions.

Our method instead adopts a dual-system architecture comprising a keyframe identifier and an executor. The most closely related method is EventVLA (Yang et al., 2026), which predicts keyframes from predicted future actions within a single model. Unlike EventVLA, we identify keyframes from observations with a separate identifier, decoupling keyframe identification from action prediction.

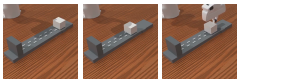
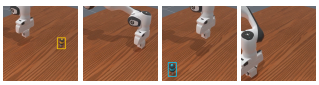
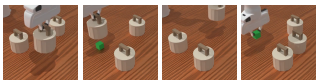
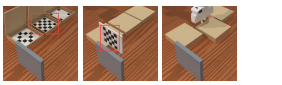
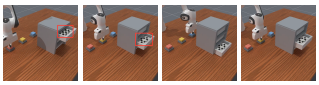
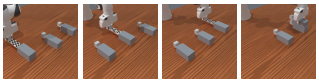
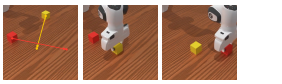
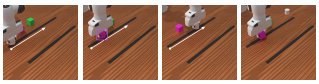
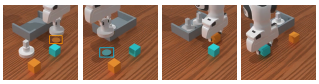
2.3 Robotic Memory Benchmarks

Along with the development of memory-augmented robotic policies, many benchmarks have emerged to evaluate the effectiveness of different memory mechanisms. RMBench (Chen et al., 2026a) pioneered the systematic evaluation of memory in robotic imitation learning. Subsequent benchmarks categorize memory into different dimensions and evaluate robotic policies across these categories, such as MIKASA-Robo (Cherepanov et al., 2026a), MemoryBench (Fang et al., 2025) and RoboMME (Dai et al., 2026). Other benchmarks focus on memory in long-horizon tasks, which involve extended trajectories and require precise manipulation (Lei et al., 2026; Yang et al., 2026; Chung et al., 2026; Shah et al., 2026a). In contrast, our VMBench focuses on visual moment memory, and further evaluates whether memory selection generalizes across scenes and tasks.

3 VMBench: A Visual Memory Benchmark for Robotic Manipulation

We introduce VMBench to test visual moment memory in manipulation tasks. It is developed on SAPIEN (Xiang et al., 2020) physics engine and Maniskill3 (Mu et al., 2021; Gu et al., 2023; Tao et al., 2024) platform. As shown in Table 1, in every task, task-relevant evidence appears before it is needed and later becomes unavailable, and the information is difficult to summarize in language. We design the tasks to cover visual appearance, spatial locations, and appearance–location bindings. The last requires remembering which visual cue corresponds to which location, rather than recalling appearance and location independently. Each demonstration also provides segment-level keyframe annotations. Each segment marks an interval in which the evidence is visible. The policy must retain this evidence and use it during later physical execution. Per-task visual evidence and key-segment counts are summarized in Appendix A, and the complete task instructions are provided in Appendix B.

Table 1: **Overview of the nine benchmark tasks**, grouped by the information they require. Images illustrate each task sequence, and the final column shows its visual out-of-distribution (OOD) setting.

Category	Task	Description	Task sequence	Visual OOD
Spatial	**Spring Compression**	Compress the spring back to its deepest previously reached position.		
	Flash Beacon	Move the cube to the location where the beacon briefly appeared.		
	Explore Cube	Search all the cups and retrieve the hidden cube.		
Appearance	**Hidden Pattern**	Push the box whose hidden pattern matches the pattern on the card.		
	Drawer Match	Open the upper or middle drawer whose pattern matches that of the bottom drawer.		
	Scroll Discovery	Reveal the scroll patterns, then stack the two matching scrolls.		
Spatial & Appearance	**Placement Imitation**	Move each cube to its position shown in the demonstration.		
	Mine Sweeper	Place each cube where the detector briefly showed the same color.		
	Seal Checking	Lift the stamps, then place each cube on the matching revealed color.		

For the main experiments, we collect 100 successful demonstrations per task, totaling 900 demonstrations across the nine tasks, using scripted closed-loop experts with a Panda manipulator. Each demonstration records RGB observations from an overview camera and a wrist camera, both at 224×224 resolution, with a control frequency of 20 Hz. Task-specific randomization varies object positions, target locations, and visual-cue associations. Each demonstration includes one to three automatically annotated key segments marking intervals when task-relevant evidence is visible. Dataset collection, randomization, annotation, and conversion procedures are detailed in Appendix C.

The differences between VMBench and existing memory-based robotic manipulation benchmarks primarily lie in two aspects. First, VMBench places particular emphasis on visual moment details. Concise text summaries may not faithfully preserve all task-relevant visual information. Therefore, each task in VMBench requires precisely memorizing and retrieving task-relevant visual cues. For example, in the Mine Sweeper task, the robot holds a detector and scans a "minefield" on the tabletop. As the detector approaches each target location, its color changes, and returns to its original color once it moves away. The robot must then recall these color changes and use them to place each block at its corresponding target location.

Second, we provide both standard and visual out-of-distribution (OOD) settings for each task. Examples are shown in Table 1. The visual OOD setting changes room and table appearance, lighting, and background or tabletop distractors outside the manipulation area, while preserving task mechanics, memory events, and success conditions. Details of the visual OOD construction are provided in Appendix C. We use the visual OOD setting to assess the generalization under unseen visual conditions.

4 MomentMem: Memorizing Key Moments for Robot Manipulation

4.1 Overview

As shown in Figure 2 (a), we present MomentMem, a hierarchical framework that gives a robot policy visual moment memory. MomentMem consists of two components. A keyframe identifier, imple-

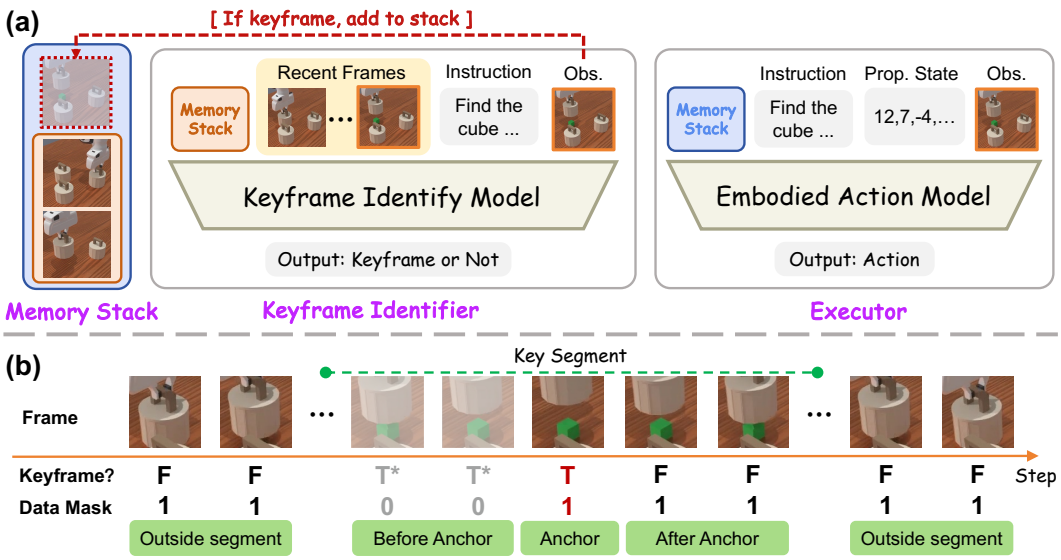


Figure 2: **The methodology of MomentMem.** **(a)** The keyframe identifier judges whether the current observation is a task-relevant keyframe, and the executor uses previously selected keyframes to predict actions. **(b)** At the beginning of each epoch, we sample one anchor for each key segment. Frames before the anchor remain in the context of the identifier, but do not contribute to its loss. Frames after the anchor are labeled as negative, because the anchor already represents the same event.

mented as a VLM, judges whether the current observation is a task-relevant keyframe. An executor, implemented as a VLA, predicts actions from the selected keyframes. The identifier passes only the selected keyframes to the executor, instead of the full observation sequence. This design retains the task-relevant history while avoiding distracting observations and context explosion (Shah et al., 2026b).

