SEGMEM: SEGMENT-WISE PROCEDURAL MEMORY WITH FUTURE-AWARE GATING FOR LONG-HORIZON ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon robotic manipulation requires conditioning on procedural history. However, naively accumulating all past observations causes inference latency to grow unboundedly, which is intolerable in the real world. This necessitates organizing manipulation memory under a strictly controlled context budget. Prior work typically addresses this by either retaining sparse visual frames, risking collapse when a critical evidence is omitted, or repeatedly compressing the full interaction into a global latent state, where high-frequency read or write cycles induce information interference. These trade-offs highlight *temporal granularity* as a central design problem. Our pilot study reveals that segmented recurrence mitigates global state interference, yet its effectiveness remains sensitive to the chosen granularity. To this end, we propose SEGMEM, a segment-wise memory architecture that recurrently compresses variable-length observation subsequences into fixed-size segment slots. Crucially, to overcome the sensitivity of manual granularity, SEGMEM employs a future-aware gate that learns when to close an active segment based on its anticipated utility for subsequent action prediction. Under a fixed budget, SEGMEM significantly outperforms prior methods across both simulation benchmarks and real-world experiments. Further analyses demonstrate that the learned gate surpasses alternative segmentation strategies and remains robust under temporal shifts, establishing adaptive temporal organization as a key principle for memory-augmented robotic manipulation.

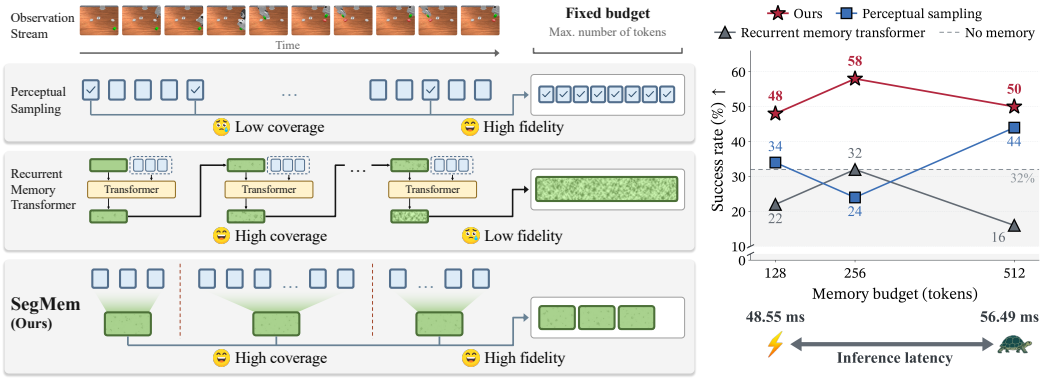


Figure 1: **Comparison of our SEGMEM with prior studies.** Existing policies face a fidelity–coverage trade-off: perceptual sampling misses evidence, while global recurrence causes representational interference. In comparison, our SEGMEM introduces a segment-wise memory that compresses variable-length observation segments into fixed-size latent states, achieving both high coverage and fidelity. On RoboMME (Dai et al., 2026), SEGMEM significantly outperforms all baselines, surpassing 512-token competing methods using merely 128 tokens.

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.

1 INTRODUCTION

Vision-language-action (VLA) models have substantially advanced language-conditioned robotic manipulation by transferring semantic knowledge from pretrained vision–language models to visuomotor control (Zitkovich et al., 2023; Kim et al., 2025; Intelligence et al., 2025b; Black et al., 2026). Despite these advances, most VLA policies remain predominantly reactive, conditioning action prediction primarily on the current visual observation. They therefore struggle with long-horizon tasks whose decisions depend on procedural information carried by the interaction history—what the robot has previously observed, attempted, or completed. Meanwhile, closed-loop robotic control imposes strict latency requirements, placing a practical limit on the amount of historical context available to the policy. This work therefore focuses on how to organize extended interaction histories for VLA policies under a specified context budget.

Existing memory-augmented VLA policies can be broadly categorized into two paradigms, reflecting a fidelity–coverage trade-off. On the one hand, explicit history-memorized methods retain sampled observations or selected keyframes (Sridhar et al., 2025; Yang et al., 2026; Zeng et al., 2026), preserving fine-grained visual details but risking omitting decision-critical evidence under the given context budget. On the other hand, latent historical state methods compress extended interaction histories into memory tokens or recurrent states (Shi et al., 2026; Li et al., 2026; Cherepanov et al., 2026; Liu et al., 2026), enabling a bounded representation to summarize the entire interaction. However, repeatedly consolidating information from different task phases into a shared latent state may induce representational interference, whereby newly integrated observations can obscure information accumulated during earlier phases.

In this work, we aim to retain historical information across interactions while mitigating this representational interference. Since recurrent memory is good at modeling extended histories by incrementally integrating streaming observations into contextualized representations, we investigate segmented recurrence, which restricts recurrent integration to local temporal segments while preserving completed segment states separately. Our motivating study yields two key findings: segmented recurrence improves action prediction and task success over a single global recurrent state, yet its effectiveness depends on the segmentation granularity, with different tasks favoring different granularities. This indicates that the benefit of segmentation depends not only on maintaining separate memory states, but also on where their boundaries are placed. Together, these findings motivate an adaptive segmentation mechanism rather than a fixed temporal schedule.

