SIMPLEMEMVLA: A SIMPLE BUT EFFECTIVE NATIVE-VIDEO MEMORY FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon manipulation is partially observable: the information needed to choose the next action may appear only in observations from minutes earlier. Existing memory mechanisms for VLAs, such as retrieval banks, learned compressors and recurrent states, must decide what to keep from the past before knowing what a future decision will require. They were motivated by the assumption that minute-scale history is too large to process directly, which no longer holds for modern VLM backbones. We propose SimpleMemVLA, a VLA without a dedicated memory module that uses the backbone's native video context directly as memory. It keeps the sampled history intact in the timestamped video format the backbone was pretrained to process, routes the evidence it finds to a standard flow-matching action head through the hidden states of a generated sub-task, and prefills the history shared by consecutive decisions during action execution, keeping latency close to that of a single-frame VLA. SimpleMemVLA achieves state-of-the-art results on four memory benchmarks without loss on general-purpose control, and with the same backbone and training setup it outperforms retrieval, compression and recurrent-state methods by a wide margin. History interventions show that the policy reads specific evidence from its past and follows edited histories without parameter updates, a visual form of in-context learning. On a physical dual-arm robot, it completes two tasks whose decisive evidence disappears before the robot acts. [1]

1 INTRODUCTION

As vision-language-action (VLA) models move from short tabletop skills to long-horizon tasks, partial observability becomes unavoidable (Kaelbling et al., 1998): the information needed to choose the next action—which object was revealed, where an occluded target was placed, how many times an action has already been completed—may appear only in observations from minutes earlier (Ma et al., 2024; Sapkota et al., 2025; Shi et al., 2026a; Koo et al., 2025). Most general-purpose VLAs, however, condition their actions on a single image or a sub-second observation window (Brohan et al., 2023; Team et al., 2024; Kim et al., 2024; Black et al., 2025; Intelligence et al., 2025; Liu et al., 2024b; Nvidia et al., 2025), and cannot solve such tasks no matter how well they are trained, since two states with identical current observations may require different actions.

Existing work provides memory through dedicated mechanisms (Figure 1): retrieval banks that select observations from an external store (Li et al., 2025; Sridhar et al., 2026; Hu et al., 2026; Lin et al., 2025a; Lei et al., 2025), learned compressors that summarize history within a fixed token budget (Shi et al., 2026a; Jang et al., 2025; Wang et al., 2026b; Koo et al., 2025), and recurrent states that continually update a compact representation of the history (Li et al., 2026; Cherepanov et al., 2026; Liu et al., 2024a; Qu et al., 2026). Although these mechanisms differ in implementation, each must determine what remains available from the history before the needs of a future decision are known: a relevant frame may be omitted during retrieval, visual details lost during compression, or earlier evidence overwritten by a recurrent update. We refer to this as *write-time commitment*.

[1] Code available at `https://anonymous.4open.science/r/SimpleMemVLA`

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.

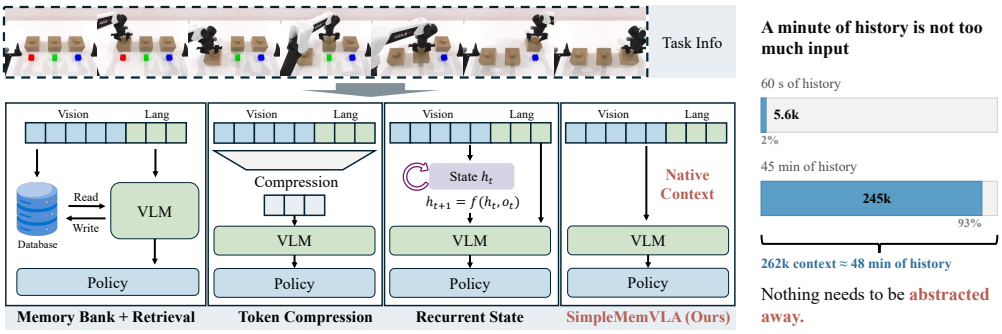


Figure 1: **VLA memory mechanisms versus SimpleMemVLA.** *Left:* Three prior design families insert dedicated memory machinery between the observation stream and policy, while SimpleMemVLA uses the timestamped stream directly as native context. *Right:* A 60 s history uses only 5.6k tokens of the backbone's 262k-token context window, which can hold roughly 45 minutes.

These mechanisms were motivated by the assumption that minute-scale history is too large to process directly, but that assumption no longer holds: modern VLM backbones are pretrained to read temporally ordered video through native multimodal interfaces (Wang et al., 2024; Bai et al., 2025b; Yang et al., 2025; Bai et al., 2025a), and at the sampling rates used for manipulation a 60 s history occupies roughly 5.6k tokens of a 262k-token context window. The open question is therefore no longer capacity, but whether a policy can actually make use of a history it is simply given.

We propose **SimpleMemVLA**, a VLA without a dedicated memory module that uses the pretrained backbone's native video context directly as memory: the policy looks at what has already happened in order to decide what to do now. We keep the sampled history intact until the current decision and present it in the timestamped video format the backbone was pretrained to process (Bai et al., 2025a), so the backbone identifies the relevant evidence at the moment of the decision rather than relying on an earlier choice about what to retain. This raises two questions. First, how does the evidence found in a long history drive the action head, which conditions on a short token sequence rather than thousands of visual tokens? We route it through a single narrow channel: the backbone generates the current textual sub-task, and the contextual hidden states and token embeddings of that span are the only path from history to a standard flow-matching action head (Lipman et al., 2022; Black et al., 2025; Intelligence et al., 2025; Nvidia et al., 2025). Second, can a minute-scale prompt be processed within the real-time budget of a control loop? Consecutive decisions share most of their visual history, so we prefill the shared prefix during action execution, bringing decision latency to 0.68 s, close to a single-frame VLA, with outputs identical to full recomputation.

SimpleMemVLA sets a new state of the art on all four memory benchmarks while remaining on par with the strongest reactive VLAs on general-purpose control. Re-implementing retrieval, compression and recurrent-state methods on the same backbone and training setup, native context reaches 88.3% on RoboMME against 31.5% for the strongest of the three, and even a symbolic pipeline given ground-truth perception reaches only 84.1%, suggesting that what limits these mechanisms is write-time commitment rather than the accuracy of what they commit. History interventions confirm that the policy reads specific past evidence: masking one completed pick-and-place event lowers its inferred count by exactly one, and replacing that evidence with video from another rollout redirects the policy without any parameter update, a visual form of in-context learning (Brown et al., 2020; Alayrac et al., 2022). On a physical dual-arm robot, it succeeds in 58.3% and 70.0% of trials on two tasks whose decisive evidence disappears before the robot acts.

2 Related Work

Vision-Language-Action Models Generalist VLA research largely focuses on mapping the observations available at the current decision to actions. Action generation ranges from discretized actions and specialized tokenizers (Brohan et al., 2023; Kim et al., 2024; Hung et al., 2025; Pertsch et al., 2025; Kim et al., 2025) to continuous diffusion and flow-matching experts (Chi et al., 2023; Lipman et al., 2022; Team et al., 2024; Liu et al., 2024b; Black et al., 2025). Architectural advances

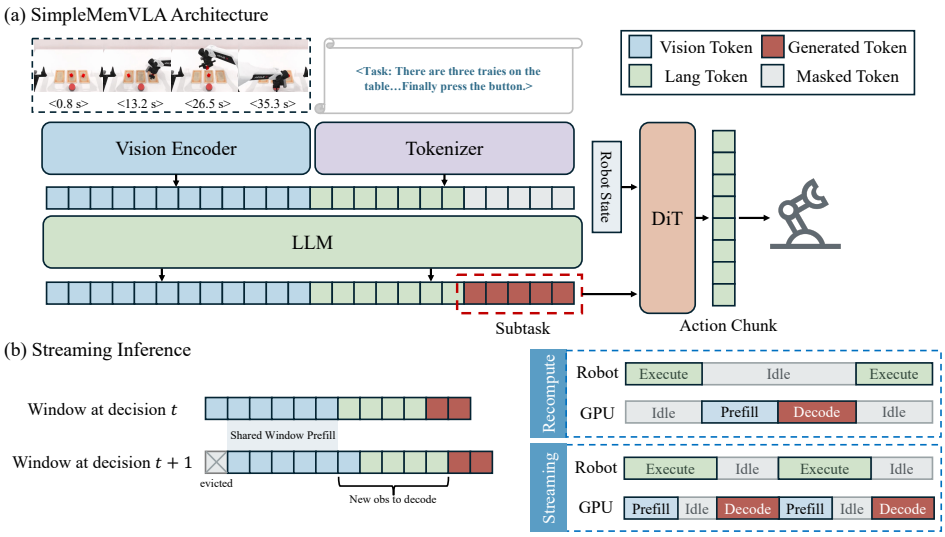


Figure 2: **SimpleMemVLA architecture and streaming inference.** *(a)* The architecture uses only standard VLA components, with self-attention over plaintext-timestamped history serving as memory. *(b)* Consecutive decisions differ by only one temporal patch, enabling shared-prefix prefill during action execution and reducing latency from 1.02 s to 0.68 s with identical outputs (Section 4.6).

include hierarchical control (Nvidia et al., 2025; Intelligence et al., 2025), spatial and trace representations (Qu et al., 2025; Zheng et al., 2025), video-pretrained world models (Cheang et al., 2024; Cen et al., 2025; Guo et al., 2025), reasoning and interactive post-training (Yin et al., 2026; Tan et al., 2025), and cross-embodiment transfer (Zheng et al., 2026; Bu et al., 2025). These advances improve control, but leave open how policies should represent and access minute-scale execution history (Ma et al., 2024; Sapkota et al., 2025). SimpleMemVLA addresses this question through the backbone's native video context.

Memory Mechanisms for VLAs Long-horizon, partially observable tasks require evidence from earlier observations (Kaelbling et al., 1998; Ma et al., 2024; Sapkota et al., 2025; Shi et al., 2026a; Koo et al., 2025; Li et al., 2026). Existing methods differ in how they expose this history to the policy. *Symbolic* pipelines store scene graphs, concept banks or execution states in predefined schemas (Dai et al., 2026; Sun et al., 2026a; Huang et al., 2026; Lei et al., 2025), discarding information outside their vocabulary even with perfect perception. *Retrieval* exposes selected experiences or event evidence (Sridhar et al., 2026; Yang et al., 2026a), leaving unselected observations unavailable at that decision. Fixed frame sampling provides another restricted selection of history (Lin et al., 2026). *Compression* maps history into bounded learned representations (Shi et al., 2026a; Jang et al., 2025; Wang et al., 2026b), choosing retained details before their future relevance is known. *Recurrent* methods continually update bounded states (Cherepanov et al., 2026; Qu et al., 2026; Gao et al., 2026), from which overwritten evidence cannot be recovered. SimpleMemVLA avoids such *write-time commitment* by presenting sampled timestamped history as native video context, letting the backbone's attention select relevant evidence at each decision. MEM (Torne et al., 2026) combines compressed short-term visual context with long-term language summaries. In streaming video understanding, SimpleStream (Shen et al., 2026) finds that an off-the-shelf VLM with a sliding frame window matches or outperforms dedicated memory methods.

3 SimpleMemVLA

3.1 Problem Setup and Overview

We consider language-conditioned manipulation under partial observability. At step t, given an instruction ℓ, the robot receives an observation o_t consisting of camera images and a proprioceptive

state q_t, and selects an action a_t. We denote the observation history together with the instruction by $h_t = (o_{\le t}, \ell)$ and consider a history-conditioned policy $\pi(a_t \mid h_t)$.

SimpleMemVLA uses the backbone's native video context directly as memory (Figure 2). A window of sampled head-camera frames enters the Qwen3.5-4B backbone through its video channel, while current wrist views enter as images. The backbone generates a one-sentence description of the current sub-task, g_t. The contextual hidden states and token embeddings of this span form the only channel through which visual history reaches the flow-matching action head, which also receives current proprioception and predicts an action chunk. We first describe how sampled history is presented as native video context (Section 3.2). We then explain how the resulting sub-task representations drive action generation (Section 3.3) and how prefix prefill reduces decision-time latency (Section 3.4).

3.2 Native Video History as Context

SimpleMemVLA represents the retained visual history as temporally ordered video with plaintext timestamps, using the backbone's native video interface (Bai et al., 2025a). Let f_c be the native frame rate of the observation stream and o_t^{h} the head-camera frame at step t. Rather than maintaining a separate learned memory state, SimpleMemVLA constructs, at every prediction step, a window covering the last T_w seconds subsampled at a rate $f_v \ll f_c$ into at most $K = T_w f_v$ frames,

$$V_t \;=\; \big(o^{\mathrm{h}}_{t-(K-1)s},\, \ldots,\, o^{\mathrm{h}}_{t-s},\, o^{\mathrm{h}}_{t}\big), \qquad s = f_c/f_v, \tag{1}$$

where s is the subsampling stride and an episode younger than T_w simply yields a shorter clip. Per suite, T_w is set to cover the horizon over which its tasks leave evidence and f_v is the lowest rate that does not skip decisive events, trading token budget against coverage (Table 8). V_t enters the backbone through its *video* channel, whose processor groups adjacent frames into temporal patches and prefixes each patch with the backbone's native plaintext timestamp, exactly as in video pretraining (Bai et al., 2025a). Standard deployment uses window-relative timestamps, labeling each patch by its offset within the active window, whereas the exploratory SWA variant in Appendix G uses episode-absolute timestamps to keep cached patches immutable during future continual operation over unbounded input streams. These timestamps provide an explicit temporal reference by encoding each patch's position within the window in a format the backbone already understands, allowing it to locate observed events in time relative to the current decision.