At time t, the agent receives a visual observation o_t, a proprioceptive state s_t, and a task instruction l. The identifier π_S runs every k steps. It takes the keyframe stack M_{t-1}, a buffer B_t of recent subsampled observations including o_t, and the instruction l. It estimates the probability p_t of a "Yes" decision and stores the current observation when $p_t > 0.5$. The executor π_E predicts an action chunk A_t from the stack, the current observation, the state, and the instruction.

$$p_t = \pi_S(\text{Yes} \mid M_{t-1}, B_t, l), \quad M_t = \begin{cases} M_{t-1} \cup \{o_t\} & p_t > 0.5 \\ M_{t-1} & p_t \le 0.5 \end{cases}, \quad A_t \sim \pi_E(\cdot \mid M_t, o_t, s_t, l) \tag{1}$$

where M_t stands for the keyframe stack.

Both components are trained from the same segment-level annotations. We describe these annotations in Section 4.2, and the training of the two components in Section 4.3 and Section 4.4.

4.2 MomentMem Keyframe Annotation

As shown in Figure 2 (b), a task-relevant event often stays visible for several consecutive frames. These frames are equivalent evidence of the same event. Labeling one fixed frame as the only correct keyframe is therefore too strict, because it penalizes the model for selecting an equally valid neighbor. We instead annotate each event with an inclusive segment $\mathcal{S}_j = [u_j, v_j]$. Frames within a segment are treated as equivalent observations of the same event.

This annotation defines the desired behavior of the identifier. Specifically, the identifier should select one frame per segment, avoid additional selections within it, and reject frames outside all segments. We obtain the segments automatically in simulation, where privileged information marks when each piece of evidence is visible. We label the segments manually in the real-world experiments.

4.3 The Keyframe Identifier of MomentMem

We fine-tune a VLM to serve as the keyframe identifier, using segment-level keyframe annotations as supervision. Given the context defined in Equation 1, the model outputs a single word, "Yes" or "No", indicating whether the current observation should be stored as a keyframe.

Anchor sampling. The identifier should also recover from a miss. If it misses the first frame of a segment, it should still select a later frame from the same segment. We therefore sample one anchor c_j from every annotated segment at the start of each epoch. The anchor receives a positive label. Frames after the anchor receive negative labels, because the event is already stored in the keyframe stack. Frames before the anchor are masked from the loss, because they are also valid representatives of the event. Such frames remain in the preceding context, which simulates a miss at the earlier steps. Frames outside all segments receive negative labels. We resample the anchors across epochs, which prevents the model from binding an event to one fixed frame. Most frames lie outside all segments and are easy negatives. We use focal loss to down-weight these frames.

Scheduled sampling. Teacher forcing always gives the identifier a correct memory history. A single error at inference time instead changes every later context. We reduce this mismatch with sequence-level scheduled sampling. At the start of each epoch, the current model replays every training trajectory and predicts a label for each frame. For each frame, we then insert either the teacher decision or the model decision into the history. The probability of using the model decision increases during training.

4.4 The Memory-Conditioned Executor of MomentMem

The executor does not predict keyframe labels. The segment annotations instead define its memory input. For each training sample at time t, we independently draw one anchor from every segment, and keep only the anchors before t as the keyframe stack in context. This procedure exposes the executor to different valid representatives of each event. It also approximates the variation that the identifier produces at inference time. The identifier learns when to store a representative, while the executor learns how to act from such representatives.

5 Experiments and Results

We organize our experiments around four questions. **Q1:** Does MomentMem improve closed-loop success on tasks that require visual moment memory? **Q2:** Does the identifier transfer to unseen visual appearances without new memory annotations? **Q3:** Can the identifier adapt to an unseen task from only a few memory annotations? **Q4:** Does MomentMem remain effective beyond VMBench, on an existing benchmark and a real robot?

5.1 Experimental Setup

Baselines. We compare MomentMem against $\pi_{0.5}$ (Black et al., 2025) and three memory-augmented policies. Mem-0 (Chen et al., 2026a) uses explicit memory and high-level planning to condition action execution. NativeMEM (Wang et al., 2026) compresses past visual observations into compact memory tokens. EventVLA (Yang et al., 2026) learns sparse visual evidence memory jointly with action prediction. Together, these baselines cover memory-free execution, high-level memory, compressed visual history, and learned visual evidence memory.

Training and evaluation. On VMBench, all methods are trained on the same data, with 100 trajectories per task, and are jointly trained across all tasks. Each task is evaluated on 100 rollouts with unseen initial configurations. For MomentMem, the identifier and executor are trained separately using segment annotations. At test time, executor receives only the frames selected online by identifier.

5.2 A1: MomentMem Improves Visual Moment Memory Recall

MomentMem substantially improves closed-loop success. As shown in Table 2, MomentMem achieves 50.7% average success, 32.4 points above the strongest baseline, and performs best on six of nine tasks. The largest gains occur on Placement Imitation, Mine Sweeper, and Seal Checking, which require retaining precise visual moment memory after it disappears from view. These results support preserving selected raw observations over compressing history into language or features.

Memory-free execution fails on these tasks. $\pi_{0.5}$ reaches only 7.0% average success, and scores 0% on four of the nine tasks. On Mine Sweeper, we observe that $\pi_{0.5}$ often places a block inside

Table 2: **Performance comparison on VMBench.** We compare MomentMem with several baselines on VMBench. All methods are trained on the same dataset, with 100 trajectories per task. We evaluate all methods under 100 unseen initialization conditions per task. For each method, the models are jointly trained and evaluated across all tasks.

	Explore Cube	Mine Sweeper	Hidden Pattern	Seal Checking	Spring Compression	Drawer Match	Flash Beacon	Scroll Discovery	Placement Imitation	Avg.
$\pi_{0.5}$	0%	1%	24%	0%	0%	21%	0%	16%	1%	7.0%
Mem-0	0%	0%	**34%**	0%	25%	**58%**	0%	17%	2%	15.1%
EventVLA	2%	1%	24%	2%	34%	51%	15%	**29%**	6%	18.2%
NativeMEM	0%	0%	25%	0%	44%	44%	9%	19%	1%	15.8%
Ours	**8%**	**85%**	29%	**73%**	**81%**	34%	**25%**	23%	**98%**	**50.7%**

the candidate region, but not at the location specified by the earlier visual moment memory. This suggests that the base policy can learn the manipulation procedure, but cannot use information that is no longer visible. This failure mode supports memory as an important bottleneck in this task.

The advantage depends on what the task needs to remember. The strongest gains occur on tasks requiring appearance–location bindings. Counterfactual tests show that MomentMem adjusts its actions to remembered color and spatial cues. EventVLA responds less consistently to edited spatial cues and also exhibits missed cue frames and execution failures. These diagnostics suggest that the gap involves memory selection, memory use, and physical execution. However, tasks in the Appearance category remain challenging for all evaluated methods. On Hidden Pattern, both MomentMem and EventVLA rarely follow an edited target pattern, even when keyframes cover both relevant events. This points to a shared difficulty in using retained texture information for target selection. Appendix F provides detailed analyses.

5.3 A2: Memory Selection Transfers to Unseen Visual Appearances

We test whether the identifier still selects the right frames when the appearance of the scene changes. This is done by training the identifier only on the standard appearance and applying it directly to trajectories from the visual OOD setting. The resulting keyframes then train an executor on the target-appearance action data. The new appearance therefore provides action demonstrations but no human memory annotations. We apply the same protocol to Mem-0 and EventVLA, which keep their trained memory modules fixed and retrain only their action components. NativeMEM has no separate memory module, so we retrain it entirely on the target-appearance data.