Based on these insights, we propose SEGMEM, a segment-wise recurrent memory architecture that compresses variable-length subsequences of the observation stream into compact slots while retaining completed segments as distinct historical states. A future-aware boundary gate determines whether to continue updating the active segment or close it and initialize a new one. To train this gate, we construct two counterfactual branches corresponding to the two update choices and leverage the difference in their future action-prediction losses as the supervisory signal. We evaluate SEGMEM on six representative RoboMME tasks (Dai et al., 2026), all eight LIBERO-Mem tasks (Chung et al., 2025), and two real-world tasks on an AgiBot G1 robot. Under a fixed budget, SEGMEM achieves state-of-the-art performance, exceeding the strongest baseline by 8.1 percentage points in simulation and 12.5 percentage points in the real world. Further analyses show that the learned gate significantly outperforms alternative segmentation strategies. Notably, it remains robust to temporal distribution shifts, continuing to align its boundaries with task-relevant transitions.

Our contributions are threefold:

- We formulate memory-augmented long-horizon manipulation as the problem of organizing extended interaction histories under a specified context budget, and demonstrate that providing additional history without effective organization can degrade action policies.
- We propose SEGMEM, which recurrently compresses variable-length observation streams into separate segment slots and learns their boundaries with a future-aware gate, balancing broad history coverage with the preservation of information required for subsequent decisions.
- We demonstrate consistent improvements across simulation and real-world tasks under different memory budgets, and further examine the effectiveness and robustness of the learned boundary gate through controlled analyses.

2 Related Work

Vision-Language-Action Models. Vision-language-action (VLA) models extend pretrained vision language models (Beyer et al., 2024; Bai et al., 2025) from multimodal understanding to embodied decision-making by grounding language instructions in visual observations and predicting executable robot actions (Zitkovich et al., 2023; Kim et al., 2025; Brohan et al., 2023). Recent advances in VLA modeling have further demonstrated strong generalization across diverse tasks and environments (Black et al., 2026; Intelligence et al., 2025b). Nevertheless, mainstream VLA policies remain predominantly *reactive*, inferring each action from the current observation or a short observation window. Consequently, they struggle with long-horizon manipulation tasks that require integrating procedural information over time.

Memory-augmented Policies for Long-horizon Manipulation. To exploit procedural information for decision-making in long-horizon manipulation, memory-augmented VLA policies have primarily followed two paradigms. The first retains a sparse set of high-fidelity frames from extended interaction histories (Intelligence et al., 2025a; Chen et al., 2026; Sridhar et al., 2025; Yang et al., 2026; Zeng et al., 2026). However, failing to retain task-relevant frames can lead to compounding errors and ultimately a sharp degradation in task performance. The second compresses interaction histories into compact latent states (Shi et al., 2026; Jiang et al., 2026; Zhu et al., 2026; Chen, 2026; Cherepanov et al., 2026; Li et al., 2026). Although latent memory can cover substantially longer interaction histories under the same memory budget, continually consolidating information from successive task phases into shared latent representations can induce representational interference.

3 Preliminaries and Pilot Study

3.1 Problem Formulation and Recurrent Memory

Problem Formulation. We consider language-conditioned robotic manipulation in a partially observable environment. At each decision step t, the robot receives the current multi-view observation $\mathbf{o}_t = \{o_t^v\}_{v=1}^V$ from V camera views, proprioceptive state s_t, and language instruction ℓ. Since long-horizon decisions may depend on procedural information beyond the current observation, we denote the complete interaction history preceding step t by $\mathcal{H}_t = \{(\mathbf{o}_\tau, s_\tau, \mathbf{a}_\tau)\}_{\tau<t}$, where $\mathbf{a}_\tau$ is the executed action chunk at step τ. We formulate a memory-augmented VLA policy as

$$\mathbf{a}_t \sim \pi_\theta(\cdot \mid \mathbf{o}_t, s_t, \ell, \mathcal{M}_t)\,, \tag{1}$$

where $\mathcal{M}_t = g_\eta(\mathcal{H}_t)$ denotes the policy-facing memory obtained by organizing the complete and growing interaction history $\mathcal{H}_t$ into at most B memory tokens, i.e., $|\mathcal{M}_t| \le B$. Here, B is the fixed memory-token budget, while η and θ parameterize the memory organizer and the memory-augmented VLA policy, respectively.

Recurrent Memory. We instantiate the memory organizer using a Recurrent Memory Transformer (RMT) (Bulatov et al., 2022). Let $\mathbf{X}_t = E_\psi(\mathbf{o}_t)$ denote the visual tokens extracted from the current observation by the encoder E_ψ, and let $\mathbf{M}_t^{\text{glob}} \in \mathbb{R}^{B\times d}$ denote the global recurrent state before incorporating $\mathbf{X}_t$, where d is the token dimension. At each memory-update step, the recurrent state and observation tokens are jointly processed as

$$[\,\mathbf{M}_{t+1}^{\text{glob}}; \widetilde{\mathbf{X}}_t\,] = T_\phi\Big([\,\mathbf{M}_t^{\text{glob}}; \mathbf{X}_t\,]\Big)\,, \tag{2}$$

where T_ϕ denotes the recurrent Transformer and $\widetilde{\mathbf{X}}_t$ denotes the contextualized observation tokens. The global recurrent state is projected into the VLA embedding space to produce the policy-facing memory, $\mathcal{M}_t = P_\rho(\mathbf{M}_t^{\text{glob}})$.