The current wrist frames $\{o_t^{\mathrm{w},i}\}_{i=1}^{W}$, where W is the number of wrist cameras, enter through the *image* channel without timestamps, under a single modality rule: multi-frame cameras become video and single-frame cameras become images. The prompt builder Φ combines the history video, current wrist images and language instruction as follows:

$$x_t \;=\; \Phi\big(V_t,\, \{o_t^{\mathrm{w},i}\}_{i=1}^{W},\, \ell\big). \tag{2}$$

The prompt also includes a plain-text description of the embodiment, camera layout and timestamp convention. The next section describes how evidence read from this context reaches the action head.

3.3 Action Generation from Sub-task Hidden States

This section describes how information selected from the visual history reaches the action expert. The expert accepts only a short token sequence rather than the thousands of visual tokens in the history window, so the backbone must distill the relevant information into a compact conditioning signal. SimpleMemVLA uses the generated sub-task span as this interface, yielding a signal that is compact, directly inspectable and editable. The backbone f_θ, with parameters θ, generates this span, while the DiT-style flow-matching expert v_ϕ, with parameters ϕ, conditions on its contextual representation and the current proprioceptive state. We define the generated sub-task, conditioning set and conditional flow-matching objective (Lipman et al., 2022) as:

$$\begin{aligned} g_t \;&=\; (g_{t,1}, \ldots, g_{t,m}) \;\sim\; f_\theta(\,\cdot \mid x_t), \\ C_t \;&=\; \big[\, e(g_{t,1}) \oplus h(g_{t,1}),\; \ldots,\; e(g_{t,m}) \oplus h(g_{t,m}),\; \psi(\bar{q}_t) \,\big], \\ \mathcal{L}_{\mathrm{act}} \;&=\; \mathbb{E}_{\tau,\,\varepsilon}\, \big\| v_\phi\big(A^\tau, \tau \mid C_t\big) - \big(\varepsilon - \bar{A}_t\big) \big\|_2^2. \end{aligned} \tag{3}$$

Here, g_t is a one-sentence description of the robot's current sub-task, generated as an ordinary assistant response under an unmodified chat template rather than as chain-of-thought. The function

$h(\cdot)$ returns the backbone hidden states over this response, $e(\cdot)$ its corresponding token embeddings, $\oplus$ denotes their fusion and $\psi(\bar{q}_t)$ is a single-token encoding of the normalized current proprioceptive state. In the objective, $\bar{A}_t \in \mathbb{R}^{H \times d_a}$ denotes the normalized action chunk for the next H steps, with each of its d_a action dimensions z-scored using dataset statistics. We sample noise $\varepsilon \sim \mathcal{N}(0, I)$ and a flow time $\tau \in [0, 1]$ from a distribution biased toward the noise endpoint, then construct the linear path $A^\tau = (1 - \tau)\,\bar{A}_t + \tau\,\varepsilon$. Because the action expert receives no prompt tokens directly, all history-dependent information needed for control must reach it through the sub-task span.

Training uses demonstrated action chunks and annotated sub-tasks g_t^*. The sub-task labels are generated offline by a cloud VLM, which is given each demonstration and describes the sub-task underway at every supervision anchor (Appendix B). During training, the backbone processes the annotated sub-task under teacher forcing, and the action-conditioning sequence C_t is constructed by substituting g_t^* for g_t in Equation 3. The same annotations supervise all three mechanism variants in Section 4.4, keeping sub-task supervision fixed in the comparison. The joint objective is

$$\mathcal{L} = \lambda_{\text{sub}} \mathcal{L}_{\text{sub}} + \lambda_{\text{act}} \mathcal{L}_{\text{act}}, \tag{4}$$

where $\mathcal{L}_{\text{sub}}$ is the token-level cross-entropy over the annotated answer span.

At deployment, the backbone instead generates g_t autoregressively from x_t, and the action expert uses its representations through the same conditioning rule. In both stages, the sub-task hidden states are contextualized by the visual history and can carry task information beyond the visible wording. Our interface ablations show that history-dependent control is carried primarily by these hidden states, while token embeddings support stable execution (Section 4.7).

The expert initializes an action chunk from Gaussian noise and integrates the learned velocity field toward a clean chunk using a small number of Euler steps. The resulting chunk is denormalized and its first n_e actions are executed. Observations continue to be buffered at the native control rate during execution, allowing the window in Equation 1 to be reconstructed exactly for the next prediction.

3.4 Streaming Inference with Prefix Prefill

The final requirement is deployment efficiency: retaining minute-scale history should not place the cost of reprocessing the entire window on the critical path of every decision. Consecutive decisions share nearly the entire video-history prefix, so SimpleMemVLA prefills this shared prefix while the robot executes the current action chunk and stores the resulting key–value cache. At the next decision, the policy processes only the newly arrived temporal patch and the text instruction before decoding the next sub-task and action chunk. Overlapping history processing with action execution reduces decision-time latency without changing the policy output (Section 4.6).

The current scheme makes bounded, minute-scale native context practical for deployment. As a step toward continual inference over native video streams of unbounded duration, Appendix G explores a variant trained with sliding-window attention (SWA), which keeps its active context and cache bounded as the history grows. Neither the native-context memory nor its streaming implementation depends on a particular robot. Within the architecture, embodiment-specific choices enter only through the configuration tuple $(\mathcal{C}_{\text{hist}}, \mathcal{C}_{\text{cur}}, T_w, f_v, H, d_a)$, which specifies the history and current camera sets, window length, sampling rate, action horizon and action dimensionality. Moving between bimanual and single-arm platforms therefore changes this configuration rather than the memory mechanism. Appendix B lists the concrete configuration used for each benchmark.

4 Experiments

We organize our experiments around five questions. First, **how effective is SimpleMemVLA?** We evaluate it on four memory-centric and two general-purpose benchmarks against published baselines (Section 4.2) and on two history-dependent tasks using a physical dual-arm robot (Section 4.3). Second, **does the gain come from the memory interface itself?** Holding everything else fixed, we rebuild one method from each mechanism family on our stack and compare them with native video context (Section 4.4). Third, **does the policy actually use its visual history as memory?** Holding the current observation and policy fixed, we remove or counterfactually replace the evidence in earlier frames and measure whether the output changes (Section 4.5). Fourth, **can long-context**

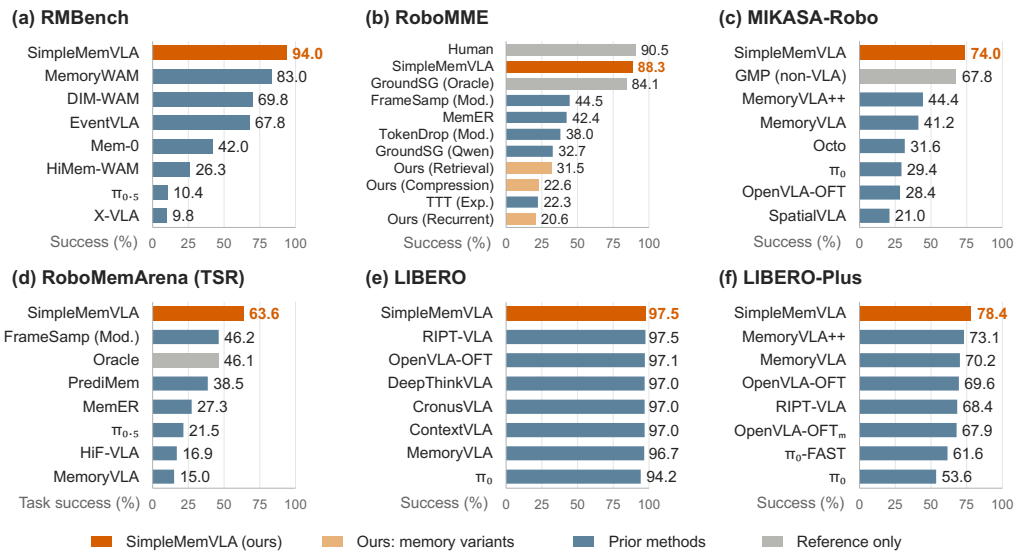


Figure 3: **SimpleMemVLA leads the four memory benchmarks and remains competitive on general-purpose control.** Full results appear in Appendix A (Tables 2–7). Light-orange bars denote our controlled memory variants. Gray bars denote human, oracle and non-VLA references excluded from ranking.

memory be deployed efficiently? We evaluate streaming inference, which reuses the shared history across consecutive decisions instead of recomputing it (Section 4.6). Finally, **how does each element of the recipe contribute?** We ablate history length, frame order, plaintext timestamps and the hidden-state and token-embedding inputs to the action head (Section 4.7).

4.1 Experimental Setup

We evaluate SimpleMemVLA on four memory-centric benchmarks—RMBench (Chen et al., 2026), RoboMME (Dai et al., 2026), MIKASA-Robo (Cherepanov et al., 2025) and RoboMemArena (Lei et al., 2026)—alongside LIBERO (Liu et al., 2023) and LIBERO-Plus (Fei et al., 2025) for general-purpose manipulation and zero-shot robustness, and two real-world tasks (Section 4.3). We train one model per simulation training suite, shared across its tasks. LIBERO-Plus reuses the LIBERO model without further training. RMBench averages nine shared tasks, excluding Place-Mat. Most baselines use task-specific models, whereas ours uses one multi-task model. Main-text simulation results use closed-loop evaluation on held-out seeds and streaming inference (Section 4.6). Benchmark descriptions, evaluation protocols and baseline provenance are provided in Appendix B, with the SWA variant evaluated separately in Appendix G.

4.2 Effectiveness Across Benchmarks

Leading performance across memory benchmarks. SimpleMemVLA leads all four memory benchmarks with 94.0% on RMBench (+11.0 points), 88.3% on RoboMME (+43.7), 74.0% on MIKASA-Robo (+29.6) and 63.6% task success on RoboMemArena (+17.4), relative to the strongest prior VLA baseline in each suite (Figure 3; Appendix A, Tables 2–5).

Gains concentrate on tasks with greater memory demands. On RMBench, SimpleMemVLA is the only method in Table 2 whose average rises from the single-memory to the multi-memory setting, from 91.6% to 97.0%. On RoboMemArena, its largest gains over the strongest overall baseline, FrameSamp+Modul, occur in Occlusion and Counting (+25.2 and +40.0 points in task success), while it does not lead on Transferring (Table 5). On MIKASA-Robo, the margin over the best prior VLA on each RememberColor task widens from 12 to 28 to 39 points as the number of candidates increases from 3 to 5 to 9 (Table 4). Together, these patterns suggest that the advantage is

Table 1: **Real-world autonomous manipulation.** Each initial configuration is evaluated over ten trials. Demonstrations denote the real-robot data used for fine-tuning.

Task	Demonstrations	Configurations	Successes / Trials	SR (%)
Cover Blocks	180	6	35/60	58.3
Put Back Block	308	4	28/40	70.0

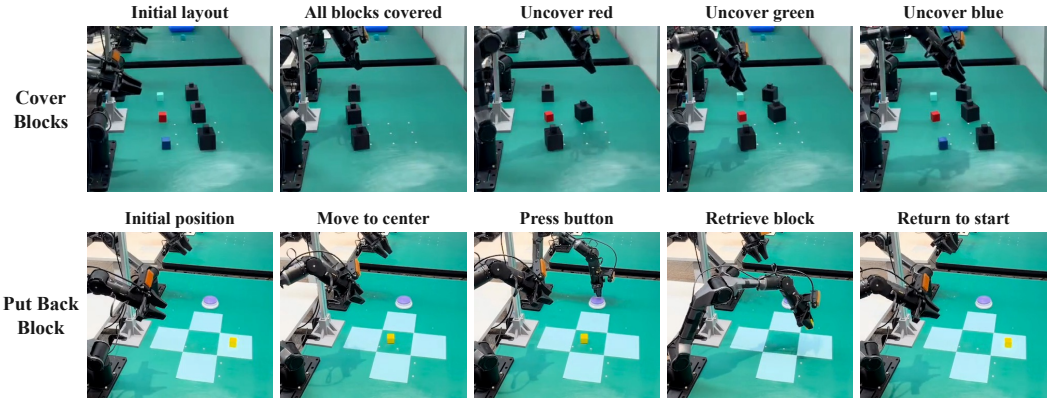


Figure 4: **Real-world autonomous rollouts.** Top: Cover Blocks. Bottom: Put Back Block. Each row shows selected frames from one successful rollout in chronological order.

tied to using historical evidence rather than a uniform improvement in low-level control; Section 4.4 tests this attribution under a matched backbone and training setup.

Preserving general-purpose manipulation performance. SimpleMemVLA achieves 97.5% on LIBERO, matching RIPT-VLA, and the same model transfers zero-shot to LIBERO-Plus at 78.4%, exceeding MemoryVLA++ by 5.3 points (Tables 6 and 7 in Appendix A). These results show that the memory gains coexist with competitive standard manipulation and strong robustness under the evaluated perturbations.