Table 3: **Success rates on VMBench with unseen appearances.** For MomentMem, Mem-0, and EventVLA, memory-related components trained on the standard appearance automatically annotate target-appearance demonstrations. We train their action components on these demonstrations using only the generated memory-related annotations, and retain the transferred memory components for evaluation. NativeMEM compresses visual history without extra memory annotations; we train and evaluate it directly on target-appearance data.

	Explore Cube	Mine Sweeper	Hidden Pattern	Seal Checking	Spring Compression	Drawer Match	Flash Beacon	Scroll Discovery	Placement Imitation	Avg.
Mem-0	0%	0%	24%	0%	28%	21%	0%	1%	0%	8.2%
EventVLA	0%	1%	**40%**	0%	30%	48%	6%	4%	4%	14.8%
NativeMEM	0%	0%	31%	0%	31%	49%	8%	25%	3%	16.3%
Ours	**12%**	**79%**	35%	**55%**	**79%**	**51%**	**59%**	**28%**	**99%**	**55.2%**

Memory selection transfers to unseen appearances without new annotations. As shown in Table 3, MomentMem reaches 55.2% average success under appearance changes, while the strongest baseline reaches 16.3%. It achieves the best performance on eight of the nine tasks. The performance also matches the standard setting, where MomentMem reaches 50.7%. We hypothesize that the VLM backbone recognizes the same task-relevant event across appearances. The identifier only maps this recognition to a binary decision, which depends less on the training appearance.

Automatic memory labels are sufficient to train the executor in the new appearance. The executor in this setting never receives a manually annotated memory label. All of its memory inputs come from the transferred identifier. These results indicate that a trained identifier can annotate new trajectories, which reduces the annotation cost when the visual appearance changes.

5.4 A3: Memory Selection Transfers to an Unseen Task with Few Annotations

We test whether the identifier transfers to Flash Beacon, held out from eight source tasks. Ours adapts the source-trained identifier using five memory-annotated target trajectories, then automatically labels all 100 target demonstrations for executor training. Full Memory Supervision trains only on Flash Beacon with all target memory annotations, providing a task-specific positive control. From Qwen instead adapts directly from pretrained Qwen using the same five trajectories and labeling procedure, testing whether Qwen's prior knowledge alone explains the result. All settings use the same 100 action demonstrations and 300 evaluation seeds; few-shot refers only to memory supervision.

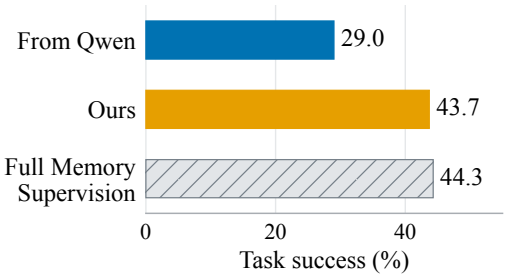


Figure 3: **Memory selection transfers with few target annotations.** On Flash Beacon, Ours reaches 43.7% success with five annotated trajectories, close to Full Memory Supervision (44.3%) and above From Qwen (29.0%). All settings use the same 300 evaluation seeds.

Ours reaches 43.7% success, close to Full Memory Supervision at 44.3% (Figure 3). From Qwen reaches 29.0%, suggesting that the transfer benefit extends beyond the pretrained backbone's prior knowledge. Together, these results support few-shot cross-task transfer of memory selection on this held-out task. Appendix H provides the full protocol and baseline comparison.

5.5 A4: MomentMem Is Applicable Across Benchmarks and Real-World Tasks

Table 4: **Performance comparison on RMBench.** For Mem-0 and EventVLA, we directly use the results reported in their original papers (Chen et al., 2026a; Yang et al., 2026). We train NativeMEM ourselves. The reported Mem-0 results are obtained by training a separate model for each task, while the other methods are trained jointly across all tasks.

	Observe and Pick Up	Rearrange Blocks	Put Back Block	Swap Blocks	Swap T	Battery Try	Blocks Ranking Try	Cover Blocks	Press Button	Avg.
Mem-0	4%	89%	90%	67%	14%	28%	18%	68%	0%	42.0%
EventVLA	21%	**96%**	95%	96%	87%	35%	**81%**	**97%**	3%	67.9%
NativeMEM	0%	3%	0%	62%	0%	13%	0%	0%	0%	8.7%
Ours	**82%**	78%	**100%**	**99%**	**97%**	**56%**	22%	84%	**10%**	**69.8%**

MomentMem remains competitive on an existing memory benchmark. We evaluate MomentMem on RMBench (Chen et al., 2026a) without changing its architecture. As shown in Table 4, MomentMem reaches 69.8% average success, essentially matching EventVLA at 67.9%. It obtains the best result on six of the nine tasks, including a 61-point gain over the strongest baseline on Observe and Pick Up. RMBench therefore shows that the same keyframe-memory architecture transfers beyond the tasks for which VMBench was designed.

MomentMem also improves physical robot execution.

We evaluate MomentMem on a Franka Panda robot using real-world counterparts of Placement Imitation and Explore Cube, as illustrated in Figure 4. We collect 51 and 54 demonstrations respectively, and manually annotate their key segments. Each method is evaluated over 30 trials per task. As shown in Table 5, MomentMem completes 17 more episodes than $\pi_{0.5}$ across the 60 trials. The larger gain occurs on Placement Imitation,

Table 5: Real-world task success over 30 trials per task.

Task	$\pi_{0.5}$	MomentMem
Placement Imitation	3/30	**14/30**
Explore Cube	11/30	**17/30**

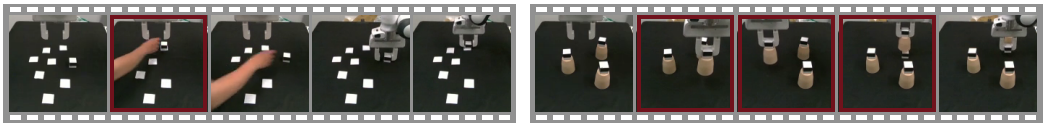


Figure 4: **Real-world experiments. Left:** A task similar to Placement Imitation in VMBench. A human demonstrator places the cube at a target location and then returns it to its initial location. The robot must place the cube at the same location. **Right:** A task similar to Explore Cube in VMBench. The robot must find the cube hidden under one of three upside-down cups. Checking the same cup twice is considered a failure.

where the target location is shown once and then removed from the scene. These results provide evidence that the benefit of MomentMem persists under real sensing and control dynamics.

5.6 Ablation Studies

Table 6: **Ablation of identifier training on VMBench.** We report closed-loop success rates under the standard appearance. Point Annotation replaces each key segment with one fixed positive frame. The w/o SS variant removes scheduled sampling and uses only teacher-forced memory histories. Full MomentMem uses both segment-level annotations and scheduled sampling.

	Explore Cube	Mine Sweeper	Hidden Pattern	Seal Checking	Spring Compression	Drawer Match	Flash Beacon	Scroll Discovery	Placement Imitation	Avg.
Point Annotation	0%	0%	25%	0%	30%	1%	0%	1%	5%	6.9%
w/o SS	0%	0%	**29%**	2%	**87%**	27%	0%	**23%**	3%	19.0%
Full MomentMem	**8%**	**85%**	**29%**	**73%**	81%	**34%**	**25%**	**23%**	**98%**	**50.7%**

Segment annotations prevent contradictory supervision. We replace each annotated segment with a single positive frame, following the point annotation used by prior work. As shown in Table 6, this change reduces average success by 43.8 percentage points. Point supervision treats neighboring frames as negatives simply because they are not the selected frame. These frames show the same event, so such supervision contradicts itself. The large degradation supports treating them as equivalent candidates through segment-level annotation.

Scheduled sampling prevents errors from accumulating across memory decisions. Removing sequence-level scheduled sampling reduces average success by 31.7 points. The degradation concentrates on tasks with multiple key segments, such as Mine Sweeper and Placement Imitation, which both fall to 3% or below. By contrast, Spring Compression has only one key segment and reaches 87%. Memory selection is sequential, so an early mistake changes the memory context used for every later decision. Scheduled sampling exposes the identifier to such imperfect histories during training, and therefore improves long-horizon selection.