3.2 Pilot Study

Because a standard RMT consolidates the entire interaction history into a single global recurrent state and is therefore susceptible to representational interference, we conduct a pilot study to examine whether distributing the same memory budget across multiple recurrent segments improves history representation.

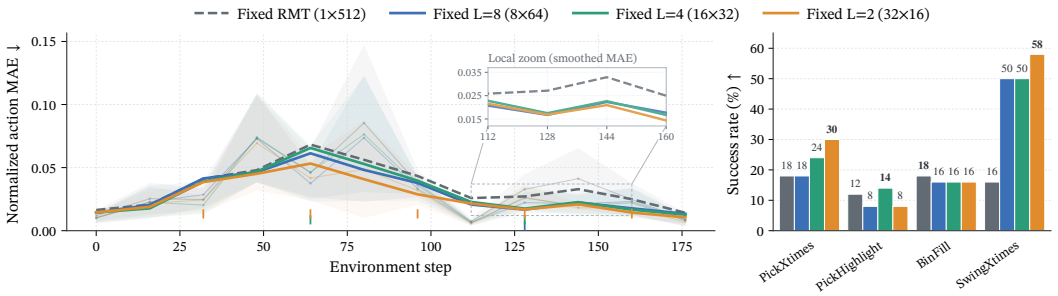


Figure 2: Pilot study results of recurrent-memory organization under a specified 512-token budget. **Left:** Normalized action MAE along a held-out execution for a global RMT and fixed segmented variants; lower is better. **Right:** Success rates across four history-dependent manipulation tasks. Each allocation is denoted by the number of segments $\times$ the number of tokens per segment.

Consider the N historical memory updates available before decision step t, indexed by $i \in \{1, \ldots, N\}$. Given a fixed segment length L, the active segment at update i is $j_i = \lfloor (i-1)/L \rfloor + 1$. Assuming that N is divisible by L, this produces $J_L = j_N = N/L$ segments. Under a total budget of B tokens, each segment is allocated $m_L = B/J_L$ tokens, such that $J_L m_L = B$. Let U_ϕ denote the segment-wise recurrent update induced by T_ϕ. The segmented memory is computed as

$$\mathbf{M}_i^{(j_i)} = U_\phi\left(\mathbf{M}_{i-1}^{(j_i)}, \mathbf{X}_i\right), \qquad \mathcal{M}_t^{(L)} = P_\rho\left([\,\overline{\mathbf{M}}^{(1)}; \ldots; \overline{\mathbf{M}}^{(j_N-1)}; \mathbf{M}_N^{(j_N)}\,]\right). \tag{3}$$

Here, $\mathbf{M}_i^{(j)} \in \mathbb{R}^{m_L \times d}$ denotes the state of segment j after update i, while $\overline{\mathbf{M}}^{(j)}$ denotes the archived state of a completed segment. A new segment is initialized from a shared learned state and archived upon reaching the next boundary. The resulting $\mathcal{M}_t^{(L)}$ serves as the policy-facing memory $\mathcal{M}_t$ in Eq. (1).

We train a separate memory-augmented VLA policy for a global allocation of 1×512 and each segmented allocation of 8×64, 16×32, and 32×16. We then run all policies on the same held-out trajectory and measure their normalized action-prediction error:

$$\mathrm{NMAE}_t^{(L)} = \frac{1}{H_a d_a} \left\| \widehat{\mathbf{a}}_t^{(L)} - \mathbf{a}_t^{\mathrm{gt}} \right\|_1, \tag{4}$$

where $\widehat{\mathbf{a}}_t^{(L)}$ and $\mathbf{a}_t^{\mathrm{gt}}$ are the predicted and ground-truth action chunks, respectively, and H_a and d_a denote the action-chunk horizon and action dimension. Further details are provided in Appendix A.1.

Limitations of Global Recurrent Memory. As shown in Figure 2, fixed-segmentation variants achieve lower action-prediction error than the global RMT throughout most of the trajectory. They also increase the mean success rate across the four tasks from 16% to 23%, 26%, and 28% for the L=8, L=4, and L=2 configurations, respectively. This pattern is consistent with the information bottleneck induced by repeatedly compressing a growing history into a single fixed-capacity state, which progressively impairs the retention and retrieval of earlier information as the interaction horizon increases (Jelassi et al., 2024; Merrill et al., 2024). Segmented recurrence instead archives completed states, limiting their exposure to subsequent updates and thereby mitigating representational interference from later observations.