4.3 Real-World Evaluation

On a physical dual-arm robot, we evaluate real-world versions of the RMBench tasks Cover Blocks and Put Back Block, which require recalling color-to-position bindings under occlusion and an object's initial location, respectively. We fine-tune on 180 and 308 real-robot demonstrations, respectively. At deployment, history grows with the episode until a 60 s cap, then retains the most recent 60 s, sampled at 2 fps (at most 120 frames). Exact streaming inference prefills shared history during action execution, keeping decision latency manageable for closed-loop control (Section 3.4).

With ten autonomous trials per initial configuration, SimpleMemVLA succeeds in 58.3% (35/60) across six Cover Blocks layouts and 70.0% (28/40) across four Put Back Block positions (Table 1). Figure 4 shows representative autonomous rollouts. Appendix C gives per-configuration results and implementation details. Qualitative observations indicate that failures mainly arise from low-level execution errors such as unsuccessful grasps. Even after a failed grasp, the Cover Blocks policy can still identify the covers hiding the red, green and blue blocks. These results show that native video context remains usable as memory under real-world perception and within a real-time control loop.

4.4 Controlled Comparison of Memory Interfaces

We compare native video context with three memory interfaces on a shared stack, fixing the training data, backbone, sub-task supervision, action head and optimizer. Across all sixteen RoboMME tasks ($n = 50$ episodes per task), success is 88.3% for native context, versus 31.5% for retrieval, 22.6% for token compression and 20.6% for recurrent state (Figure 3b; Table 3 in Appendix A).

Retrieval uses eight uniformly sampled frames that preserve visual content but omit temporal order and timestamps, yielding 42.0% on Permanence but only 14–20% on the two swap tasks and

10.0% on PatternLock, where temporal relationships matter. Compressing the full history into 64 tokens nearly matches native context on SwingXtimes (92% versus 96%) but reaches only 13.5% on Permanence and 12.5% on Reference, suggesting better retention of aggregate counts than specific past percepts. The recurrent variant instead carries history forward through repeated updates to a fixed 16-token state, without direct access to past observations. Counting is its strongest category (31.0%), followed by Permanence (24.0%), Reference (17.5%) and Imitation (10.0%), all below native context. These results are consistent with a cost of write-time commitment in the tested interfaces: restricting history before the current decision can remove needed evidence. Even a symbolic pipeline given ground-truth perception, GroundSG, reaches only 84.1%, below the 88.3% of native context, suggesting that what limits these mechanisms is write-time commitment rather than the accuracy of what they commit. Task-level results appear in Appendix D (Figure 7; Table 10), with implementation details in Appendix B.

4.5 History Interventions

The benchmark results show strong memory-task performance. We next test whether the deployed policy actually uses its visual history as memory. At each target decision, we hold the model, instruction, current observation, robot state and deployment path fixed, edit only the historical video, and regenerate the output (Figure 11 in Appendix F.1).

Removing task-relevant evidence. Masking the segment showing where RGB blocks were covered in RMBench cover-blocks (Figure 11 rows 2 and 4) or the historical color cue in MIKASA-Robo RememberColor causes selection of the wrong cover or color, respectively. Masking task-irrelevant history changes neither output, showing sensitivity to evidence rather than masking itself. On RoboMME PickXtimes, masking one completed pick-and-place event in an otherwise valid rollout lowers the inferred count by one (e.g., "the fourth time" becomes "the third time") and induces an additional pick. Masking a matched segment without such an event leaves the count unchanged, indicating memory of the exact event count rather than mere occurrence (Figure 11 rows 8 and 9).

Replacing historical evidence. Replacements splice a visual segment from one rollout into another's remaining history. On RMBench, a replacement covering event showing the RGB blocks under different covers redirects the policy's choices accordingly. A replacement preserving the original cover assignments leaves its choices unchanged (Figure 11 row 3). On MIKASA-Robo, replacing a magenta cue with red redirects grasping to the red block (Figure 11 row 6). On RoboMME PatternLock, replacing the current demonstration video with another causes the policy to follow the replacement demonstration. Another demonstration with the same move leaves behavior unchanged (Figure 11 row 11). Neither edit type appears during training. The frozen policy chooses the cover or colored block indicated by the replacement memory, follows the substituted demonstration, and counts one fewer repetition when a completed event is masked. It makes these changes without intervention-specific supervision or parameter updates, demonstrating an emergent form of visual in-context learning. Together, the masking and replacement experiments show that native context serves both as memory of past events and as an inference-time visual interface through which new evidence can re-specify behavior.

4.6 Streaming Inference with Prefix Prefill

The preceding results establish both the effectiveness of SimpleMemVLA and its use of visual history. The remaining practical question is the cost of processing that history. With a 60 s context window, full recomputation processes approximately 5.6k tokens at every decision, compared with approximately 0.5k tokens for single-frame input. The streaming implementation of Section 3.4 uses a shared-prefix cache to reduce decision latency while producing the same output (Figure 2b). On one H100, for the representative minute-scale configuration evaluated in Figure 5a, streaming reduces decision latency from 1.02 s to 0.68 s, close to the approximately 0.65 s latency of the same model with single-frame input.

In our scaling experiment, an artificially constructed input corresponding to 45 min of history contains approximately 245k tokens, near the backbone's 262k-token context limit. At this input length, full-recomputation latency reaches 32.1 s per decision, whereas the streamed decision path requires only 1.18 s (Figure 5b). The key–value cache size depends on the retained context length and remains bounded under the standard 60 s history cap. Appendix G presents a sliding-window-attention

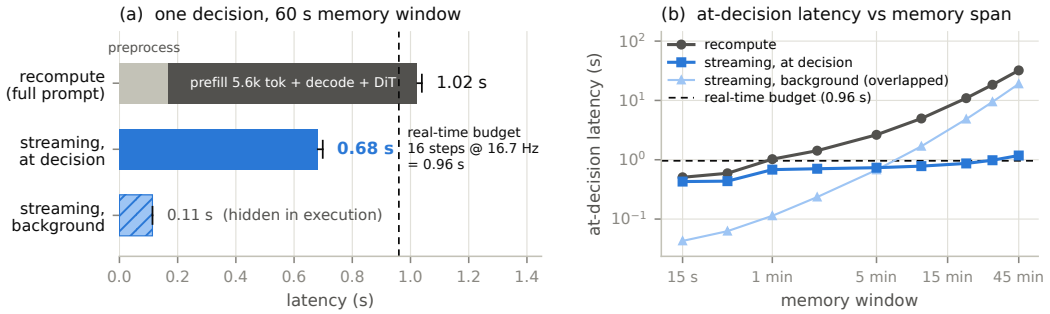


Figure 5: **Streaming inference reduces decision latency.** Measured on one H100 (bf16, batch size 1). *(a)* With 60 s histories, streaming reduces decision latency from 1.02 s to 0.68 s, close to the same-model single-frame baseline of approximately 0.65 s. *(b)* On artificially constructed 45 min inputs (245k tokens), decision-path latency falls from 32.1 s to 1.18 s.

(SWA) variant that uses the same bounded attention window during training and inference. At deployment, it appends one new temporal patch and evicts one expired cache block at every decision, keeping cache state independent of the total episode length (Figure 13).

4.7 Ablations: Dissecting the Memory Pathway

Evidence must remain in the window. We probe complementary requirements on RMBench: cover-blocks recalls a single past event, while press-button counts repeated events throughout the trajectory (Figure 12 in Appendix F.2). A 30 s window retains the covering event from approximately 25 s earlier. A 15 s window excludes it, causing failure. Shorter windows progressively remove completed presses and reduce press-button success. Both tasks fail with only the current frame (Figure 12a). **Order and timestamps provide complementary temporal information.** Shuffling frame order causes both tasks to fail. Zeroing plaintext timestamps leaves cover-blocks unaffected but substantially degrades press-button, indicating that explicit timing supports counting (Figure 12b). **Contextual hidden states carry the primary history-dependent control signal.** Replacing token embeddings with another target's largely preserves behavior. Replacing hidden states redirects it toward that target, and zeroing them causes failure. Contextual hidden states therefore carry the primary history-dependent control signal, while token embeddings support stable execution (Figure 12d). **The representation must be refreshed for each decision.** Deleting the target word makes the visible sub-task identical across decisions, yet the policy preserves the required cover sequence and press count. Reusing hidden states from even one decision earlier sharply reduces success, whereas equally stale position IDs leave it unchanged (Figure 12c). The backbone forms this representation from retained history after the current decision context becomes available, selecting evidence at read time rather than choosing what to retain at write time. Overall, the ablations show that native-context memory needs retained temporal evidence and refreshed contextual hidden states, not a separate memory module.

5 Conclusion

We have shown that a pretrained VLM's native video context can serve as the memory of a VLA without a dedicated memory module. Because the history is kept intact until each decision, the policy avoids write-time commitment, and this alone yields state-of-the-art results on four memory benchmarks, a wide margin over retrieval, compression and recurrent-state mechanisms under the same backbone and training setup, and autonomous history-dependent manipulation on a physical dual-arm robot. Prefilling the shared history during action execution keeps decision latency close to that of a single-frame VLA. Although our findings are limited to the evaluated backbones, benchmarks and memory interfaces, we argue that native context should serve as a matched baseline for future VLA memory mechanisms. Before adding dedicated memory machinery, the first comparison should be the same policy given access to its own visual history.

AI USE STATEMENT

In this work, we used generative AI tools for writing and editing the manuscript, and for offline sub-task annotation of VLA training data. All AI-generated sub-task texts were reviewed and approved by human annotators before being used. We have reviewed all AI-assisted writing. We take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies vision-language-action policies for robot manipulation in public simulation benchmarks and on a physical dual-arm robot in a controlled laboratory setting. It involves no human subjects, no personally identifiable data, and no release of sensitive datasets. Real-robot experiments were conducted under standard laboratory safety practices with supervised operation. We identify no specific ethical or dual-use concerns beyond those generally associated with robot learning research, and we adhere to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

We provide anonymous source code, training configurations, and evaluation protocols at `https://anonymous.4open.science/r/SimpleMemVLA`. Section 3 specifies input construction (sampling rates, timestamps, and modality rules) and the shared training/inference implementation, including streaming prefill. Section 4 and the appendix detail benchmarks, protocols, controlled variants, history interventions, and ablations. All reported evaluations use held-out seeds.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *Neural Information Processing Systems*, 2022. doi: 10.52202/068431-1723.

Shuai Bai, Yuxuan Cai, Rui-Zhe Chen, Ke-qin Chen, Xiong-Hui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Rongyao Fang, Chang Gao, et al. Qwen3-VL technical report. *arXiv.org*, 2025a. doi: 10.48550/arXiv.2511.21631.

Shuai Bai, Ke-qin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, K. Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv.org*, 2025b. doi: 10.48550/arXiv.2502.13923.

Kevin Black, Noah Brown, Danny Driess, A. 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. *Robotics: Science and Systems XXI*, 2025. doi: 10.15607/rss.2025.xxi.010.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, K. Choromanski, Tianli Ding, Danny Driess, Kumar Avinava Dubey, Chelsea Finn, Peter R. Florence, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, 2023. doi: 10.48550/arXiv.2307.15818.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, J. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, G. Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Neural Information Processing Systems*, 2020.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. UniVLA: Learning to act anywhere with task-centric latent actions. In *Robotics: Science and Systems XXI*, 2025. doi: 10.15607/rss.2025.xxi.014.

Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. Recurrent memory transformer. In *Advances in Neural Information Processing Systems 35*, 2022. doi: 10.52202/068431-0805.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. WorldVLA: Towards autoregressive action world model. *arXiv.org*, 2025. doi: 10.48550/arXiv.2506.21539.

Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, et al. GR-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv.org*, 2024. doi: 10.48550/arXiv.2410.06158.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Qiwei Liang, Zixuan Li, Xianlian Lin, Yiheng Ge, Zhenyu Gu, et al. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv.org*, 2025. doi: 10.48550/arXiv.2506.18088.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yanjiao Qin, Haochen Shi, Zixuan Li, Yifan Hu, Yingsheng J Zhang, Kaixuan Wang, Yue Chen, et al. Rmbench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv.org*, 2026. doi: 10.48550/arxiv.2603.01229.

E. Cherepanov, Nikita Kachaev, A. Kovalev, and Aleksandr I. Panov. Memory, benchmark & robots: A benchmark for solving complex tasks with reinforcement learning. *arXiv.org*, 2025. doi: 10.48550/arXiv.2502.10550.

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, 2026.

Cheng Chi, S. Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, B. Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems Conference*, 2023. doi: 10.1177/02783649241273668.

Yinpei Dai, H. 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 *arXiv.org*, 2026. doi: 10.48550/arXiv.2603.04639.

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 *International Conference on Machine Learning*, 2025. doi: 10.48550/arXiv.2501.18564.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Yihuai Gao, Jeff Jinyun Liu, Shuang Li, and Shuran Song. Gated memory policy: In-context memorization and adaptation. *arXiv*, 2026. doi: 10.48550/arxiv.2604.18933.

Yanjiang Guo, L. Shi, Jianyu Chen, and Chelsea Finn. Ctrl-World: A controllable generative world model for robot manipulation. *arXiv.org*, 2025. doi: 10.48550/arXiv.2510.10125.

Yanbin Hu, Jin Cui, Jiayi Lu, Ruixuan Yang, Junjie Ye, Bo Zhao, Xingyu Chen, Xuguang Lan, and Pengju Ren. ECHO: Continuous hierarchical memory for vision-language-action models. *arXiv.org*, 2026. doi: 10.48550/arXiv.2605.10993.