6 Conclusion

In this work, we introduce MomentMem, a framework that selects keyframes from raw observations to preserve visual moment memory for robot manipulation. We also introduce VMBench, a benchmark of manipulation tasks that require recalling visual moments after they leave the field of view. On VMBench, MomentMem substantially outperforms the strongest memory-augmented baseline, and its identifier generalizes to unseen appearances without additional memory annotations. It also supports few-shot adaptation to a held-out manipulation task. We hope future work can train the identifier across diverse tasks, enabling adaptation to new tasks without any memory annotations.

Limitations. MomentMem runs a VLM identifier, which adds inference cost compared with a single-model policy. On RMBench, MomentMem matches rather than surpasses the strongest baseline, and the task-level results are mixed. MomentMem still struggles with texture matching on tasks in the Appearance category. Retaining relevant keyframes alone does not resolve this difficulty (Appendix F). VMBench also uses a single embodiment and a fixed camera configuration, so the effect of viewpoint changes remains untested.

AI USE STATEMENT

Generative AI tools, specifically Large Language Models (LLMs), were used to improve the clarity and linguistic quality of the manuscript and to assist with code implementation. No LLMs were used for research idea generation or experimental design. All AI-assisted code was carefully reviewed and tested by the authors to ensure correctness, and all AI-assisted text was manually checked and revised. The authors take full responsibility for the final manuscript and experimental results.

ETHICS STATEMENT

This work focuses on the memory issues of robot manipulation. Our evaluation is conducted on standard robotic manipulation benchmarks in simulation and on real-world robotic platforms under controlled experimental settings. The study does not involve human subjects, collection of personal or sensitive information, or the release of new datasets containing such information. Real-world experiments were carried out following standard laboratory safety and research integrity practices.

REPRODUCIBILITY STATEMENT

The backbone models and checkpoints used in this work are publicly available, and the experimental benchmarks are standardized and publicly accessible. We will release the implementation code in a public GitHub repository. Additional experimental results are provided in the appendix to further facilitate reproduction and verification of our findings.

REFERENCES

Vaidehi Bagaria, Bijo Sebastian, and Nirav Kumar Patel. Recursive belief vision language action models. *arXiv preprint arXiv:2602.20659*, 2026. URL https://arxiv.org/abs/2602.20659.

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

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Proceedings of the Conference on Robot Learning*, volume 305, pp. 17–40, 2025. URL https://proceedings.mlr.press/v305/black25a.html.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, Hongcheng Wang, 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:2603.01229v2*, 2026a. URL https://arxiv.org/abs/2603.01229v2.

Yipeng Chen, Wentao Tan, Lei Zhu, Fengling Li, Jingjing Li, Guoli Yang, and Heng Tao Shen. Non-markovian long-horizon robot manipulation via keyframe chaining. *arXiv preprint arXiv:2603.01465*, 2026b. URL https://arxiv.org/abs/2603.01465.

Egor Cherepanov, Nikita Kachaev, Alexey K. Kovalev, and Aleksandr I. Panov. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning. In *International Conference on Learning Representations*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1088fffae5f2aa8c8324e4f45f62248d-Abstract-Conference.html.

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*, 2026b. URL https://arxiv.org/abs/2606.12497.

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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 3407–3415, 2026. URL https://ojs.aaai.org/index.php/AAAI/article/view/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. In *International Conference on Machine Learning*, 2026. URL https://robomme.github.io/.

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. URL https://arxiv.org/abs/2501.18564.

Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini Robotics: Bringing AI into the physical world. *arXiv preprint arXiv:2503.20020*, 2025. URL https://arxiv.org/abs/2503.20020.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, et al. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, 2024. URL https://www.roboticsproceedings.org/rss20/p090.html.

GigaBrain Team, Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Haoyun Li, Jie Li, Jiagang Zhu, Lv Feng, et al. GigaBrain-0: A world model-powered vision-language-action model. *arXiv preprint arXiv:2510.19430*, 2025. URL https://arxiv.org/abs/2510.19430.

GigaBrain Team, Boyuan Wang, Bohan Li, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Jie Li, Jindi Lv, Jingyu Liu, et al. GigaBrain-0.5M*: a VLA that learns from world model-based reinforcement learning. *arXiv preprint arXiv:2602.12099*, 2026a. URL https://arxiv.org/abs/2602.12099.

GigaBrain Team, Angen Ye, Axiang Sun, Can Jin, Chenxi Cheng, Chong Shi, Dengke Shang, Dingqian Zhang, Guan Huang, Guangqiang Wang, et al. GigaBrain-0.7: Scaling embodied foundation models to emergent capabilities with a three-system architecture. *arXiv preprint arXiv:2608.15875*, 2026b. URL https://arxiv.org/abs/2608.15875.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. ManiSkill2: A unified benchmark for generalizable manipulation skills. *arXiv preprint arXiv:2302.04659*, 2023. URL https://arxiv.org/abs/2302.04659.

Qingda Hu, Ziheng Qiu, Zijun Xu, Kaizhao Zhang, Xizhou Bu, Zuolei Sun, Bo Zhang, Jieru Zhao, Zhongxue Gan, and Wenchao Ding. Resolving state ambiguity in robot manipulation via adaptive working memory recoding. *IEEE Robotics and Automation Letters*, 11(7):8439–8446, 2026a. doi: 10.1109/LRA.2026.3693565. URL https://doi.org/10.1109/LRA.2026.3693565.

Yanbin Hu, Jin Cui, Jiayi Lu, Ruixuan Yang, Jun Ye, Boran Zhao, Xingyu Chen, Xuguang Lan, and Pengju Ren. ECHO: Continuous hierarchical memory for vision-language-action models. *arXiv preprint arXiv:2605.10993*, 2026b. URL https://arxiv.org/abs/2605.10993.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, et al. OpenVLA: An open-source vision-language-action model. In *Proceedings of the Conference on Robot Learning*, volume 270, pp. 2679–2713, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

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. URL https://arxiv.org/abs/2605.10921.

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*, 2026. URL https://arxiv.org/abs/2603.12942.

Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. ManiSkill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021. URL https://arxiv.org/abs/2107.14483.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. URL https://arxiv.org/abs/2503.14734.

Rutav Shah, Rajat Kumar Jenamani, Xiaohan Zhang, Lingfeng Sun, Roberto Martín-Martín, Yuke Zhu, Deva Ramanan, and Karl Schmeckpeper. Scaling short-term memory of visuomotor policies for long-horizon tasks. *arXiv preprint arXiv:2606.16178*, 2026a. URL https://arxiv.org/abs/2606.16178.

Rutav Shah, Yisu Li, Femi Bello, Yuke Zhu, and Roberto Martín-Martín. Memory retrieval in visuomotor policies for long-horizon robot control. In *Proceedings of Robotics: Science and Systems*, 2026b. URL https://www.roboticsproceedings.org/rss22/p010.html.

Hao Shi, Weiye Li, Bin Xie, Yulin Wang, Renping Zhou, Tiancai Wang, Xiangyu Zhang, Ping Luo, and Gao Huang. MemoryVLA++: Temporal modeling via memory and imagination in vision-language-action models. *arXiv preprint arXiv:2606.09827*, 2026a. URL https://arxiv.org/abs/2606.09827.

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. In *International Conference on Learning Representations*, 2026b. URL https://arxiv.org/abs/2508.19236.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Scaling up memory for robotic control via experience retrieval. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/9da515b1ad19d032a7398f00f5ff9b0c-Abstract-Conference.html.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, et al. ManiSkill3: GPU parallelized robotics simulation and rendering for generalizable embodied AI. *arXiv preprint arXiv:2410.00425*, 2024. URL https://arxiv.org/abs/2410.00425.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, Karan Dhabalia, Michael Equi, Quan Vuong, Jost Tobias Springenberg, Sergey Levine, Chelsea Finn, and Danny Driess. MEM: Multi-scale embodied memory for vision language action models. *arXiv preprint arXiv:2603.03596*, 2026. URL https://arxiv.org/abs/2603.03596.