Sensitivity to Segmentation Granularity. Despite the overall benefit of segmented recurrence, its performance remains sensitive to the fixed segment length L. As shown in Figure 2, shorter segments perform best on PICKXTIMES and SWINGXTIMES, whereas PICKHIGHLIGHT favors an intermediate granularity and BINFILL exhibits little variation across segmentation choices. Under a fixed memory budget, decreasing L provides finer temporal resolution by retaining more segment-level states, but allocates fewer memory tokens to each segment; increasing L offers greater within-segment capacity at the cost of coarser temporal organization. These differences suggest that task-relevant information is not uniformly distributed over time: some intervals contain rapid state changes that require fine-grained temporal separation, whereas others evolve slowly and can be summarized over longer spans (Zhang et al., 2024; Chen et al., 2024; Pagnoni et al., 2024). This

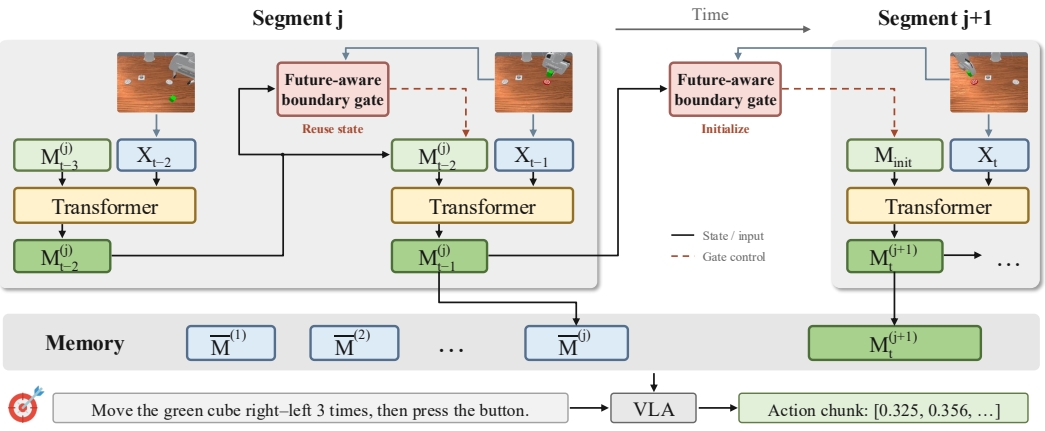


Figure 3: **Overview of the proposed SEGMEM architecture.** SEGMEM organizes an extended interaction history into compact segment memories. Before incorporating each observation, a learned gate either merges it into the active segment or archives the active segment and initializes a new one.

motivates an adaptive segmentation mechanism that organizes the interaction history according to its evolving temporal structure.

4 SEGMENT-WISE PROCEDURAL MEMORY WITH FUTURE-AWARE GATING

4.1 MEMORY PRETRAINING WITH RANDOM SEGMENTATION

Online boundary decisions produce segments with varying temporal durations. We therefore pretrain the recurrent compressor with randomly segmented histories, enabling it to consolidate observation sequences of different lengths into fixed-size segment states. Given an interaction history $\mathcal{H}_t$ containing N observations, we encode each observation as $\mathbf{X}_i = E_\psi(\mathbf{o}_i)$ and randomly partition the sequence into J contiguous segments. Let $j_i \in \{1, \dots, J\}$ denote the segment containing observation i. Each segment is represented by an m-token state $\mathbf{M}_i^{(j_i)} \in \mathbb{R}^{m \times d}$, yielding a total memory budget of $B = Jm$. Within a segment, the state is updated only from its previous state and the incoming visual tokens:

$$\mathbf{M}_i^{(j_i)} = U_\phi\left(\mathbf{M}_{i-1}^{(j_i)}, \mathbf{X}_i\right), \tag{5}$$

where U_ϕ denotes the segment-wise recurrent update. At the beginning of a new segment, the previous state is replaced by a shared learned initializer $\mathbf{M}_{\text{init}}$. Once a segment is completed, its final state is archived as $\overline{\mathbf{M}}^{(j)}$ and remains unchanged; archived states do not participate in subsequent recurrent updates.

After processing the history, the archived states and the current active state are concatenated and projected by P_ρ to form the policy-facing memory $\mathcal{M}_t$ in Eq. (1). The compressor and VLA policy are jointly optimized using the original action-generation loss. Resampling the segment boundaries during training exposes the compressor to diverse temporal durations, preparing it for the memory distribution produced by the learned gate.

4.2 FUTURE-AWARE BOUNDARY LEARNING

To determine when a new segment should be created before further recurrent consolidation induces representational interference, we supervise candidate boundary decisions according to their effects on subsequent action prediction. We evaluate these effects at both short- and long-horizon query steps to capture the immediate and persistent consequences of each decision.

For adaptive segmentation, we introduce a causal boundary gate G_ω parameterized by ω. Before $\mathbf{X}_t$ is written into memory, the gate receives the active state $\mathbf{M}_{t-1}^{(j_t)}$, the previous and current visual tokens $\mathbf{X}_{t-1}$ and $\mathbf{X}_t$, and the active segment age r_t. It predicts the probability of starting a

new segment as $p_t = G_\omega(\mathbf{M}_{t-1}^{(j_t)}, \mathbf{X}_{t-1}, \mathbf{X}_t, r_t)$. To supervise G_ω, we collect candidate boundary points from training rollouts and estimate the future action utility of merging and splitting through paired counterfactual continuations. For each candidate t, the *Merge* branch writes $\mathbf{X}_t$ into the active segment, whereas the *Split* branch archives the active state, initializes a new segment, and writes $\mathbf{X}_t$ into it. Both branches subsequently receive the same observations and follow identical merge-only updates, isolating the effect of the decision at t. We quantify the future action utility of the two branches by comparing their action-generation losses over future query steps $\mathcal{Q}_t$ spanning both short and long horizons. At each query $q \in \mathcal{Q}_t$, the branch-specific memory is provided to the memory-augmented VLA pretrained in Sec. 4.1, whose parameters remain frozen throughout boundary learning. We denote the resulting losses for the Merge and Split branches by $L_{t,q}^{\mathrm{M}}$ and $L_{t,q}^{\mathrm{S}}$, respectively, and compute them using the same action target and stochasticity. We then define the split advantage as

$$d_{t,q} = L_{t,q}^{\mathrm{M}} - L_{t,q}^{\mathrm{S}}, \qquad \Delta_t = \frac{1}{|\mathcal{Q}_t|} \sum_{q \in \mathcal{Q}_t} d_{t,q} - \lambda_{\mathrm{sp}}, \tag{6}$$

where λ_{sp} discourages unnecessary boundaries. Thus, $\Delta_t > 0$ indicates that splitting better supports subsequent action prediction.