Yuzhi Huang, Weijue Bu, Ziyi Xiong, Jie Wu, Fanding Huang, Jingyan Jiang, and Zhi Wang. Chain-vla: Chaining vision-language-action queries through a unified execution state for long-horizon manipulation. *arXiv preprint arXiv:2608.02326*, 2026.

Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U-Xuan Tan, Navonil Majumder, and Soujanya Poria. NORA: A small open-sourced generalist vision language action model for embodied tasks. *arXiv.org*, 2025. doi: 10.48550/arXiv.2504.19854.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, A. Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv.org*, 2025. doi: 10.48550/arXiv.2504.16054.

Huiwon Jang, Sihyun Yu, Heeseung Kwon, Hojin Jeon, Younggyo Seo, and Jinwoo Shin. ContextVLA: Vision-language-action model with amortized multi-frame context. *arXiv.org*, 2025. doi: 10.48550/arXiv.2510.04246.

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

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, A. Balakrishna, Suraj Nair, Rafael Rafailov, E. Foster, Grace Lam, Pannag R. Sanketi, et al. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning*, 2024. doi: 10.48550/arXiv.2406.09246.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Robotics: Science and Systems XXI*, 2025. doi: 10.15607/rss.2025.xxi.017.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Y. Seo, and Jinwoo Shin. HAMLET: Switch your vision-language-action model into a history-aware policy. *arXiv.org*, 2025. doi: 10.48550/arXiv.2510.00695.

Huashuo Lei, Wenxuan Song, Huarui Zhang, Jieyuan Pei, Jiayi Chen, Haodong Yan, Han Zhao, Pengxiang Ding, Zhipeng Zhang, Lida Huang, et al. RoboMemArena: A comprehensive and challenging robotic memory benchmark. *arXiv.org*, 2026. doi: 10.48550/arXiv.2605.10921.

Mingcong Lei, Honghao Cai, Binbin Que, Zezhou Cui, Liangchen Tan, Junkun Hong, Gehan Hu, Shuangyu Zhu, Yimou Wu, Shaohan Jiang, et al. RoboMemory: A brain-inspired multi-memory agentic framework for lifelong learning in physical embodied systems. *arXiv.org*, 2025. doi: 10.48550/arXiv.2508.01415.

Hao Li, Shuai Yang, Yilun Chen, Xinyi Chen, Xiaoda Yang, Yang Tian, Hanqing Wang, Tai Wang, Dahua Lin, Feng Zhao, et al. CronusVLA: Efficient and robust manipulation via multi-frame vision-language-action modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026. doi: 10.1609/aaai.v40i22.38903.

Runhao Li, Wenkai Guo, Zhenyu Wu, Changyuan Wang, Haoyuan Deng, Zhenyu Weng, Yap-Peng Tan, and Ziwei Wang. MAP-VLA: Memory-augmented prompting for vision-language-action models in robotic manipulation. *arXiv.org*, 2025. doi: 10.48550/arXiv.2511.09516.

Li Lin, Wujun Xu, Wei Meng, Kaiwen Xia, K. Cheong, and Shuai Wang. FibVLA: An efficient temporal vision-language-action model with fibonacci sampling. *arXiv*, 2026.

Min Lin, Xiwen Liang, Bingqian Lin, Jingzhi Liu, Zijian Jiao, Kehan Li, Yuhan Ma, Yuecheng Liu, Shen Zhao, Yuzheng Zhuang, et al. Echovla: Robotic vision-language-action model with synergistic declarative memory for mobile manipulation. *arXiv.org*, 2025a. doi: 10.48550/arXiv.2511.18112.

Minghui Lin, Pengxiang Ding, Shu Wang, Zifeng Zhuang, Yang Liu, Xinyang Tong, Wenxuan Song, Shangke Lyu, Siteng Huang, and Donglin Wang. HiF-VLA: Hindsight, insight and foresight through motion representation for vision-language-action models. *arXiv.org*, 2025b. doi: 10.48550/arXiv.2512.09928.

Y. Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2022.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qian Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems 36*, 2023. doi: 10.52202/075280-1939.

Jiaming Liu, Mengzhen Liu, Zhenyu Wang, Lily Lee, Kaichen Zhou, Pengju An, Senqiao Yang, Renrui Zhang, Yandong Guo, and Shanghang Zhang. RoboMamba: Multimodal state space model for efficient robot reasoning and manipulation. *arXiv.org*, 2024a. doi: 10.48550/arXiv.2406.04339.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: A diffusion foundation model for bimanual manipulation. *International Conference on Learning Representations*, 2024b. doi: 10.48550/arXiv.2410.07864.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied AI. *IEEE Transactions on Neural Networks and Learning Systems*, 2024. doi: 10.1109/TNNLS.2025.3650584.

Nvidia, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi-JimFan, Yu Fang, Dieter Fox, Fengyuan Hu, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv.org*, 2025. doi: 10.48550/arXiv.2503.14734.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. In *Robotics: Science and Systems XXI*, 2025. doi: 10.15607/rss.2025.xxi.012.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yani Ding, Zhigang Wang, Jiayuan Gu, Bin Zhao, Dong Wang, et al. SpatialVLA: Exploring spatial representations for visual-language-action model. In *Robotics: Science and Systems XXI*, 2025. doi: 10.15607/rss.2025.xxi.011.

Hongyu Qu, Jianzhe Gao, Xiaobin Hu, Shaohuan Yang, Xinlei Yu, Rui Yan, Wenguan Wang, Xiangbo Shu, and Shuicheng Yan. Dual latent memory in vision-language-action models for robotic manipulation. *arXiv*, 2026.

Ranjan Sapkota, Yang Cao, Konstantinos I. Roumeliotis, and Manoj Karkee. Vision-language-action models: Concepts, progress, applications and challenges. *arXiv.org*, 2025. doi: 10.48550/arXiv.2505.04769.

Yujiao Shen, Shulin Tian, Jingkang Yang, and Ziwei Liu. A simple baseline for streaming video understanding. *arXiv preprint arXiv:2604.02317*, 2026.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Feng 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 *arXiv.org*, 2026a. doi: 10.48550/arXiv.2508.19236. arXiv:2508.19236.

Haochen 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.org*, 2026b. doi: 10.48550/arXiv.2606.09827.

A. Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. MemER: Scaling up memory for robot control via experience retrieval. In *arXiv.org*, 2026. doi: 10.48550/arXiv.2510.20328.

Mingyang Sun, Xiujian Liang, Jiude Wei, Qichen He, Donglin Wang, Cewu Lu, and Jianhua Sun. Analytic concept-centric memory for agentic embodied manipulation. *arXiv.org*, 2026a. doi: 10.48550/arXiv.2606.29774.

Xiaoquan Sun, Ruijian Zhang, Chen Cao, Yihan Sun, Jiahui Chen, Zetian Xu, Bo Chen, Haijier Chen, Zhen Yang, Jiarun Zhu, et al. Himem-wam: Hierarchical memory-gated world action models for robotic manipulation. *arXiv preprint arXiv:2606.10363*, 2026b.

Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models. *arXiv.org*, 2025. doi: 10.48550/arXiv.2505.17016.

O. Team, Dibya Ghosh, H. Walke, Karl Pertsch, Kevin Black, Oier Mees, S. Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems XX*, 2024. doi: 10.15607/rss.2024.xx.090.

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

Kai Wang, Zhaopeng Gu, Yixiang Chen, Yuan Xu, Qisen Ma, Peng Su, Zhaowen Li, Yan Huang, and Liang Wang. DIM-WAM: World-action modeling with diverse historical event memory. *arXiv.org*, 2026a. doi: 10.48550/arXiv.2606.27677.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Ke-Yang Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv.org*, 2024. doi: 10.48550/arXiv.2409.12191.

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*, 2026b.

Ronald J. Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 1990. doi: 10.1162/neco.1990.2.4.490.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv*, 2025. doi: 10.48550/arXiv.2505.09388.

Ganlin Yang, Zhangzheng Tu, Yuqiang Yang, Sitong Mao, Junyi Dong, Tianxing Chen, Jiaqi Peng, Jing Xiong, Jiafei Cao, Jifeng Dai, et al. Eventvla: Event-driven visual evidence memory for long-horizon vision-language-action policies. *arXiv.org*, 2026a. doi: 10.48550/arXiv.2606.20092.

Sizhe Yang, Juncheng Mu, Tianming Wei, Chenhao Lu, Xiaofan Li, Linning Xu, Zhengrong Xue, Zhecheng Yuan, Dahua Lin, Jiangmiao Pang, et al. MemoryWAM: Efficient world action modeling with persistent memory. *arXiv.org*, 2026b. doi: 10.48550/arXiv.2606.20562.

Cheng Yin, Yankai Lin, Wang Xu, Sikyuen Tam, Xiangrui Zeng, Zhiyuan Liu, and Zhouping Yin. DeepthinkVLA: Enhancing reasoning capability of vision-language-action models. In *Third Conference on Language Modeling*, 2026.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv.org*, 2026. doi: 10.48550/arXiv.2603.16666.

Tony Zhao, Vikash Kumar, S. Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems XIX*, 2023. doi: 10.15607/rss.2023.xix.016.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayi Zou, Yilun Chen, Jia Zeng, et al. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *arXiv.org*, 2026. doi: 10.48550/arXiv.2510.10274.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daum'e, A. Kolobov, Furong Huang, and Jianwei Yang. TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2412.10345.

Shoujing Zhu, Zhenyang Liu, F. Wang, Jiafeng Wang, Bo Yue, Guiliang Liu, Si Wu, X. Xue, and Taiping Zeng. WeaveLA: Event driven cross-subtask latent memory weaving for repetitive robot manipulation. *arXiv.org*, 2026. doi: 10.48550/arXiv.2606.17463.

A Detailed Benchmark Results

Tables 2–7 provide the complete per-suite comparisons summarized in Figure 3. Task-level RoboMME results and additional LIBERO-Plus protocol details and breakdowns appear in Appendices D and E.

Table 2: **RMBench** per-task success rates (%), averaged over the nine tasks with published baselines. Most baselines train one specialist model per task while SimpleMemVLA is a single multi-task model evaluated at $n = 100$ seeds per task in the streaming deployment of Section 4.6. †No prior VLA reports Place-Mat so it is excluded from all averages. Best per column in **bold**.

	Single-memory $M(1)$						Multi-memory $M(n)$						
Method	Obs&PU	Rearr.	PutBack	SwapB	SwapT	*Avg*	Battery	Rank.	Cover	Press	Place-Mat†	*Avg*	Overall
One specialist model per task													
DP (Chi et al., 2023)	1	0	0	11	20	6.4	10	10	0	0	–	5.0	5.8
ACT (Zhao et al., 2023)	1	29	0	2	2	6.8	19	0	0	0	–	4.8	5.9
X-VLA (Zheng et al., 2026)	9	13	18	16	3	11.8	26	1	2	0	–	7.3	9.8
Mem-0 (Chen et al., 2026)	4	89	90	67	14	52.8	28	18	68	0	–	28.5	42.0
DIM-WAM (Wang et al., 2026a)	13	99	98	96	**97**	80.6	48	87	56	34	–	56.3	69.8
MemoryWAM (Yang et al., 2026b)	27	**100**	**100**	**100**	94	84.2	41	**100**	**98**	87	–	81.5	83.0
Other published baselines													
$\pi_{0.5}$ (Intelligence et al., 2025)	9	13	11	24	15	14.4	16	6	0	0	–	5.5	10.4
HiMem-WAM (Sun et al., 2026b)	28	33	32	38	27	31.6	28	24	19	8	–	19.8	26.3
Single multi-task checkpoint													
EventVLA (Yang et al., 2026a)	21	96	95	96	87	79.0	35	81	97	3	–	54.0	67.8
SimpleMemVLA (Ours)	**65**	**100**	**100**	**100**	93	**91.6**	**90**	**100**	**98**	**100**	**100**	**97.0**	**94.0**

Table 3: **RoboMME** category-level success rates (%). AVG is over all sixteen tasks. Gray rows are reference-only and excluded from ranking. **Bold** and underline mark ranks 1 and 2. The SimpleMemVLA variant rows re-create one mechanism family each on the otherwise unchanged SimpleMemVLA stack. Per-task results are in Table 10.