Ziye Wang, Modi Shi, Chaojun Ni, Jiazhi Yang, Mengdi Li, Zhizhong Su, Tianwei Lin, and Hongyang Li. NativeMEM: Native memory compression for long-horizon robotic manipulation. *arXiv preprint arXiv:2607.06678*, 2026. URL https://arxiv.org/abs/2607.06678.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. SAPIEN: A simulated part-based interactive environment. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11094–11104. IEEE, 2020. URL https://conferences.computer.org/cvpr/pdfs/CVPR2020-1XMljIyuXWg2zC9bnL19Tw/716800l1094/716800l1094.pdf.

Ganlin Yang, Zhangzheng Tu, Yuqiang Yang, Sitong Mao, Junyi Dong, Tianxing Chen, Jiaqi Peng, Jing Xiong, Jiafei Cao, Jifeng Dai, Wengang Zhou, Yao Mu, and Tai Wang. EventVLA: Event-driven visual evidence memory for long-horizon vision-language-action policies. *arXiv preprint arXiv:2606.20092*, 2026. URL https://arxiv.org/abs/2606.20092.

Yihan Zeng, Minghao Ye, Yiyuan Chen, Yide Shentu, Philipp Wu, Zike Yan, and Zhongyu Li. KEMO: Event-driven keyframe memory for long-horizon robot manipulation with VLA policies. *arXiv preprint arXiv:2606.23589*, 2026a. URL https://arxiv.org/abs/2606.23589.

Zijian Zeng, Fei Ding, Huiming Yang, and Xianwei Li. HELM: Harness-enhanced long-horizon memory for vision-language-action manipulation. *arXiv preprint arXiv:2604.18791*, 2026b. URL https://arxiv.org/abs/2604.18791.

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 *Proceedings of the Conference on Robot Learning*, volume 229, pp. 2165–2183, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A TASK-RELEVANT VISUAL EVIDENCE

Table 7 summarizes the evidence that each task in VMBench requires the robot to retain and use after it becomes unavailable. Each key segment marks a period when one piece of evidence is visible. We use privileged information from the simulator to annotate keyframe segment intervals, with the aim of approximating human annotation.

Table 7: Visual evidence and its later use in the nine tasks of VMBench. The final column gives the number of key segments per demonstration.

Task	Evidence to remember	Later use	Segments
Explore Cube	Contents beneath the three covers	Retrieve the hidden cube	3
Mine Sweeper	Detector colors and their locations	Place blocks at matching locations	2
Hidden Pattern	Box–pattern associations and a target pattern	Select the matching box	2
Seal Checking	Colors revealed beneath the seals	Place blocks at the corresponding locations	2
Spring Compression	A demonstrated compression position	Reproduce the compression	1
Drawer Match	Patterns inside the three drawers	Reopen the matching drawer	3
Flash Beacon	Two transient target locations	Move the cube to both locations in order	2
Scroll Discovery	Patterns revealed by unrolling the scrolls	Stack the matching scrolls	3
Placement Imitation	Demonstrated object–position associations	Reproduce the placements	2

B TASK INSTRUCTIONS

The following boxes list the task instructions for the nine tasks in VMBench. The instructions are reproduced verbatim from the environment definitions and describe the task procedure and objective; the instance-specific patterns, colors, and target locations must be inferred from visual observations. The identifier and executor receive the task instruction. The separate memory-instruction field stored in the dataset is not supplied to the identifier in our experiments.

Spring Compression

```
A white cube sits on a long horizontal rail connected to a spring.
Before the robot moves, the cube slides to a compressed position,
pauses, and returns to its starting position.  As soon as it returns,
close the gripper and push the cube at a steady speed to the same
compressed position, then lift the closed gripper while the cube stays
there.
```

Flash Beacon

```
A white cube and two beacons are on the table.  Lift the cube.  When
the first beacon appears, start moving toward it at a moderate speed
so that it disappears before you arrive.  Keep moving without
returning the arm to its starting pose, and place the cube at that
spot.  The second beacon appears as soon as the cube touches the
table.  Keep holding the cube and continue directly toward the second
beacon, then place and release the cube at that spot.
```

Explore Cube

```
Three identical opaque cups are spread across the table, and one cup
hides a green cube.  Inspect all three cups in any order.  Each time,
lift a cup high enough to see under it, then put it back upside down
in the same place before checking the next.  Once all three cups have
been checked, move the cup hiding the cube aside, then grasp and lift
the cube.  Leave the other cups in place and leave every cup upside
down on the table.
```

Hidden Pattern

```
Three boxes open together to show three different patterns, then
close.  A target card appears briefly with one of those patterns and
then disappears.  Then push forward the box that showed the same
pattern as the target card.
```

Drawer Match

```
A three-level cabinet hides a pattern in each drawer.  Press the
upper, middle, and bottom drawer buttons in that order; each drawer
opens once and then closes.  Then press the upper or middle button for
the drawer whose pattern matches the bottom drawer.  Finish when the
selected drawer is fully open.
```

Scroll Discovery

```
Three narrow scrolls hide patterned strips.  From left to right, pull
each strip until the whole pattern is visible, then release it so it
rolls back up.  After the third strip rolls up, immediately grasp one
of the two scrolls that showed the same pattern and stack it on top of
the other, without touching the scroll with the different pattern.
```

Placement Imitation

```
Two cubes of different colors start at separate positions on the
table.  One at a time, each cube moves to an unmarked target position,
pauses, and returns.  Once both demonstrations are complete, place
each cube at the position where it paused, then release it on the
table.
```

Mine Sweeper

```
Two parallel black lines, a white probe, and two cubes of different
colors are on the table.  Move the probe directly to one end of the
nearer line and scan once to the other end.  Move directly to the
farther line and scan it once in the opposite direction.  The probe
briefly changes color at one point on each line.  Then put the probe
back, return the arm to its starting pose, and place each cube where
the probe showed the same color.  Release each cube on the table.
```

Seal Checking

```
Two white stamps cover two colored spots.  A tray and two cubes of
different colors are also on the table.  Lift each stamp by its square
handle, place it fully inside the tray, and watch the spot underneath
fade away.  After both stamps have been checked, return the arm to its
starting pose.  Then place each cube where the spot showed the same
color and release it on the table.
```

C Dataset Construction

C.1 Simulation and Demonstration Collection

We construct VMBench in ManiSkill3 with SAPIEN physics, using a Panda manipulator, an overview camera, and a wrist camera. The data-generation pipeline records both RGB streams at 224×224 resolution and a control frequency of 20 Hz. Each task implements a scripted, closed-loop expert that computes actions from the simulator state and executes them through the robot controller. Demonstrations therefore contain the actions actually applied to the environment and the resulting observations.

Each rollout starts from a seeded initialization, with the scene rebuilt at reset. Task-specific randomization varies object positions, target locations, and the associations between visual cues and their required responses. For example, Hidden Pattern samples distinct patterns and assigns them to boxes; Drawer Match and Scroll Discovery randomize the matching patterns; and Placement Imitation, Mine Sweeper, and Seal Checking randomize color–location associations. Color pairs are sampled without replacement from a palette of eight colors, excluding pairs that are too similar under the implemented color-distance criterion. Spatial sampling also enforces task-specific separation and reachability constraints. These variations produce different memory contents while retaining the same task instruction.

The expert rolls out until success, failure, termination, or the step limit. Only successful episodes are retained; unsuccessful attempts are discarded and generation continues with new seeds. For tasks that expose a separate evaluation-success flag, a retained demonstration must satisfy that flag as well as the expert's success condition. The generator reports an error if it cannot collect the requested number of successful episodes. Our main experiment uses 100 demonstrations per task, giving 900 demonstrations across the nine tasks. Generation can be distributed across shards, with seed-spacing checks that account for retry ranges.