The per-query advantages can vary in both magnitude and direction across query horizons, making their unnormalized mean unreliable when the queries disagree. We therefore normalize Δ_t by its uncertainty across queries. To prevent noisy comparisons from producing overly confident targets, we further clip and temperature-scale the normalized evidence. Finally, because comparisons with weak evidence provide unreliable supervision, we downweight them when optimizing the gate:

$$e_t = \frac{\Delta_t}{\mathrm{SE}\left(\{d_{t,q}\}_{q \in \mathcal{Q}_t}\right) + \epsilon}, \qquad y_t = \sigma\left(\frac{\mathrm{clip}(e_t, -\kappa, \kappa)}{\tau_g}\right), \tag{7}$$

$$\mathcal{L}_{\mathrm{gate}} = -\frac{1}{\sum_{t \in \mathcal{C}} w_t} \sum_{t \in \mathcal{C}} w_t \left[y_t \log p_t + (1 - y_t) \log(1 - p_t)\right], \tag{8}$$

where $\mathrm{SE}(\cdot)$ measures the standard error across query steps, ϵ ensures numerical stability, κ is the clipping threshold, τ_g is the target temperature, and $\sigma(\cdot)$ is the sigmoid function. The resulting y_t is a soft boundary target, with values above 0.5 favoring a split. Here, $\mathcal{C}$ denotes the set of candidate boundaries and w_t is a confidence weight that reduces the contribution of ambiguous comparisons.

Since boundary decisions affect how subsequent observations are consolidated into segment memories, we first collect supervision by replaying demonstrations with boundaries determined by the split advantage Δ_t. We then replay the demonstrations using the learned gate and recompute counterfactual targets to refine the gate. Further details are provided in Appendix A.2.

4.3 Policy Adaptation and Online Inference

Task-specific policy adaptation. Although random-segmentation pretraining enables the compressor to handle segments of varying lengths, the learned gate produces a task-dependent distribution of memory boundaries that differs from the random segmentations seen in Stage 1. We therefore adapt the policy to the memory organization induced by the learned gate. Specifically, we roll out the trained gate G_ω over the task-specific demonstrations and apply its Merge and Split decisions to construct the corresponding segment memories. For each action target $\mathbf{a}_t^{\mathrm{gt}}$, the archived and active segment states are assembled and projected by P_ρ to obtain the gate-induced policy-facing memory $\widehat{\mathcal{M}}_t$, where the hat distinguishes it from the randomly segmented memories used during pretraining. We then adapt the VLA policy π_θ and memory projection P_ρ using the original action-generation loss, yielding the optimized models π_{θ^*} and P_{ρ^*}, with $\widehat{\mathcal{M}}_t$ conditioning the policy as in Eq. (1). The visual encoder E_ψ, recurrent compressor U_ϕ, and boundary gate G_ω remain fixed during this stage.

Online inference. During inference, SegMem maintains an active segment, an archive of completed segments, the previous visual tokens, and the active segment age. At each memory-update step, the gate makes a boundary decision before the current visual tokens $\mathbf{X}_t$ are incorporated. A *Merge* updates the active state with $\mathbf{X}_t$, whereas a *Split* archives the active state, initializes a new segment, and writes $\mathbf{X}_t$ into it. Minimum and maximum segment-duration constraints prevent degenerate boundary decisions. The archived segment states and the current active state are then

Table 1: Full-horizon success rates (%) on RoboMME and LIBERO-Mem. *All-6 Avg.* includes six RoboMME tasks; *Short* and *Long* average over LIBERO-Mem tasks T1–T2 and T3–T8, respectively. Best and second-best results in each column are shown in **bold** and underlined, respectively.

Method	RoboMME				LIBERO-Mem			Overall
	PickXTimes	PickHighlight	SwingXTimes	All-6 Avg.	Short	Long	All-8 Avg.	
No Persistent History								
$\pi_{0.5}$	28	8	32	16.3	30.0	8.3	13.8	15.0
Qwen-OFT	0	0	6	3.0	12.5	1.7	4.4	3.7
Explicit Historical Evidence								
Mem0	0	4	0	2.7	25.0	1.7	7.5	5.1
Perceptual Sampling (context)	34	24	44	24.0	5.0	22.5	18.1	21.1
EventVLA	18	**36**	4	11.3	40.0	13.3	20.0	15.7
Latent Historical State								
MemoryVLA	14	4	6	9.0	10.0	5.0	6.3	7.6
Global Recurrent	18	12	16	12.0	**62.5**	20.8	31.3	21.6
SEGMEM (Ours)	**36**	26	**50**	**26.3**	57.5	**25.0**	**33.1**	**29.7**

assembled under the memory budget B and projected by $P_{\rho^\star}$ to form the policy-facing memory $\mathcal{M}_{t+1}$. The adapted policy $\pi_{\theta^\star}$ uses this memory for action prediction as formulated in Eq. (1).