Memory	Method	Integr. / VLM	Counting	Permanence	Reference	Imitation	AVG
Human	Human Performance	-	88.5	91	93	89.5	90.5
Symbolic (Oracle)	SimpleSG (Dai et al., 2026)	GT VLM	82.6	21.6	32.3	61.9	49.6
Symbolic (Oracle)	GroundSG (Dai et al., 2026)	GT VLM	83.9	93.3	95.2	64.0	84.1
Symbolic	SimpleSG	Gemini	39.0	13.5	22.5	18.0	23.3
Symbolic	SimpleSG	QwenVL	44.6	19.6	25.2	26.6	29.0
Symbolic	GroundSG	Gemini	12.8	15.8	11.3	6.5	11.6
Symbolic	GroundSG	QwenVL	38.0	39.3	31.6	21.9	32.7
Perceptual	TokenDrop (Dai et al., 2026)	Context	57.9	26.9	23.7	29.5	34.5
Perceptual	TokenDrop	Modul	52.3	26.8	34.7	38.3	38.0
Perceptual	TokenDrop	Expert	59.4	23.6	25.3	31.1	34.9
Perceptual	FrameSamp (Dai et al., 2026)	Context	50.1	23.3	20.9	28.3	30.7
Perceptual	FrameSamp	Modul	65.2	25.1	36.3	<u>51.4</u>	<u>44.5</u>
Perceptual	FrameSamp	Expert	<u>66.8</u>	25.2	24.1	28.9	36.3
Recurrent	TTT (Dai et al., 2026)	Context	36.1	21.4	22.0	9.7	22.3
Recurrent	TTT	Modul	34.5	22.2	20.4	10.7	22.0
Recurrent	TTT	Expert	36.0	22.9	19.7	10.8	22.3
Recurrent	RMT (Dai et al., 2026)	Context	32.2	15.4	19.7	10.6	19.5
Recurrent	RMT	Modul	34.1	15.8	21.1	9.7	20.2
Recurrent	RMT	Expert	34.4	15.7	14.7	7.8	18.1
Other	$\pi_{0.5}$ (Intelligence et al., 2025)	-	28.8	17.0	17.2	8.8	17.9
Other	WeaveLA (Zhu et al., 2026)	-	31.0	19.5	12.5	9.5	18.1
Other	$\pi_{0.5}$ w/ past actions (Dai et al., 2026)	-	29.1	22.8	15.9	11.2	19.7
Other	SAM2Act+ (Fang et al., 2025)	-	35.3	26.0	16.8	7.3	21.4
Other	MemER (Sridhar et al., 2026)	-	48.8	<u>53.2</u>	<u>38.0</u>	29.5	42.4
Ours variant	SimpleMemVLA (Retrieval)	-	46.5	42.0	13.5	24.0	31.5
Ours variant	SimpleMemVLA (Token Compression)	-	46.0	13.5	12.5	18.5	22.6
Ours variant	SimpleMemVLA (Recurrent State)	-	31.0	24.0	17.5	10.0	20.6
Ours	SimpleMemVLA	-	**91.5**	**96.0**	**82.5**	**83.0**	**88.3**

Table 4: **MIKASA-Robo** five-task success rates (%). GMP is a per-task non-VLA reference. Best VLA per column in **bold**.

Method	ShellGame	Intercept	RC-3	RC-5	RC-9	Avg
Vision-language-action models						
CronusVLA (Li et al., 2026)	32	5	31	13	9	18.0
SpatialVLA (Qu et al., 2025)	23	27	27	17	11	21.0
OpenVLA-OFT (Kim et al., 2025)	47	14	59	16	6	28.4
π_0 (Black et al., 2025)	33	42	35	22	15	29.4
Octo (Team et al., 2024)	46	39	45	17	11	31.6
MemoryVLA (Shi et al., 2026a)	88	24	44	30	20	41.2
MemoryVLA++ (Shi et al., 2026b)	97	40	50	19	16	44.4
Compact non-VLA memory policy (reference)						
GMP (Gao et al., 2026)	98	83	**80**	**61**	17	67.8
SimpleMemVLA (Ours)	**99**	**83**	71	58	**59**	**74.0**

Table 5: **RoboMemArena** category-level TSR and CSR (%) over all 26 tasks under the official protocol. MemER is the benchmark authors' reimplementation, FrameSamp+Modul the only external leaderboard entry, the gray oracle row excluded from ranking. Best per column in **bold**.

	Transferring		Occlusion		Counting		Sequence		Average	
Method	TSR	CSR	TSR	CSR	TSR	CSR	TSR	CSR	TSR	CSR
$\pi_{0.5}$ (Intelligence et al., 2025)	20.0	42.8	12.7	17.2	14.3	50.9	60.0	71.6	21.5	38.7
HiF-VLA (Lin et al., 2025b)	17.5	38.9	12.7	27.1	8.6	45.9	42.5	70.2	16.9	39.8
MemoryVLA (Shi et al., 2026a)	15.0	37.2	7.3	13.1	14.3	55.1	37.5	65.2	15.0	35.3
MemER (Sridhar et al., 2026)	20.0	36.1	16.4	33.2	27.1	65.1	65.0	79.1	27.3	49.1
FrameSamp+Modul (Dai et al., 2026)	**63.8**	**72.1**	39.1	56.5	31.4	57.8	73.8	86.9	46.2	63.9
PrediMem (Lei et al., 2026)	22.5	45.2	27.3	38.4	45.7	69.3	72.5	**89.5**	38.5	55.2
Ground truth (oracle) (Lei et al., 2026)	32.5	54.8	33.6	49.8	51.4	75.6	85.0	92.3	46.1	64.8
SimpleMemVLA (Ours)	35.8	37.4	**64.3**	**76.8**	**71.4**	**79.3**	**75.5**	81.1	**63.6**	**72.1**

Table 6: **LIBERO** success rates (%) on the four standard suites, 500 trials per suite. Best per column in **bold**.

Method	Spatial	Object	Goal	Long	Avg
General-purpose policies and VLAs					
Diffusion Policy (Chi et al., 2023)	78.3	92.5	68.3	50.5	72.4
TraceVLA (Zheng et al., 2025)	84.6	85.2	75.1	54.1	74.8
Octo (Team et al., 2024)	78.9	85.7	84.6	51.1	75.1
OpenVLA (Kim et al., 2024)	84.7	88.4	79.2	53.7	76.5
SpatialVLA (Qu et al., 2025)	88.2	89.9	78.6	55.5	78.1
π_0-FAST (Pertsch et al., 2025)	96.4	96.8	88.6	60.2	85.5
π_0 (Black et al., 2025)	96.8	98.8	95.8	85.2	94.2
DeepThinkVLA (Yin et al., 2026)	96.6	99.0	96.4	**96.2**	97.0
OpenVLA-OFT (Kim et al., 2025)	97.6	98.4	97.9	94.5	97.1
RIPT-VLA (Tan et al., 2025)	–	–	–	–	**97.5**
Memory-augmented VLAs					
MemoryVLA (Shi et al., 2026a)	**98.4**	98.4	96.4	93.4	96.7
CronusVLA (Li et al., 2026)	97.3	**99.6**	96.9	94.0	97.0
ContextVLA (Jang et al., 2025)	**98.4**	99.0	97.2	93.4	97.0
SimpleMemVLA (Ours)	98.2	98.8	**98.0**	95.0	**97.5**

Table 7: **LIBERO-Plus** zero-shot robustness transfer to the 10,030 perturbed tasks, all policies trained on standard LIBERO only. Best per column in **bold**.

Method	Camera	Robot	Lang.	Light	Backg.	Noise	Layout	Total
General-purpose policies and VLAs								
OpenVLA (Kim et al., 2024)	0.8	3.5	23.0	8.1	34.8	15.2	28.5	15.6
WorldVLA (Cen et al., 2025)	0.1	27.9	41.6	43.7	17.1	10.9	38.0	25.0
Diffusion Policy (Chi et al., 2023)	1.6	32.3	77.0	25.0	19.8	20.3	42.2	31.7
NORA (Hung et al., 2025)	2.2	37.0	65.1	45.7	58.6	12.8	62.1	39.0
UniVLA (Bu et al., 2025)	1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9
Fast-WAM (Yuan et al., 2026)	16.4	44.5	68.9	78.2	53.7	37.7	60.7	51.5
π_0 (Black et al., 2025)	13.8	6.0	58.8	85.0	81.4	**79.0**	68.9	53.6
π_0-FAST (Pertsch et al., 2025)	65.1	21.6	61.0	73.2	73.2	74.4	68.8	61.6
OpenVLA-OFT$_m$ (Kim et al., 2025)	55.6	21.7	81.0	92.7	91.0	78.6	68.7	67.9
RIPT-VLA (Tan et al., 2025)	55.2	31.2	77.6	88.4	91.6	73.5	74.2	68.4
OpenVLA-OFT (Kim et al., 2025)	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6
Memory-augmented VLAs								
MemoryVLA (Shi et al., 2026a)	42.7	44.9	84.4	92.8	95.0	62.1	**84.7**	70.2
MemoryVLA++ (Shi et al., 2026b)	36.4	**68.9**	**88.7**	93.8	90.6	63.5	83.8	73.1
SimpleMemVLA (Ours)	**76.9**	68.4	68.6	**96.4**	**96.8**	74.2	78.0	**78.4**

Table 8: Per-benchmark instantiation of the SimpleMemVLA configuration tuple and the resulting training cost. Everything else about the method is identical across suites. Training time is wall-clock hours on 128 H100 GPUs.

	RMBench	RoboMME	MIKASA-Robo	RoboMemArena	LIBERO
Embodiment	Aloha-AgileX, bimanual	Panda, single	Panda, single	Franka, single	Franka, single
Action dimension d_a	14	8	8	7	7
History camera $\mathcal{C}_{\text{hist}}$	head	front	overhead	front	agentview
Current cameras $\mathcal{C}_{\text{cur}}$	both wrists	wrist	wrist	wrist	wrist
Window T_w (s)	60	60	3	126	30
Video rate f_v (fps)	2	2	20	1	2
Frames K	120	120	60	126	60
Stride s	8	10	1	20	10
Action horizon H	30	30	16	16	16
Training time (h)	23	20	16	28	14

B Implementation Details

Table 8 lists the per-benchmark instantiation of the configuration tuple from Section 3.4. All suites share the same backbone (Qwen3.5-4B, native video interface with native plaintext timestamps), the same conditioning rule and the same losses. Optimization uses AdamW with separate learning rates for the backbone and the action head (10^{-5} and 5×10^{-5}), cosine decay and gradient clipping. The stride s is the integer rounding of f_c/f_v.

Recurrent-state training. The recurrent variant maintains a 16-token state, with gradients propagated through unrolls of four consecutive decisions. Finite-horizon backpropagation limits training cost (Williams & Peng, 1990); recurrent memory transformers likewise treat the unroll length as a training hyperparameter (Bulatov et al., 2022). Our results characterize the evaluated implementation under this training protocol.

Sub-task labels. The sub-task label g_t^* of Section 3.3 is written offline by a cloud VLM: shown each demonstration, it states a one-sentence sub-task for every supervision anchor and the dataset stores the result per frame. The three mechanism variants of Section 4.4 are trained on these identical labels and the deployed policy never queries the annotator, so annotation is a training-data cost shared by every within-stack comparison and absent at inference.

Benchmarks and metrics. **RMBench** (Chen et al., 2026) comprises ten memory-dependent bimanual tasks built on RoboTwin 2.0 (Chen et al., 2025), grouped into single-memory $M(1)$ and multi-memory $M(n)$ settings. Cross-method averages exclude Place-Mat, for which no prior baseline is reported, and cover the remaining nine tasks. Most published RMBench baselines train a separate specialist for each task. **RoboMME** (Dai et al., 2026) contains sixteen single-arm tasks spanning Counting, Permanence, Reference and Imitation, and reports fourteen memory variants built on a shared $\pi_{0.5}$ backbone. **MIKASA-Robo** (Cherepanov et al., 2025) comprises five single-arm tasks testing memory under occlusion and partial observability. **RoboMemArena** (Lei et al., 2026) contains twenty-six single-arm tasks across Transferring, Occlusion, Counting and Sequence, with trajectories averaging more than one thousand control steps. It reports all-stage task success rate (TSR) and stage completion rate (CSR).

LIBERO (Liu et al., 2023) evaluates general-purpose manipulation across the Spatial, Object, Goal and Long suites. **LIBERO-Plus** (Fei et al., 2025) expands these suites into 10,030 perturbed tasks across seven dimensions. We evaluate the LIBERO-trained model without further training; the detailed transfer protocol and breakdowns are provided in Appendix E.

Baseline provenance and per-suite protocols. RMBench third-party rows come from the papers cited in the text and EventVLA is quoted in its visual-anchors-only configuration. RoboMME rows follow the benchmark report, except the third-party WeaveLA on its own retrained $\pi_{0.5}$ stack. RoboMemArena rows follow the benchmark report, with the external FrameSamp+Modul entry

Table 9: **Real-world results by initial configuration.** Each entry reports successes over ten autonomous trials. Cover Blocks labels list colors from far to near. Put Back Block labels follow the task coordinate convention.

Cover Blocks	RGB	GRB	GBR	RBG	BRG	BGR	Overall
Successes/trials	8/10	6/10	7/10	6/10	5/10	3/10	35/60

Put Back Block	Up	Down	Left	Right	Overall
Successes/trials	7/10	9/10	4/10	8/10	28/40

quoted from the official leaderboard and evaluation run under the official protocol of 51 rollouts per task. MIKASA-Robo and LIBERO-Plus VLA baselines follow the MemoryVLA++ report (Shi et al., 2026b) and GMP numbers come from its Figure 11 with per-task training and tuned horizons. LIBERO numbers for Diffusion Policy, Octo, OpenVLA, π_0 and π_0-FAST follow the OpenVLA-OFT report (Kim et al., 2025) at 500 trials per suite and RIPT-VLA is quoted at its reported four-suite average. RMBench uses 100 evaluation seeds per task. Instructions are frozen to each benchmark's own task strings throughout, so no prompt is tuned per method.

Two engineering bounds delimit how far the native window stretches. The backbone's 262k position limit corresponds to about 48 min of history at the deployment sampling rate. The vision processor further enforces a total pixel budget that would shrink per-frame resolution once the window exceeds a few minutes, which we lift at runtime by scaling its longest-edge budget with the frame count so that every frame keeps the production resolution.

C Real-World Evaluation Details

Robot and observations. We use a dual-arm Piper robot equipped with one head-mounted camera and two wrist-mounted cameras. Following the input construction in Section 3, head-camera observations provide the visual history, while the two wrist cameras provide current observations.