C.2 Automatic Memory Annotations

Each environment records the intervals during which task-relevant evidence is available, using simulator events and task state. Every interval is stored as an inclusive pair of frame indices $[u_j, v_j]$ at the original recording rate. The metadata also records the event identity, the information to retain, and the corresponding timestamps. These annotations identify an interval of valid evidence rather than prescribing one fixed representative frame.

For Hidden Pattern, the two intervals cover the revealed box patterns and the target card. Spring Compression marks the demonstrated compression position, while Placement Imitation marks the two demonstrated target positions. Drawer Match and Scroll Discovery mark the three pattern-reveal intervals. Explore Cube marks the three inspections beneath the cups; Mine Sweeper marks the two probe-color events; Flash Beacon marks the two visible beacons; and Seal Checking marks the two revealed colors before they become too faint to identify. Thus, each demonstration has one to three annotated segments, as summarized in Table 7. Privileged event metadata supplies supervision for memory selection and is not included in the identifier's or executor's observation input.

C.3 Observation and Action Alignment

Raw trajectories are saved in HDF5, with accompanying JSON metadata containing the episode seed, task instruction, phase intervals, and memory annotations. A rollout with T actions contains observations from o_0 through o_T. Conversion creates T training records, pairing each action a_t with its preceding observation o_t and timestamp. The terminal observation is not assigned an additional action. Both camera streams and the full-rate memory boundaries retain this frame indexing.

Hidden Pattern, Spring Compression, and Placement Imitation begin with an observation phase in which the environment presents a visual demonstration without requesting policy actions. These observations remain available for memory selection. For MomentMem, the recorded stationary actions during the observation and intervening memory phases are excluded from action supervision, and the executor is trained only on the execution phase. The remaining tasks require robot actions

to acquire the evidence, so their recorded actions remain eligible for supervision throughout the trajectory.

Our EventVLA implementation instead retains supervision on these stationary actions, because its keyframe predictions are derived from action hidden states. It therefore learns these actions even though the environment does not request them during observation.

C.4 Training-Data Conversion

We convert the recordings into LeRobot-format datasets with per-episode videos, tabular state–action records, task metadata, and a separate file of key-segment annotations. The converter checks camera streams, temporal alignment, and segment bounds, and rejects missing or malformed memory intervals. It reconstructs the controller's commanded targets from recorded simulator states to express end-effector actions in the overview-camera coordinate system. For the executor used in this paper, actions comprise relative end-effector translation, relative rotation represented in Rot6D, and a gripper command. Dataset conversion preserves the annotated intervals; representative-frame sampling is performed during training as described in Section 4.

C.5 Visual OOD Data and Training Splits

Visual OOD environments use the same task implementations and expert procedures, with altered room and table appearance, lighting, and additional background or tabletop distractors outside the manipulation area. Appearance randomization uses a separate seed-derived random-number stream, preserving the task randomization for a given seed. Task mechanics, memory events, and success conditions are shared with the standard setting, and OOD trajectories pass through the same recording and conversion pipeline. In the appearance-transfer experiment, target-domain executor training uses keyframes predicted by the transferred identifier rather than the simulator's target-domain memory annotations.

For the standard-setting identifier, we reserve 20 demonstrations per task for validation and train on the remaining 80. The executor uses all 100 demonstrations per task. The identifier validation set is used for checkpoint selection, while closed-loop policy evaluation uses unseen initial configurations, as described in Section 5.

D Implementation and Training Details

We implement the keyframe identifier with Qwen3-VL-4B (Bai et al., 2025) and the executor with $\pi_{0.5}$ (Black et al., 2025). On VMBench, the identifier receives only overview images. The executor receives current overview and wrist images, while its keyframe stack contains only overview images. The identifier receives neither privileged memory instructions nor absolute historical frame indices. VMBench operates at 20 Hz, and the identifier samples observations at 5 Hz. Table 8 summarizes the configuration used for the standard appearance.

For the main experiments on VMBench and RMBench, we reserve 20% of the trajectories with segment-level keyframe annotations for identifier validation and use the remaining 80% for training. On VMBench, this gives 80 training and 20 validation trajectories per task for the identifier; the executor uses all 100 trajectories. We select the identifier checkpoint with the highest mean of segment accuracy and non-keyframe accuracy on the validation set. These metrics are defined in Appendix E.

Table 8: Configuration of MomentMem on VMBench under the standard appearance.

Setting	Value
Control frequency	20 Hz
Maximum identifier training steps	10,000
Selected identifier checkpoint	Step 10,000
Identifier sampling rate (validation/inference)	5 Hz
Recent observation buffer	10 images
Positive decision threshold	0.5
Maximum model-feedback probability	0.4
Identifier focal loss (α, γ)	$(0.8, 2.5)$
Executor training steps	60,000
Executor action chunk size (training)	10 actions
Executor action chunk size (execution)	10 actions

Table 9 lists the RMBench configuration. Actions are executed at 16.67 Hz. For identifier validation and inference, we select one frame out of every three, giving a sampling rate of 5.56 Hz. Training uses temporal sampling augmentation, with frame intervals of 1, 2, or 3 sampled with probabilities 0.1, 0.1, and 0.8, respectively. During evaluation, the executor executes the first 40 actions of each predicted chunk before predicting the next chunk. For the RMBench tasks, we manually annotated keyframe segments for MomentMem. We did not use the subtask annotations provided by RMBench.

Table 9: Configuration of MomentMem on RMBench. The identifier checkpoint is selected using the validation criterion above.

Setting	Value
Control frequency	16.67 Hz
Maximum identifier training steps	20,000
Selected identifier checkpoint	Step 18,000
Identifier sampling rate (validation/inference)	5.56 Hz
Recent observation buffer	10 images
Positive decision threshold	0.5
Maximum model-feedback probability	0.9
Identifier focal loss (α, γ)	$(0.8, 2.5)$
Executor training steps	90,000
Executor action chunk size (training)	50 actions
Executor action chunk size (execution)	40 actions

The anchor labels described in Section 4.3 and Figure 2(b) assume a teacher-forced history in which the anchor is stored. Scheduled sampling instead requires labels to reflect the memory history shown to the identifier. Within a key segment, frames before the sampled anchor are masked from the loss until a representative is stored. From the anchor onward, the target is Yes until the visible history contains a representative of that segment; it then becomes No. Frames outside all segments receive

No. For example, if the identifier misses an anchor, a later frame in the same segment remains a positive example. This history-dependent supervision allows recovery from missed selections.

The executor retains only sampled anchors earlier than the current frame and masks unused memory slots. On VMBench, its actions contain relative end-effector translation, relative rotation represented in Rot6D, and a gripper command. Both translation and rotation are expressed in the overview-camera coordinate system. State and action normalization use training-data statistics. On RMBench, the executor instead uses the benchmark's dual-arm absolute joint-position and gripper actions. These action conventions are shared by training and evaluation.

For transfer across visual appearances or tasks, each frame selected offline by the identifier becomes a single-frame key segment. The executor therefore uses the same training procedure without privileged target annotations. Images, actions, and task instructions remain unchanged.

In the Flash Beacon few-shot experiment, only five target-task trajectories provide memory annotations for identifier adaptation. In both From Qwen and Ours, the adapted identifier labels all 100 target-task demonstrations, including those five trajectories. The executor is trained on all 100 demonstrations using these automatically generated memory labels. Appendix H gives the full comparison.

E KEYFRAME SELECTION QUALITY

On VMBench, the 20% validation split contains 20 demonstrations per task, or 180 in total. This is the checkpoint-selection set, not an independent test set. Evaluation replays each demonstration autoregressively: the identifier's previous decisions determine its later memory context. Validation uses no class balancing.

Segment hit rate measures the fraction of key segments containing at least one selected frame. Segment accuracy requires exactly one selection per segment. To measure selections outside the key segments, we also report the fraction of demonstrations with no such selection, termed non-keyframe accuracy. This trajectory-level metric avoids the dominance of easy negative frames in frame-level accuracy.

Table 10: **Identifier validation on VMBench under the standard appearance.** Hit rate and segment accuracy are computed over key segments; non-keyframe accuracy is computed over demonstrations. The pooled row aggregates counts across tasks.