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Datasets and Benchmarks. We evaluate SegMem on two simulation benchmarks for history-dependent manipulation. From RoboMME (Dai et al., 2026), we select six representative tasks in which the correct action depends on task progress or previously observed cues. We further evaluate on all eight tasks (T1–T8) of LIBERO-Mem (Chung et al., 2025), which assess object-level memory over both short and extended interaction horizons.

Baseline Methods. We compare SegMem with representative methods spanning three history-modeling settings: (i) *No Persistent History*, including $\pi_{0.5}$ (Intelligence et al., 2025b) and Qwen-OFT (Community, 2026); (ii) *Explicit Historical Evidence*, including Mem0 (Chen et al., 2026), EventVLA (Yang et al., 2026), and Perceptual Sampling (Context), abbreviated as **PSC** hereafter; and (iii) *Latent Historical State*, including MemoryVLA (Shi et al., 2026) and Global Recurrent. PSC and Global Recurrent follow their corresponding implementations in RoboMME (Dai et al., 2026). Please refer to Appendix A.3 for more details.

Implementation Details. We instantiate SegMem and backbone-compatible baselines on $\pi_{0.5}$, while methods tied to the Qwen-OFT action interface retain that backbone. Each current observation contains one head-view and one wrist-view image, whereas the memory organizer operates on the head-view stream. The memory bank contains at most 16 segments with 32 latent tokens per segment, yielding a total budget of 512 tokens. Memory is updated every 16 control steps on RoboMME and every 12 control steps on LIBERO-Mem. For future-aware boundary learning, we evaluate action losses at offsets of $0, 16, 32, 64$ control steps on RoboMME and $0, 12, 24, 48$ control steps on LIBERO-Mem. We constrain each segment to span between 2 and 16 memory updates and calibrate the splitting threshold on held-out validation data. Due to computational constraints, we do not train a single policy jointly across all tasks; instead, each model is trained and evaluated independently on one task. Additional training and evaluation details are provided in the Appendix A.2.

5.2 MAIN RESULTS

Simulation Benchmarks. As shown in Table 1, SEGMEM achieves the best overall performance, outperforming the strongest baselines by 8.1–8.6 percentage points. Notably, several memory-augmented baselines perform worse than the history-free $\pi_{0.5}$ policy, suggesting that merely incorporating historical information does not guarantee better decisions and that effectively organizing history under a bounded context is equally important. At the task level, different forms of history dependence favor different memory representations: SEGMEM performs particularly well

Table 2: Training-strategy and intra-segment-aggregation ablations. PO-AD denotes the **po**licy **ad**aptation phase.

(a) Training strategy

ID	Boundary	Po-Ad	Task Success Rate (%) PickXTimes	PickHighlight	SwingXTimes	Avg.
A	Random	✗	30	10	18	19.3
B	Random	✓	28	10	30	22.7
C	Learned	✗	**36**	12	20	22.7
D (Full)	**Learned**	✓	**36**	**26**	**50**	**37.3**

(b) Intra-segment aggregation

Aggregation	SwingXTimes	PickHighlight	BinFill	Avg.
Temporal mean	40	6	8	18.0
Last frame	**54**	4	14	24.0
Recurrent (ours)	50	**14**	**16**	**26.7**

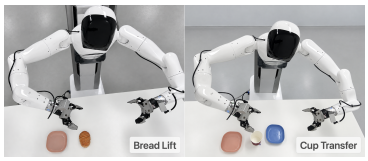


Figure 4: Illustration of the two real-world tasks.

Table 3: Results of real-world experiments.

Method	Bread Lift	Cup Transfer	Overall
$\pi_{0.5}$	35%	35%	35.0%
PSC	40%	95%	67.5%
OURS	60%	100%	80%

on progress-dependent and repeated-process tasks, such as PickXTimes and SwingXTimes. By contrast, EventVLA performs best on PickHighlight, where decisions depend primarily on a transient visual cue, highlighting the advantage of retaining explicit observations when a single visual detail is critical. Finally, the horizon-wise results on LIBERO-Mem show that Global Recurrent remains competitive on shorter tasks, whereas SEGMEM performs best on longer tasks, suggesting that segment-wise organization becomes increasingly beneficial as longer interaction histories must be retained within a fixed memory budget. Complete per-task results are provided in Appendix B.1.

Real-world Experiments. We further evaluate SEGMEM on an AgiBot G1 robot against the history-free $\pi_{0.5}$ policy and PSC, both using the same $\pi_{0.5}$ backbone. We consider two history-dependent tasks, as presented in Figure 4. In *Bread Lift*, the robot repeatedly grasps a piece of bread, places it on a pink plate, releases it, raises the gripper, and grasps it again for $n \in 1, \ldots, 5$ cycles. In *Cup Transfer*, the robot grasps a cup and alternates it between the blue and pink plates for $n \in 1, 2, 3$ cycles. As shown in Table 3, the real-world results are consistent with our simulation findings: SEGMEM achieves the best performance on both tasks and outperforms both baselines overall. Notably, PSC rarely completes Bread Lift instances requiring three or more cycles, as sampled frames provide limited cues for distinguishing repeated release–lift–regrasp phases. Its strong performance on Cup Transfer suggests that explicit observations remain effective when interaction stages are visually distinct and the required horizon is shorter. In contrast, SEGMEM more reliably maintains progress across repeated interaction phases, demonstrating the benefit of organizing extended histories for long-horizon execution. Additional details are provided in Appendix A.4.