Tasks and evaluation. In Cover Blocks, the robot first covers all three colored blocks with identical lids and then uncovers them in red–green–blue order. In Put Back Block, the robot moves the block from its initial mat to the center, presses the button, retrieves the block and returns it to the same mat. We evaluate all six color arrangements for Cover Blocks and four initial positions for Put Back Block, with ten autonomous trials per configuration.

Fine-tuning data. Fine-tuning uses 180 real-robot demonstrations for Cover Blocks and 308 for Put Back Block.

Deployment. Following the RMBench action representation, the policy predicts 14-dimensional actions comprising absolute joint-position targets and one gripper command per arm. Each prediction produces a chunk of 30 actions, of which the first 16 are executed. Actions are executed at 25 Hz, matching the sampling rate of the demonstration data. The head-camera history grows with the episode up to a 60 s cap, after which it retains the most recent 60 s. History is sampled at 2 fps, yielding at most 120 frames, and processed using exact streaming inference.

Figure 6 shows additional successful autonomous rollouts. These selected examples are separate from the aggregate evaluation in Table 9.

D Full RoboMME Results

Table 10 reports the complete task-level RoboMME results underlying Table 3, including the per-task scores of our controlled variants, and Figure 7 plots the three rebuilds against the native stream. Figures 9 and 10 visualize the benchmark pool.

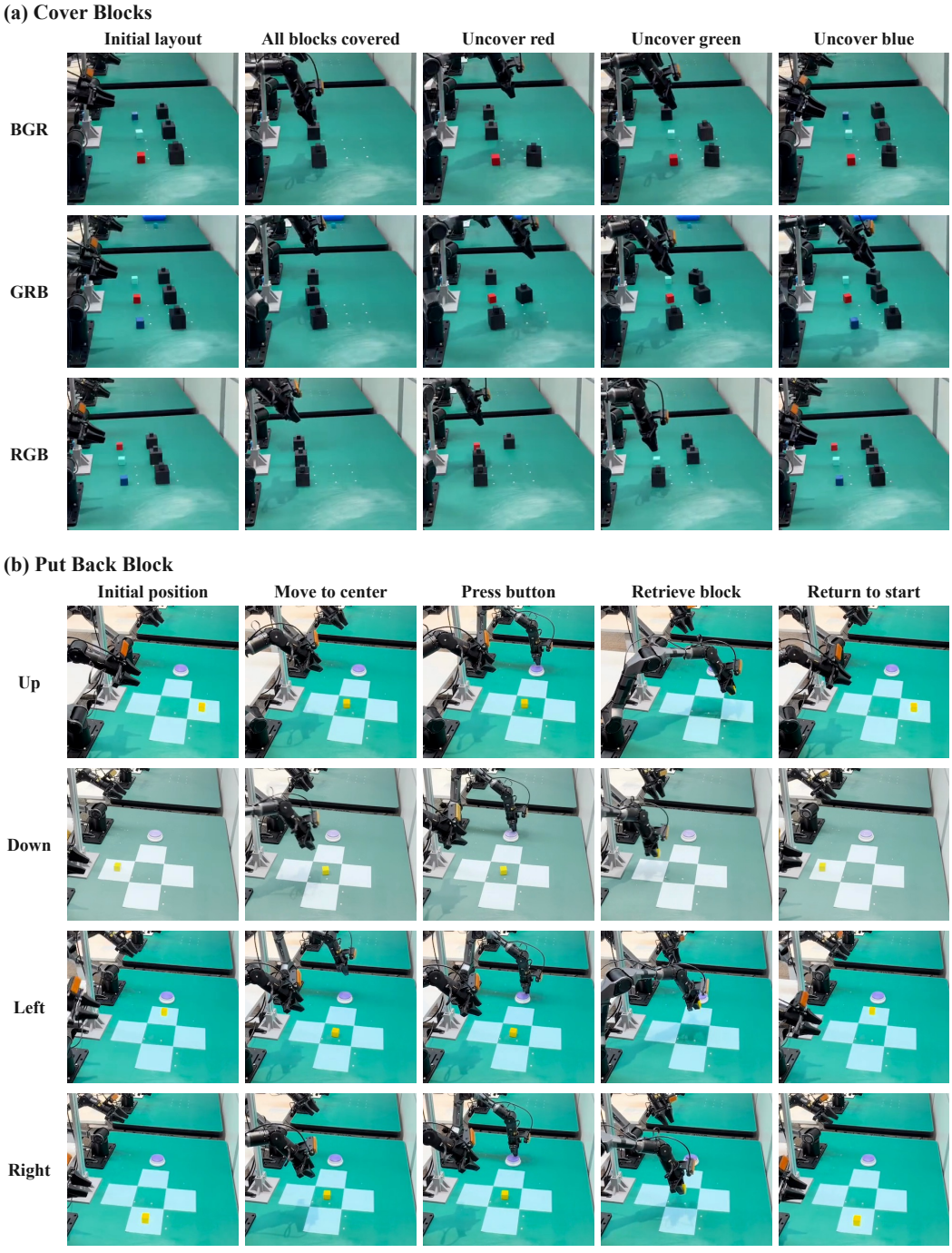


Figure 6: **Additional real-world autonomous rollouts.** (a) Cover Blocks under three initial color arrangements, labeled from far to near. (b) Put Back Block from four initial positions, labeled using the task coordinate convention rather than camera-image directions. Each row shows chronological frames from one successful rollout.

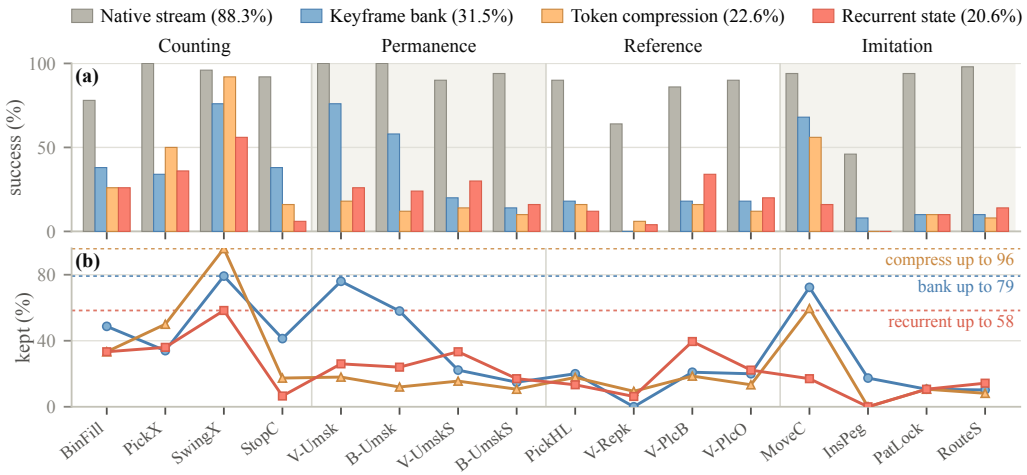


Figure 7: **Task-level effects of restricted memory interfaces on RoboMME.** The three controlled variants differ from SimpleMemVLA only in how history enters the model. *(a)* Success rates across 16 tasks grouped by benchmark dimension. *(b)* The same results normalized to native-context performance. Retrieval, token compression and recurrent state retain at most 79%, 96% and 58%, respectively, with the token-compression peak confined to the count task SwingXtimes. Results are from Table 10, with $n = 50$ episodes per task.

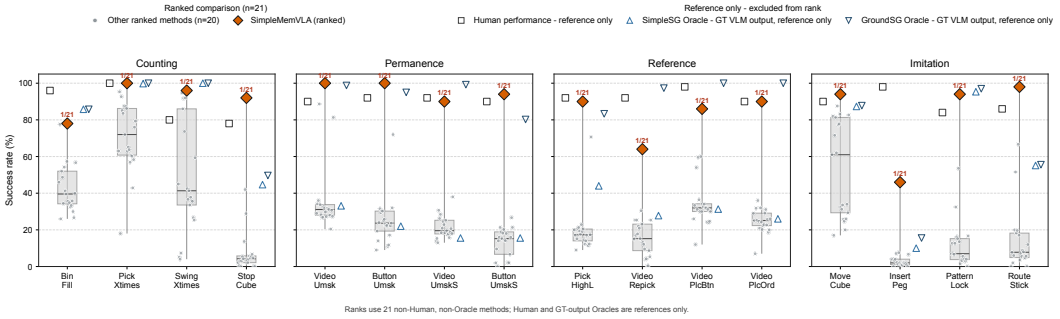


Figure 8: **First on all sixteen RoboMME tasks among the 21 deployable methods.** Gray dots denote the 20 other non-Human, non-Oracle methods evaluated by the benchmark. Boxes show the interquartile range of the 21-method ranked pool, with median and full range. Orange diamonds mark SimpleMemVLA and labels report its competition rank out of 21. Human performance and the two symbolic Oracle variants, which replace VLM outputs with ground-truth outputs, are shown only as references and are excluded from all ranks and distribution statistics.

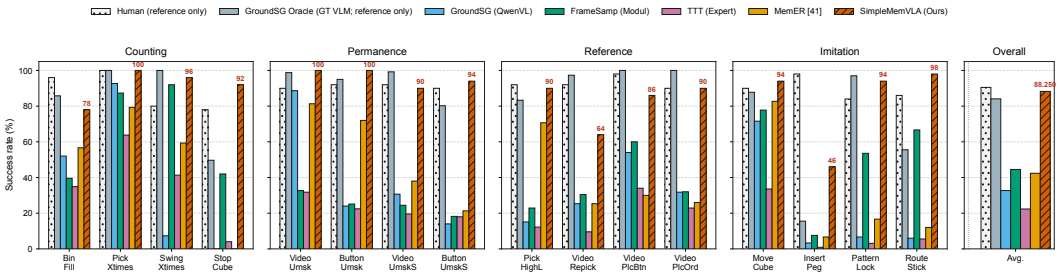


Figure 9: Task-wise RoboMME success rates (%, higher is better) for representative methods. Panels group the 16 tasks into Counting, Permanence, Reference and Imitation. Overall is recomputed from all 16 task scores. Human and GroundSG (Oracle, using ground-truth VLM outputs) are shown only as references and are excluded from the ranked comparison. For each remaining baseline family, the row with the highest overall AVG is shown. Values above orange hatched bars report SimpleMemVLA scores.



Figure 10: Complete task-wise RoboMME success rates (%, higher is better) for all 24 rows. Tasks follow the source-table order and methods retain the same order in every panel. Colors encode memory families. The orange hatched bar denotes SimpleMemVLA.

Table 10: Complete task-level RoboMME success rates (%). Human and GT-VLM Oracle rows (gray) are reference-only and excluded from ranking. **Bold** and underline mark ranks 1 and 2 among the remaining 21 methods. Task abbreviations follow Table 3's four dimensions in source order. The three SimpleMemVLA variant rows are the controlled within-stack rebuilds of Section 4.4 and are excluded from ranking.