Task	Segment hit rate (%)	Segment accuracy (%)	Non-keyframe accuracy (%)
Explore Cube	98.3	96.7	90.0
Mine Sweeper	100.0	100.0	100.0
Hidden Pattern	100.0	100.0	100.0
Seal Checking	97.5	95.0	80.0
Spring Compression	100.0	90.0	100.0
Drawer Match	100.0	100.0	100.0
Flash Beacon	97.5	87.5	80.0
Scroll Discovery	86.7	83.3	65.0
Placement Imitation	85.0	82.5	45.0
Pooled	**95.8**	**93.0**	**84.4**

As shown in Table 10, the identifier selects at least one frame in 383 of 400 key segments and exactly one in 372. It makes no selection outside the key segments in 152 of 180 demonstrations. These results assess memory selection on recorded demonstrations; closed-loop success additionally depends on the executor and the observations encountered during interaction.

We also evaluate the same identifier on unseen appearances without further fine-tuning. Table 11 reports autoregressive evaluation on all 100 demonstrations per task, or 900 in total, with the decision threshold fixed at 0.5. Unlike the standard-appearance validation split, these demonstrations are not used to select the identifier checkpoint.

Table 11: **Identifier validation on VMBench under unseen appearances.** The identifier is unchanged from Table 10. Results cover 100 demonstrations per task and use the same strict segment criteria. The pooled row aggregates counts across tasks.

Task	Segment hit rate (%)	Segment accuracy (%)	Non-keyframe accuracy (%)
Explore Cube	91.7	88.3	54.0
Mine Sweeper	99.5	99.5	100.0
Hidden Pattern	97.5	97.5	97.0
Seal Checking	98.0	98.0	88.0
Spring Compression	10.0	10.0	4.0
Drawer Match	95.7	94.3	89.0
Flash Beacon	90.0	79.0	61.0
Scroll Discovery	84.7	80.3	37.0
Placement Imitation	74.0	73.0	47.0
Pooled	**87.2**	**84.7**	**64.1**

Spring Compression has low strict segment scores, but 88 of its 90 missed demonstrations contain a selected frame only 0.10 s before the annotated start, less than the identifier's 0.20 s sampling period. Across all 90 misses, the selected frame nearest the segment differs from the segment's mean spring position by less than 0.4 mm. This is far below the task's 45 mm compression-position tolerance for successful execution. These keyframes therefore retain the relevant compression information despite falling outside the annotated interval, consistent with the high task success rate.

F Analysis of Memory Selection, Use, and Execution

The results in Table 2 show different performance patterns across the three task categories. MomentMem achieves its largest gains on tasks requiring appearance–location bindings, while tasks in the Appearance category remain challenging for all evaluated methods. To better understand these differences, we study Hidden Pattern, Flash Beacon, and Seal Checking as examples of the Appearance, Spatial, and Spatial & Appearance categories, respectively. We compare MomentMem with EventVLA, the baseline with the highest average success in the standard setting.

We examine whether relevant evidence is retained, whether it influences subsequent actions, and whether the required physical operations are completed. We also evaluate whether keyframe selection remains effective under visual appearance changes. All analyses use the final checkpoints on ten fixed test scenes per task, without further training.

F.1 Counterfactual Keyframe Content

Protocol. We use the same scene seeds for both methods. Each scene is assigned original and counterfactual conditions. The counterfactual branch runs only when the intervention point is reached and changes cue content in eligible, natively selected keyframes. Scenes without an intervention point or an eligible selected frame remain in the counts. For Seal Checking, we exchange the two colors of the seal in the stored keyframes. For Flash Beacon, we move the remembered first beacon to the scene's second target location, and evaluate only the response to the first beacon. For Hidden Pattern, we replace the remembered target-card pattern with the pattern in another box.

In Table 12, *Reached* counts scenes that reach the intervention point, and *Modified* counts scenes whose selected keyframes are changed, both out of ten scenes. *Original* reports successful completion under the original condition out of all ten scenes. *Changed* counts modified scenes in which the observed target choice changes to the target specified by the counterfactual cue. An unchanged choice that already matches this target does not count; successful task completion is not required. For Seal Checking, we count scenes where at least one cube switches to its counterfactual placement target. For Flash Beacon, we compare the nearest target at first contact or first-cue completion; this is a spatial proxy for target choice. For Hidden Pattern, we compare the selected boxes. Reached scenes remain unmodified if no selected keyframe falls within the cue interval eligible for editing.

Table 12: **Target changes after keyframe interventions.** *Reached*, *Modified*, and *Original* use all ten scenes as the denominator. *Changed* counts switches to the counterfactual target among modified scenes, without requiring successful task completion.

Task	Method	Reached	Modified	Original	Changed
Seal Checking	MomentMem	7/10	7/10	6/10	**7/7**
	EventVLA	0/10	0/10	0/10	–
Flash Beacon	MomentMem	10/10	8/10	4/10	**6/8**
	EventVLA	10/10	7/10	2/10	3/7
Hidden Pattern	MomentMem	10/10	10/10	3/10	0/10
	EventVLA	10/10	10/10	0/10	**1/10**

Color and spatial evidence. On Seal Checking, exchanging the stored colors changes MomentMem's placements to match the new associations (Table 12). Six scene pairs succeed under both their original and counterfactual conditions. This shows that the remembered color–location bindings influence placement decisions. EventVLA does not reach the intervention point in these runs, so we examine it separately below.

On Flash Beacon, both methods change the approached target in some pairs. All observed MomentMem switches follow the counterfactual cue, while EventVLA also exhibits a switch in the opposite direction. MomentMem additionally shows a larger shift toward the alternative target at matched post-intervention times, measured by the cube's relative distance to the two targets. These responses indicate sensitivity to stored spatial cues, even when the subsequent manipulation fails.

Texture matching. On Hidden Pattern, selected keyframes cover both the box-pattern and target-card events in all ten MomentMem scenes and eight EventVLA scenes. Among these scenes, MomentMem never switches to the counterfactual target. EventVLA does so only once, without completing the push. Thus, the difficulty cannot be explained solely by missing frames from the relevant events. Both methods struggle to use retained texture information for target selection.

F.2 Evidence Acquisition and Physical Execution

Seal Checking. Most EventVLA runs reveal both cues but fail to return to the required home position before placement. MomentMem also has failures at this stage. These failures occur before the placement stage and do not establish whether the stored associations would have been used correctly.

To examine placement after successful evidence acquisition, we give both methods the same expert acquisition trajectory. Each method selects its own keyframes and resumes control with a fresh plan after the cues disappear. MomentMem completes both the original and counterfactual placements in seven of the ten scene pairs. EventVLA completes neither condition, with failures including unsuccessful grasps, drops, and releases at incorrect locations. Providing a successful acquisition trajectory therefore does not remove all downstream difficulties. These diagnostic results are separate from the benchmark scores and do not isolate memory use from control errors.

Flash Beacon. Some MomentMem runs reach the correct target region but fail to satisfy the grasp requirement. Both methods also make contacts away from the targets. Thus, a response to the remembered cue does not always lead to successful execution. Conversely, an incorrect contact location alone cannot distinguish an incorrect target decision from a control error.

F.3 Keyframe Identifier Transfer under Visual Changes

To test whether visual changes disrupt memory selection, we evaluate both frozen keyframe identifiers on matched standard and OOD expert trajectories. Actions and physical states are held fixed, and each method selects keyframes using its native procedure. We measure whether the selected frames cover both annotated cue events in each scene. This comparison evaluates selection independently of the policy's ability to execute the task.

Table 13: **Keyframe identifier transfer under visual changes.** Scenes covering both cue events, out of ten per condition.