5.3 ABLATION STUDY

We ablate the learned boundary gate and task-specific policy adaptation to examine their respective contributions. As shown in Table 2, each component improves performance, while combining them yields substantially greater gains. This suggests that the two components are complementary: learned boundaries provide more informative temporal organization, while policy adaptation helps the action policy interpret the resulting memory distribution. We further compare recurrent aggregation with temporal mean pooling and last-frame retention under the same fixed boundaries. Recurrent aggregation achieves the best average performance, while mean pooling can obscure temporally localized information and last-frame retention discards earlier cues. Together, these findings highlight the benefits of learned boundaries and temporal integration within segments.

5.4 IN-DEPTH ANALYSES OF BOUNDARY GATE

Is the Boundary Gate Effective? We first compare our learned gate with several alternative segmentation strategies under the same memory budget. *Fixed* follows the fixed-length segmentation protocol introduced in our preliminary study, while *Random* samples boundary locations with the segment count matched to our gate. *Visual Change* places boundaries according to changes in vi-

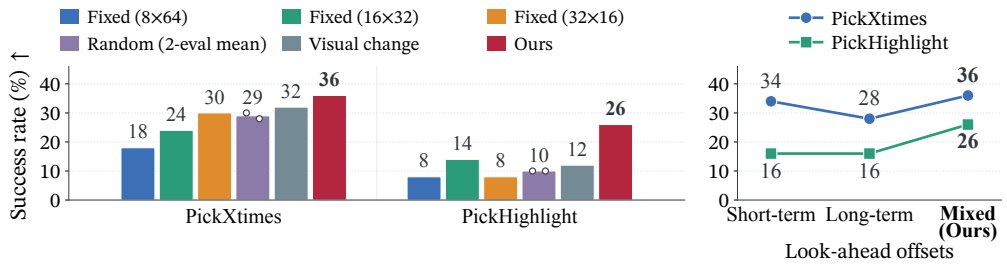


Figure 5: Boundary-gate analysis. *Left:* Comparison of segmentation strategies under the same memory budget. *Right:* Effect of the supervision horizon on the learned gate.

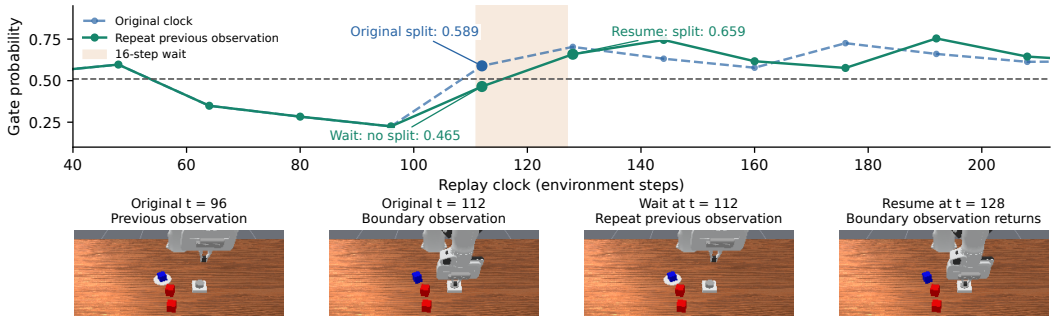


Figure 6: Boundary-gate robustness test under temporal shift. Inserting a waiting period delays the predicted boundary accordingly: the gate remains inactive during the repeated observation and triggers when the highlight disappears after execution resumes.

sual features, with its threshold calibrated to produce the same average number of segments as our method. As shown in Figure 5, our learned gate consistently outperforms all alternative strategies. The gains over segment-count-matched Random and Visual Change baselines underscore the importance of boundary placement. These results suggest that future decision utility provides a more effective basis for segmentation than perceptual changes alone. We further examine how the look-ahead horizon used for gate supervision affects performance by considering short-term, long-term, and mixed-horizon supervision. As shown in Figure 5, our mixed-horizon supervision outperforms both single-horizon variants across the two tasks. These results suggest that effective boundary learning benefits from considering both immediate action transitions and their delayed consequences.

Is the Boundary Gate Robust under Distribution Shift? To examine whether the boundary gate remains reliable when test-time dynamics deviate from training, we introduce a controlled temporal shift into a PickHighlight trajectory. Specifically, we insert a waiting period immediately before the original boundary by repeatedly presenting the preceding observation and then resume the trajectory. As shown in Figure 6, the gate does not split at the original clock time; instead, its prediction shifts with the inserted delay and triggers only after execution resumes. Importantly, the delayed split remains aligned with the disappearance of the highlight, which marks the task-relevant transition. This result suggests that the gate responds to meaningful changes in the interaction rather than memorizing a fixed temporal schedule, demonstrating robustness to temporal distribution shifts.