Memory	Method	Integr. / VLM	Counting				Permanence				Reference				Imitation				AVG
			BinFill	PickX	SwingX	StopC	V-Umsk	B-Umsk	V-UmskS	B-UmskS	PickHL	V-Repk	V-PlcB	V-PlcO	MoveC	InsPeg	PatLock	RouteS	
Human	Human Performance	-	96	100	80	78	90	92	92	90	92	98	90	90	98	84	86	90.5	
Symbolic (Oracle)	SimpleSG (Dai et al., 2026)	GT VLM	85.78	99.78	100	44.67	33.11	22	15.56	15.56	44	27.78	31.33	26	87.33	10	95.33	55.11	49.58
Symbolic (Oracle)	GroundSG (Dai et al., 2026)	GT VLM	85.78	100	100	49.67	98.78	95	99.22	80.22	83.33	97.33	100	100	87.78	15.56	97	55.56	84.08
Symbolic	SimpleSG	Gemini	46	63	45	2	29	9	14	2	20	15	26	29	61	4	7	0	23.25
Symbolic	SimpleSG	QwenVL	77.56	95.33	5.11	0.44	34.22	19.33	15.33	9.56	17.11	25.33	33.33	25.11	82	3.78	12.67	7.78	29
Symbolic	GroundSG	Gemini	26	18	4	3	36	14	13	0	9	17	12	7	17	0	7	2	11.56
Symbolic	GroundSG	QwenVL	52	92.67	7.33	0	88.67	24	30.67	14	15.11	25.33	54	31.78	71.56	3.33	6.67	6	32.7
Perceptual	TokenDrop (Dai et al., 2026)	Context	48.67	85.11	94.67	3.11	33.78	31.56	26.22	16	20.67	17.78	31.11	25.33	81.33	4	12.67	20	34.5
Perceptual	TokenDrop	Modul	34.44	83.56	86	5.33	28.22	29.33	28.44	21.33	21.33	22	59.56	36	62	7.11	32.44	51.56	38.04
Perceptual	TokenDrop	Expert	54.22	87.56	91.78	4.22	26.67	30.44	18.44	18.89	19.33	20.89	36.44	24.67	87.56	2.22	16.22	18.22	34.86
Perceptual	FrameSamp (Dai et al., 2026)	Context	41.22	72	73.67	13.67	26.89	30.22	20.89	15.22	17.67	15.22	30	20.89	77.22	1.22	15.22	19.67	30.68
Perceptual	FrameSamp	Modul	39.56	87.33	92	42	32.67	25.11	24.44	18.22	22.89	30.44	60	32	77.78	7.56	53.56	66.67	44.51
Perceptual	FrameSamp	Expert	57.33	86.22	94.67	28.89	31.78	25.78	22.89	20.22	19.11	23.11	30	24.22	83.11	2	13.56	17.11	36.25
Recurrent	TTT (Dai et al., 2026)	Context	35.56	62.89	42.44	3.33	29.78	22.89	18.44	14.44	20.44	13.11	34.22	20.22	32.44	1.11	1.56	3.56	22.28
Recurrent	TTT	Modul	34.22	65.11	36.67	2.11	27.22	22.11	25.22	14.11	14.56	12.11	32.67	22.33	31.22	1.11	3.56	7	21.96
Recurrent	TTT	Expert	34.89	63.78	41.33	4	31.78	22.44	19.56	18	12.22	9.56	34	22.89	33.56	0.89	3.11	5.56	22.35
Recurrent	RMT (Dai et al., 2026)	Context	32.44	56.89	33.56	5.78	31.33	10.89	17.33	2	14	3.78	32	29.11	25.78	2	5.56	8.89	19.46
Recurrent	RMT	Modul	33.33	60.78	37.78	4.67	31.11	11.78	17.78	2.44	17.11	4.22	32	31.11	24.67	2.21	3.78	8	20.17
Recurrent	RMT	Expert	35.78	60.22	36	5.56	28	17.11	15.78	2	11.78	0.22	24.22	22.67	20	1.89	4.22	4.89	18.15
Other	$\pi_{0.5}$ (Intelligence et al., 2025)	-	30	42.89	35.56	6.67	20.44	22.22	18.67	6.67	11.33	0.44	31.11	25.78	26	1.56	2.89	4.67	17.93
Other	$\pi_{0.5}$ w/ past actions (Dai et al., 2026)	-	26.67	58.33	26.67	4.67	30.67	23.67	20.67	16	12.33	8.67	24	18.67	34	1	4	5.67	19.73
Other	SAM2Act+ (Fang et al., 2025)	-	40	76	25.33	0	27.33	32	18	26.67	17.33	5.33	24.67	20	29.33	0	0	0	21.37
Other	MemER (Sridhar et al., 2026)	-	56.67	79.33	59.33	0	81.33	72	38	21.33	70.67	25.33	30	26	82.67	6.67	16.67	12	42.38
Ours variant	SimpleMemVLA (Retrieval)	-	38	34	76	38	76	58	20	14	18	0	18	18	68	8	10	10	31.5
Ours variant	SimpleMemVLA (Token Compression)	-	26	50	92	16	18	12	14	10	16	6	16	12	56	0	10	8	22.625
Ours variant	SimpleMemVLA (Recurrent State)	-	26	36	56	6	26	24	30	16	12	4	34	20	16	0	10	14	20.625
Ours	SimpleMemVLA	-	**78**	**100**	**96**	**92**	**100**	**100**	**90**	**94**	**90**	**64**	**86**	**90**	**94**	**46**	**94**	**98**	**88.25**

Table 11: **LIBERO-Plus breakdowns.** Top: by source LIBERO suite. Bottom: by the benchmark's five difficulty levels, which stratify tasks by the accuracy of four reference models (Fei et al., 2025). The 121 tasks that carry no difficulty label in the benchmark metadata are omitted from the bottom panel.

By source suite					
	Spatial	Object	Goal	Long	
Tasks	2,402	2,518	2,591	2,519	
Success (%)	81.8	83.9	75.0	73.1	
By difficulty level					
	Level 1	Level 2	Level 3	Level 4	Level 5
Tasks	1,644	2,202	2,094	1,886	2,083
Success (%)	90.5	88.1	83.4	74.3	56.3

Table 12: **The Sensor Noise column of Table 7, split by corruption type and severity tier.** Types are encoded in the benchmark's task naming. Success rates in %.

Corruption type	Overall	Low	Mid	High
Motion blur	87.8	100	95	76
Glass blur	85.7	93	89	79
Gaussian blur	69.2	95	78	47
Zoom blur	62.5	96	77	44
Fog	60.7	93	80	39

E LIBERO-PLUS: PROTOCOL DETAILS AND BREAKDOWNS

Protocol. We follow the official LIBERO-Plus protocol (Fei et al., 2025): one trial per task over all 10,030 tasks (zero exclusions), a frozen initial state and fixed environment seed per task, ten no-op settling steps, per-suite step budgets of 220/280/300/520 and success judged by the simulator's built-in check. The seven dimensions differ in task counts (Camera 1,599, Robot 1,550, Language 1,537, Light 1,142, Background 1,076, Noise 1,601, Layout 1,525), so the Total in Table 7 is the success rate pooled over all tasks rather than a mean of the seven columns. One detail matters when reading the Language column: for the six non-language dimensions the official instruction string appends the perturbation tag to the task instruction (e.g., "... table 1"), which we feed verbatim. Only the Language dimension replaces the instruction with a rewritten one, so that column specifically measures robustness to paraphrase.

By suite and difficulty. Table 11 shows that the two more semantic suites degrade most under perturbation: relative to their unperturbed counterparts in Table 6, Goal loses 23.0 points (98.0 to 75.0) and Long 21.9 (95.0 to 73.1), versus 16.4 for Spatial and 14.9 for Object. Across the benchmark's difficulty stratification, success decays monotonically from 90.5% at Level 1 to 56.3% at Level 5. This graded degradation, rather than a bimodal solved/unsolved split, indicates that the 78.4% total is not carried by the easy strata alone.

What the noise split reveals. The 74.2% Sensor Noise column is not a uniform weakness. Split by corruption type (Table 12), the dividing line is not static versus dynamic corruption but whether the corruption preserves the geometric evidence in the visual stream. Motion blur and glass blur smear textures and edges while leaving the global scene layout intact and are largely absorbed (87.8% and 85.7%). Zoom blur superimposes rescaled copies that shift objects' apparent positions. Fog injects a per-frame random low-frequency luminance field that makes the history window temporally inconsistent. Gaussian blur at high severity erases object boundaries outright. Success collapses exactly on these three (62.5%, 60.7% and 69.2% overall, falling to 44%, 39% and 47% at the highest severity). All five types degrade monotonically with severity, indicating graded loss of evidence rather than a threshold artifact. SimpleMemVLA leads on Camera, Light and Background perturbations, while performance remains weaker on Language and several sensor-noise conditions.

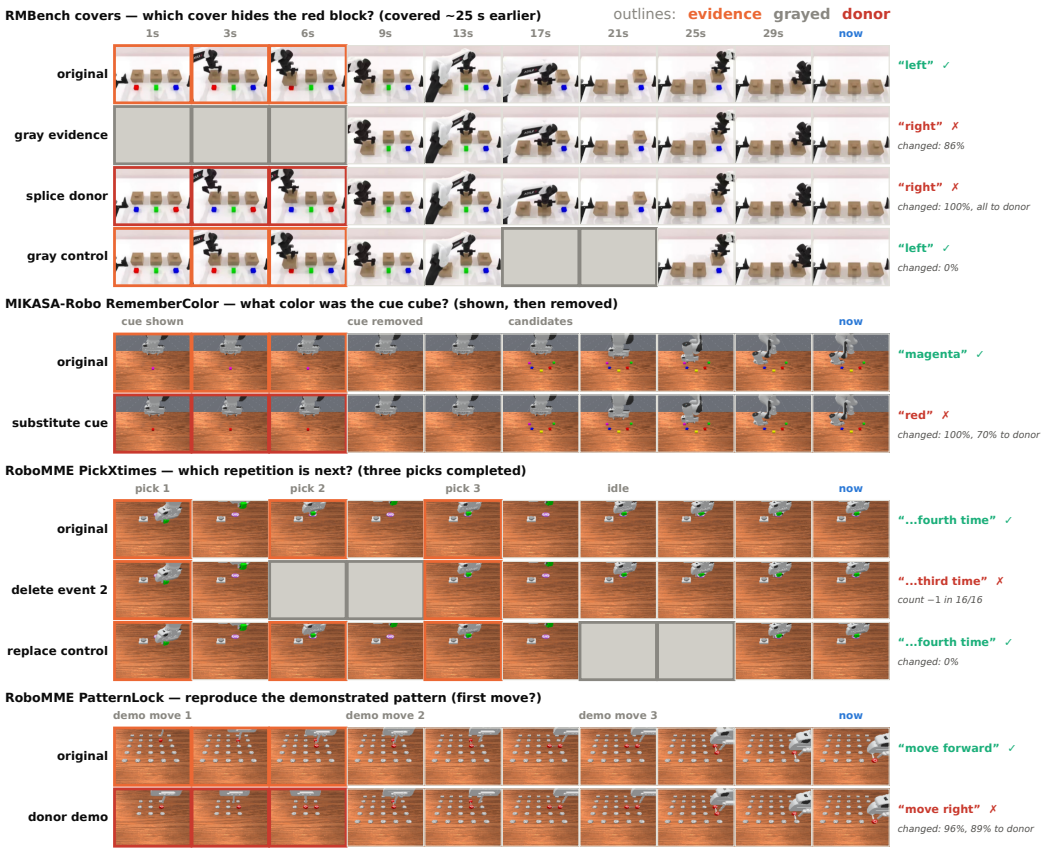


Figure 11: **History interventions redirect the output across all suites.** Blocks denote benchmarks and rows show histories from oldest to most recent, processed through the unchanged deployment pipeline. Orange borders mark evidence frames, gray fills evidence ablations and red borders donor-episode insertions. Removing evidence changes the output, donor evidence redirects it toward the donor content and matched control edits leave it unchanged.

F Additional Analysis

F.1 History Intervention Examples

Figure 11 presents the evidence-masking, donor-replacement and matched-control examples discussed in Section 4.5.

F.2 Ablation Results

Figure 12 summarizes the history and conditioning ablations discussed in Section 4.7.

F.3 The Gains Track Memory Load

If the gains came from a generically stronger backbone or better low-level control, they would be flat across memory loads. Instead the advantage grows along each suite's own memory axis (Tables 2, 3 and 4). On RMBench the margin over Mem-0 widens from +38.8 on single-memory tasks to +68.5 on multi-memory ones. On RoboMME it grows from +24.7 on Counting, whose periodic cues stay partly visible, to +44.5 on Reference, whose evidence is entirely off-frame. On MIKASA-Robo the lead over the best prior VLA stretches from 12 to 39 points as RememberColor moves from 3 to 9 candidates. Where memory is not the bottleneck the margin does not reverse but vanishes, since on

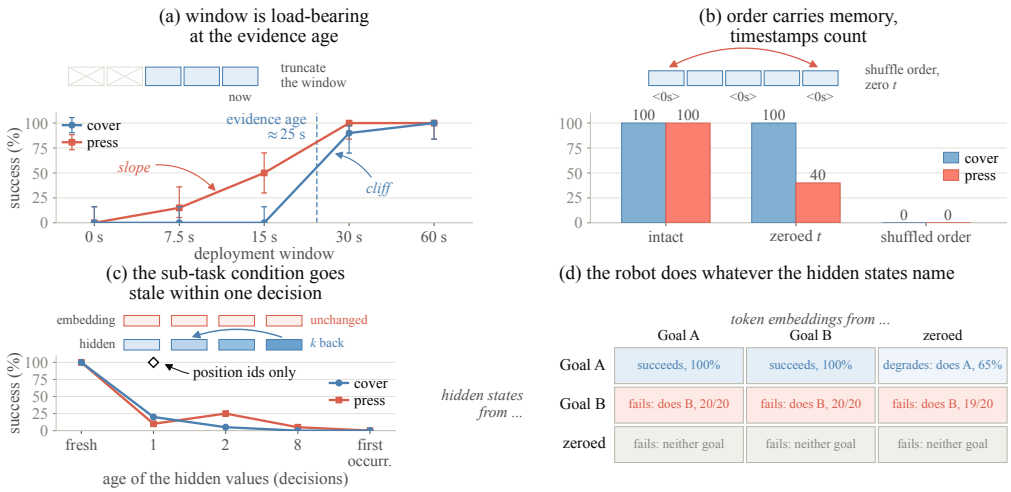


Figure 12: **Ablations of native-context memory.** *(a)* Cover-blocks fails when the covering event leaves the window. Press-button degrades gradually as completed presses disappear. *(b)* Both require frame order. Timestamps mainly support counting. *(c)* Stale sub-task hidden states sharply reduce success. Equally stale position IDs leave it unchanged. *(d)* Behavior follows hidden states rather than token embeddings, identifying contextual hidden states as the memory-to-action interface.

LIBERO the same recipe is simply on par with the best reactive VLAs at 97.5%. This dose–response pattern ties the gains to memory itself.

The same lens delimits the claim. SimpleMemVLA's residual failures are consistently not retention failures: they concentrate in precision control (Insert Peg at 46%, where even the perfect-perception oracle reaches only 15.6%, Table 10), in fine-grained re-identification (Observe&Pickup 65%, RememberColor-5 58%, Video Repick 64%) and in perturbations that rewrite the instruction or attack positional evidence itself (Appendix E). Feeding the policy its own past removes the need for memory machinery. It does not substitute for perception or control.

F.4 Why the Machinery Falls Short

The controlled comparison of Section 4.4 leaves a puzzle: the prior designs are far more elaborate than ours, yet they lose by wide margins. The following comparisons and interventions provide complementary evidence about the historical information needed at decision time.