Method	Standard	OOD
Seal Checking		
MomentMem	**8/10**	**8/10**
EventVLA	6/10	7/10
Flash Beacon		
MomentMem	**8/10**	**9/10**
EventVLA	7/10	3/10

MomentMem maintains coverage under the tested visual changes (Table 13). EventVLA remains effective on Seal Checking but loses coverage on Flash Beacon. The results support the robustness of MomentMem's keyframe identifier in these settings, while showing that selection can still miss relevant events. They concern keyframe identifier transfer rather than OOD action execution.

Discussion. We examine three possible sources of the performance gap: keyframe selection, memory use, and physical execution. MomentMem adjusts its actions to edited color and spatial cues. Its keyframe identifier also maintains cue coverage under the tested visual changes. EventVLA misses some cue events and responds less consistently to edited spatial cues in our sample. Its failures also include incomplete evidence acquisition and unsuccessful manipulation. These results point to difficulties at several stages, but do not isolate how much each stage contributes to the gap. On Hidden Pattern, both methods rarely follow the edited texture even when keyframes cover both relevant events. This indicates a shared difficulty in using retained texture information to guide actions.

G Real-World Experiments

G.1 Experimental Setup

We evaluate MomentMem and $\pi_{0.5}$ on a Franka Panda robot in two tasks adapted from VMBench, retaining their simulation names: Placement Imitation and Explore Cube. Both methods use only a single overview camera, with no wrist camera. We collect 51 and 54 demonstrations, respectively, and manually annotate their key segments because privileged simulator information is unavailable in real-world experiments. Each method is evaluated over 30 trials per task.

In Placement Imitation, seven cards are scattered across the tabletop. A human first places a cube on one card, then moves it in front of the robot. The robot must remember the demonstrated location and return the cube to the correct card within 60 seconds.

In Explore Cube, a cube is hidden under one of three inverted cups. The robot must inspect the cups, locate the cube, and grasp and lift it within 120 seconds. Revisiting a previously inspected cup or knocking over any cup counts as a failure.

For both tasks, a force-triggered protective stop also counts as a failure. We stop the robot when the Euclidean norm of the three external-force components at the end effector, estimated by the Franka controller, exceeds 100 N. This can occur when the robot presses into the tabletop or an object.

To reduce slippage, we wrap the cube in sandpaper and attach sandpaper-covered blocks to the cups as grasping surfaces. These modifications improve grasp stability without changing the memory requirements: recalling the target card or tracking previously inspected cups. Both MomentMem and $\pi_{0.5}$ use the same modified objects.

G.2 Training and Inference

Table 14 lists the hyperparameters. Inference runs on a remote server with an NVIDIA H20 GPU. The nominal control and identifier sampling rates are 15 Hz and 5 Hz, respectively. Observations captured while awaiting an action response are discarded, and this waiting interval does not advance the policy's time index. Network and inference delays therefore reduce the effective wall-clock rates.

The 60-second and 120-second task limits are measured in wall-clock time and include all request waiting time. Network fluctuations thus allow different numbers of executed actions across trials. Nevertheless, the limits were sufficient for the successful trajectories we observed to complete.

Table 14: Configuration of MomentMem in the real-world experiments. Frequencies refer to the logical timeline, excluding action-request latency.

Setting	Value
Control frequency	15 Hz
Maximum identifier training steps	8,000
Selected identifier checkpoint	Step 8,000
Identifier sampling rate (validation/inference)	5 Hz
Recent observation buffer	10 images
Positive decision threshold	0.5
Maximum model-feedback probability	0.8
Identifier focal loss (α, γ)	$(0.8, 2.5)$
Executor training steps	20,000
Executor action chunk size (training)	10 actions
Executor action chunk size (execution)	10 actions

G.3 Failure Analysis

In Explore Cube, $\pi_{0.5}$ repeatedly inspected the same cup or returned to a previously inspected cup. These failures are consistent with difficulty tracking exploration history without memory. We also observed incomplete inspections: the robot sometimes lifted the cup hiding the cube only slightly, then lowered it before the cube became visible. This behavior alone does not distinguish a memory failure from an execution error.

Despite these failures, $\pi_{0.5}$ descended steadily when approaching a cup to grasp it. We did not observe the vertical oscillation initially expected from ambiguity between descending to grasp and ascending to move on. Action chunks may preserve local motion continuity, and motion blur may provide directional cues across chunk boundaries. These explanations remain unverified, but the observations show that locally coherent motion can coexist with failures to track exploration history.

In our observed trials, once MomentMem saw the cube, it consistently moved the covering cup aside and grasped the cube accurately. By contrast, $\pi_{0.5}$ sometimes pressed into the cube and triggered the protective stop, or closed its gripper beside the cube. State aliasing during memory-free training may impair precise action prediction by associating similar observations with different actions, but these observations do not establish the cause. The absence of a wrist camera may also contribute to grasping errors, although it is shared by both methods and does not by itself explain their difference.

In Placement Imitation, $\pi_{0.5}$ placed the cube at an incorrect location after the demonstrated target was no longer visible. MomentMem also made placement errors, but achieved higher success than $\pi_{0.5}$.

H FEW-SHOT CROSS-TASK MEMORY TRANSFER

We test whether learned memory selection adapts to an unseen task from few memory annotations. Flash Beacon is the target task and is excluded from the eight source tasks. The target task provides 100 action demonstrations, but only five provide memory annotations in the few-shot settings. We evaluate source-to-target adaptation against two controls: full target memory supervision and target-only adaptation from the pretrained backbone. All six variants of MomentMem and EventVLA are evaluated on the same 300 environment seeds.

Training settings. For MomentMem, source-to-target adaptation first trains the identifier on the other eight tasks, then fine-tunes it using five annotated Flash Beacon trajectories (Ours in Figure 3). The adapted identifier labels all 100 target demonstrations offline, including the five used for adaptation. The executor is trained on these demonstrations using the predicted keyframes. Few-shot therefore concerns memory annotations, not the amount of action training data.

Full Memory Supervision is the positive control: memory selection is trained with all available target memory annotations, and the executor uses the target demonstrations with their provided annotations. Both components are trained only on Flash Beacon, without source-task training. This reference indicates the success achievable with full target memory supervision, without benefits or interference from other tasks. It provides context for judging how effective few-shot transfer is.

From Qwen is the target-only control. It starts from pretrained Qwen and fine-tunes the identifier on the same five target trajectories, without source-task training. It then uses the same automatic-labeling and executor-training procedure as Ours. This control tests whether few-shot success can be explained by the pretrained backbone's prior knowledge alone, rather than transferable memory selection learned on the source tasks.

We apply the same three supervision settings to EventVLA. Target-only adaptation omits source-task training, while source-to-target adaptation first trains memory selection on the eight source tasks. Both few-shot settings use the same five annotated target trajectories and generate memory labels for target-task action training. The positive control uses all target memory annotations without source-task training. We retain each method's architecture and use the same target action demonstrations across settings.

Table 15: **Few-shot cross-task transfer of memory selection.** Flash Beacon success rates (%) on 300 shared evaluation seeds. Full Memory Supervision provides a task-specific reference, while Target only controls for adaptation from the pretrained backbone without source-task training. Both few-shot settings use five annotated target trajectories; all settings use 100 target action demonstrations.

Memory supervision	MomentMem (%)	EventVLA (%)
Target only (From Qwen for MomentMem)	29.0	5.7
Source→Target (Ours for MomentMem)	43.7	5.0
Full Memory Supervision	44.3	7.7

Results. With five annotated target trajectories, transferred MomentMem reaches 43.7% success, close to its 44.3% full-supervision reference (Table 15). From Qwen reaches 29.0%, 14.7 points lower. The first comparison shows that few-shot transfer approaches a strong task-specific reference; the second suggests that this ability is not explained by pretrained Qwen alone. Together, they support few-shot cross-task transfer of learned memory selection on Flash Beacon.

EventVLA reaches 5.0% with source-to-target adaptation, compared with 5.7% for target-only adaptation and 7.7% under full memory supervision. These runs show no observed transfer benefit for EventVLA. Its low positive-control success also indicates difficulty on Flash Beacon beyond few-shot adaptation, so its low transfer success cannot be attributed to transfer alone.