The first is a controlled comparison that the RoboMME benchmark itself provides, since its fourteen memory variants share one $\pi_{0.5}$ backbone and differ only in the mechanism. The pool covers all three families of Figure 1, with external stores in both symbolic and retrieval form: symbolic scene-graph stores reach 11.6 to 32.7%, the retrieval-based MemER 42.4%, learned compression 30.7 to 44.5% and recurrent state 18.1 to 22.3%, against 88.3% for attending to the timestamped stream itself (Table 3). Our own rebuilds compare alternative memory interfaces on a shared backbone and training stack, reaching 20.6–31.5% against 88.3% for native context. These results characterize the evaluated implementations under the training protocols described in Appendix B. More machinery does not help: the recurrent family's best variant stays at 22.3% and the lowest scores in the table belong to symbolic stores with a VLM in the write loop. The ordering holds beyond RoboMME as well: the strongest prior memory VLA reaches 44.4% on MIKASA-Robo where the raw stream reaches 74.0% and 73.1 against 78.4% zero-shot on LIBERO-Plus (Tables 4 and 7).

The second experiment removes write quality as the explanation. GroundSG handed ground-truth VLM outputs is a write-time abstraction executed perfectly, with no perception error and no capacity limit, yet it reaches 84.1%, below the 88.3% of the raw stream. The ceiling of this abstract-then-store pipeline sits below the input it abstracts, because what the write discards before the need is known stays missing at read time. Running the same pipeline with real perception loses another fifty points (32.7%) and the strongest deployable baselines collapse exactly where the needed evidence is a specific past percept: MemER falls to 38.0 and 53.2% on Reference and Permanence, Mem-0 from

52.8 to 28.5% as the number of facts grows and MemoryVLA++ from 97% on ShellGame to 16% on RC-9, where one cue must stay distinct from nine alternatives.

The third line of evidence comes from the deployment edits of Sections 4.5 and 4.7, which name what each abstraction discards. Shuffling frames zeroes both probe tasks and a retrieved set of keyframes is exactly an unordered set. The success cliff at the evidence age shows the read needs the stale frames that recency-biased compression evicts first. Zeroed timestamps halve counting and neither summaries nor recurrent states keep a time ruler. The graying interventions show the read consumes the percept itself, which no symbolic entry can reproduce. The recurrent rebuild compresses visual content, temporal order, timestamps and older evidence into a fixed sixteen-token state. It reaches 20.6% overall and is strongest on Counting at 31.0%, consistent with partial retention of running totals but not with reliable preservation of the perceptual and temporal evidence required across the suite. The keyframe bank discards only the order and the timestamps while keeping the percepts, so its 31.5% is both higher and selective in the way this list predicts, since the tasks it survives are the ones whose answer needs no time ruler. The compressor squeezes the percepts through a fixed budget, the very channel the graying interventions single out, and its 22.6% is selective in the complementary direction, keeping the tallies a summary can hold while the percept-dependent tasks the bank kept raw collapse. The one edit that never hurts is the control that keeps the evidence frames ordered and intact while removing others, so selection as such is not the problem. Discarding before the need is known is.

Nor does the dilemma leave a lossless way out. A write format that keeps the order, the timestamps, the stale frames and the percepts themselves has stopped abstracting, because up to the sampling rate it is the stream. A store that keeps everything and defers every choice to read time escapes the dilemma, but read-time selection over an intact past is what attention over the window already does, with no store to maintain. Attention over the raw stream wins by refusing the choice every tested family made: it defers selection to read time, when the need has arrived.

G Bounded-Cost Streaming with Sliding-Window Attention

Streaming Inference with Prefix Prefill keeps decision latency close to the single-frame baseline for minute-scale windows. Its cache size depends on the retained context length and remains bounded under the standard 60 s cap. The SWA variant maintains a persistent stream with block-wise cache eviction.

Window the softmax layers, and only them. The backbone interleaves two kinds of sequence mixing: linear-attention layers, whose recurrent state is constant-size and summarizes the entire stream by construction, and softmax-attention layers, the only place where per-token state accumulates (24 and 8 of the 32 layers, respectively). SWA therefore touches nothing but the 8 softmax layers. During training and at deployment alike, each of their queries attends to at most the most recent L tokens; the linear-attention layers are left untouched and keep integrating every token since the first frame of the episode. The window is sized in whole history units, where one unit is the token block of one temporal patch: its plaintext timestamp (6–8 tokens), the vision delimiters and 80 video tokens, 88–90 tokens in all. We set the window to $M = 60$ units, exactly the 60 s memory span the recipe already deploys on RMBench (Table 8), plus the decision suffix and the generation budget, giving $L = 5{,}888$ tokens with margin. Inside the window nothing changes: selection over the explicit past remains read-time attention, the thesis of Appendix F.4. Evidence older than the window survives only through the linear-attention state, the same finite-span caveat the windowed recipe always had, now with a compressed residue beyond it instead of nothing.

Timestamps must be absolute. A streamed token may never change after it is emitted, otherwise its cached key–value entries are invalid the moment the window slides. The window-relative timestamps of the deployed recipe violate exactly this: one decision later the same physical frame carries a label smaller by one execution interval, so every unit would need re-encoding at every step. The SWA variant therefore switches the timestamp convention from window-relative to episode-absolute, computed from each frame's index on the episode's sampling grid through one function shared by training and deployment. Under absolute time a unit's tokens are immutable, its cache entries are written once and reused until evicted and positions continue monotonically without ever being renumbered.

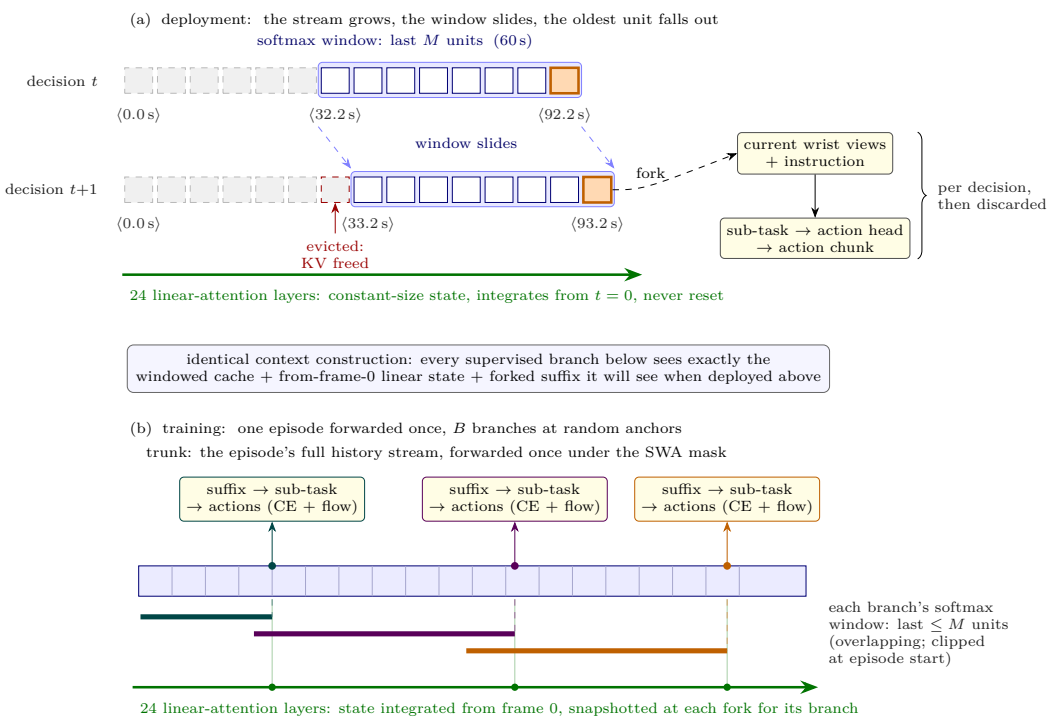


Figure 13: **The SWA variant: one context construction, trained and deployed.** **(a)** Two consecutive streaming decisions: each appends one unit (orange) to the persistent stream, the softmax window slides by one unit, and the unit leaving it (red) has its cache rows freed; the linear-attention state (green) integrates from $t = 0$ and is never reset. **(b)** Training forwards each episode once as a trunk and supervises B branches at random anchors, each attending to its last $\leq M$ units and resuming the linear state snapshotted at its fork — exactly the deployed construction. Details in the accompanying text.

The conditioning interface needs no change. The action head conditions only on the generated answer span and the state token (Equation 3), which sit at the tail of the sequence and hence always inside the window, so the train/deploy isomorphism of Section 3.3 carries over unchanged.

Deployment: append one unit, evict one unit. Figure 13a shows one decision. The persistent stream holds only the history video channel. When a decision fires, the newest temporal patch is encoded into one unit and appended (positions continue from the previous tail), softmax-cache rows older than the window are dropped in whole units and the linear-attention states are simply kept. The decision suffix, current wrist views, instruction and chat scaffold, is forked off the stream state, the sub-task is decoded, the action head conditions on its span and the fork is discarded. Eviction is exact rather than approximate: a dropped row is outside the window of every query that will ever be computed, so the streamed decision reproduces the full-prompt SWA decision token for token, which we verify at integer exactness for token and position ids and at kernel-noise level for activations. Per decision the variant therefore encodes one temporal patch, prefills one unit and decodes one short answer against a bounded cache: compute and memory are both independent of how long the robot has been running.

Training with episode-level packing. Training must expose the model to the same context construction, and the subtle point is the linear-attention layers. Ordinary per-anchor training renders each anchor a finite window and integrates the recurrent state from that window's first frame, while the streaming deployment integrates it from the episode's first frame, a train/deploy mismatch in exactly the pathway SWA makes load-bearing. SWA training therefore packs samples at the episode level (Figure 13b). One sample is an (episode, phase) pair, where a phase is one residue of the temporal-patch grid, so that every anchor of the sample forks on a unit boundary of one shared trunk

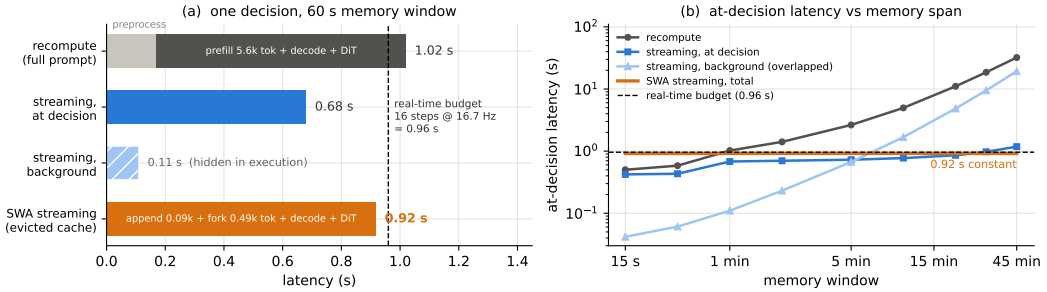


Figure 14: **Figure 5, extended with the SWA streaming variant.** Measurements use one H100, bf16 and batch size 1, with the setup of Figure 5. **(a)** The SWA decision takes 0.92 s, within the 0.96 s execution interval. **(b)** The SWA variant uses a fixed $L = 5{,}888$-token window to bound its resident cache.

Table 13: **Closed-loop success of the SWA streaming deployment**, on the RMBench protocol of Table 2 (100 held-out seeds per task).

	Single-memory $M(1)$						Multi-memory $M(n)$						
Method	Obs&PU	Rearr.	PutBack	SwapB	SwapT	*Avg*	Battery	Rank.	Cover	Press	Place-Mat†	*Avg*	Overall
SimpleMemVLA (Table 2)	65	100	100	100	93	91.6	90	100	98	100	100	97.0	94.0
+ SWA streaming (evicted cache)	56	100	100	100	93	89.8	71	100	99	100	99	92.5	91.0

rendering. The trunk, the episode's full history stream, is forwarded once with the SWA mask on the softmax layers and the linear-attention state running from frame 0; $B = 8$ anchors drawn from the phase's chain fork off it, each resuming from the linear-attention states snapshotted at its fork and attending to the last M units of trunk cache, and each pays the standard losses of Section 3.3. The packed pass is not an approximation: token and position ids match the per-anchor construction exactly and the supervised span's hidden states match within bf16 kernel noise, while the shared trunk cuts per-anchor compute several-fold. Random anchor draws with repeats cover every anchor near-uniformly across epochs. Relative to the recipe of Section 3, the SWA variant changes only the attention mask, the timestamp convention and the packing.

Cost. Figure 14 places the SWA variant on the axes of Figure 5. The long-input experiment measures the cost of increasing the retained context length; its 45 min input approaches the backbone's 262k-token limit. Standard deployment instead caps the history window at 60 s. With a fixed $L = 5{,}888$-token attention window, the SWA variant bounds the resident cache and the number of tokens processed per decision. Its measured median decision latency is 0.92 s over 153 consecutive decisions.

Results. Table 13 evaluates the SWA streaming deployment on the full RMBench protocol. Seven of the nine scored tasks stay within one point of the Table 2 deployment (five at 100%), and the overall score is 91.0 vs. 94.0. The remaining difference is concentrated in the two hardest memory tasks (Obs&PU and Battery) and reflects training under the SWA regime rather than the streamed inference itself, which reproduces the policy's full-prompt decisions exactly (verified at integer exactness for token and position ids and at kernel-noise level for activations, as argued above). The variant thus trades a small amount of accuracy on the hardest memory tasks for a deployment whose per-decision compute and memory are constant in episode length.