6 Conclusion

We studied how to organize extended interaction histories for long-horizon robotic manipulation under a fixed context budget. We introduced SegMem, which recurrently compresses observations within variable-length segments and learns their boundaries according to future action-prediction utility. SegMem demonstrates consistent improvements across RoboMME, LIBERO-Mem, and real-world tasks, particularly on tasks requiring progress tracking over extended horizons. Our results highlight the importance of temporal organization in memory-augmented robot policies and provide a promising direction for scalable history-dependent manipulation.

AI USE STATEMENT

We used ChatGPT and Codex to assist with experimental implementation and debugging, data processing, and manuscript refinement. We affirm that all core content and findings in this paper are the original work of the authors. The authors are responsible for the reported evidence, interpretations, and final manuscript.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, Thomas Unterthiner, Daniel Keysers, Skanda Koppula, Fangyu Liu, Adam Grycner, Alexey Gritsenko, Neil Houlsby, Manoj Kumar, Keran Rong, Julian Eisenschlos, Rishabh Kabra, Matthias Bauer, Matko Bošnjak, Xi Chen, Matthias Minderer, Paul Voigtlaender, Ioana Bica, Ivana Balazevic, Joan Puigcerver, Pinelopi Papalampidi, Olivier Henaff, Xi Xiong, Radu Soricut, Jeremiah Harmsen, and Xiaohua Zhai. Paligemma: A versatile 3b vlm for transfer, 2024. URL https://arxiv.org/abs/2407.07726.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023. URL https://arxiv.org/abs/2212.06817.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in neural information processing systems*, 35:11079–11091, 2022.

Josef Chen. Aura: Action-gated memory for robot policies at constant vram, 2026. URL https://arxiv.org/abs/2606.02775.

Peng Chen, Yingying Zhang, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting, 2024. URL https://arxiv.org/abs/2402.05956.

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

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. URL https://arxiv.org/abs/2606.12497.

Nhat Chung, Taisei Hanyu, Toan Nguyen, Huy Le, Frederick Bumgarner, Duy Minh Ho Nguyen, Khoa Vo, Kashu Yamazaki, Chase Rainwater, Tung Kieu, Anh Nguyen, and Ngan Le. Rethinking progression of memory state in robotic manipulation: An object-centric perspective, 2025. URL https://arxiv.org/abs/2511.11478.

StarVLA Community. Starvla: A lego-like codebase for vision-language-action model developing, 2026. URL https://arxiv.org/abs/2604.05014.

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

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025a. URL https://arxiv.org/abs/2511.14759.

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

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 21502–21521. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/jelassi24a.html.

Yuxin Jiang, Chang Yu, Yunuo Chen, Xiang Feng, Yin Yang, Nishank Gite, and Chenfanfu Jiang. Memoryvam: Integrating memory into video action model for robot manipulation, 2026. URL https://arxiv.org/abs/2606.20679.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Hang Li, Fengyi Shen, Dong Chen, Liudi Yang, Xudong Wang, Jinkui Shi, Zhenshan Bing, Ziyuan Liu, and Alois Knoll. Remem-vla: Empowering vision-language-action model with memory via dual-level recurrent queries, 2026. URL https://arxiv.org/abs/2603.12942.

Zhe Liu, Jinghua Hou, Yuxiang Lu, Zhenya Yang, Xianzhe Fan, Junwei Luo, Junyi Li, Ruihua Han, Zhi Hou, and Hengshuang Zhao. Streampi: Streaming multimodal temporal modeling for vision-language-action models, 2026. URL https://arxiv.org/abs/2608.26067.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35492–35506. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/merrill24a.html.

Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srinivasan Iyer. Byte latent transformer: Patches scale better than tokens, 2024. URL https://arxiv.org/abs/2412.09871.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation, 2026. URL https://arxiv.org/abs/2508.19236.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Memer: Scaling up memory for robot control via experience retrieval, 2025. URL https://arxiv.org/abs/2510.20328.

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

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

Yitian Zhang, Liheng Ma, Soumyasundar Pal, Yingxue Zhang, and Mark Coates. Multi-resolution time-series transformer for long-term forecasting. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li (eds.), *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 4222–4230. PMLR, 02–04 May 2024. URL https://proceedings.mlr.press/v238/zhang24l.html.

Shoujing Zhu, Zhenyang Liu, Fungmiu Wang, Jiafeng Wang, Bo Yue, Guiliang Liu, Simo Wu, Xiangyang Xue, and Taiping Zeng. Weavela: Event driven cross-subtask latent memory weaving for repetitive robot manipulation, 2026. URL https://arxiv.org/abs/2606.17463.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A Supplementary Experimental and Implementation Details

A.1 Pilot Study Setup and Results

Experimental Setup. We conduct the pilot study on seven RoboMME tasks, using 100 demonstrations per task. For each task and memory configuration, we fine-tune a separate $\pi_{0.5}$-based policy for 10,000 updates with AdamW, a batch size of 32, and a learning rate of 5×10^{-5}. All configurations use a memory budget of $B = 512$. L counts sampled memory updates rather than environment steps; $L = 8, 4, 2$ correspond to allocations of 8×64, 16×32, and 32×16, respectively. We evaluate the final checkpoints using 50 closed-loop episodes per task.

Action-Error Diagnostics. We replay the same demonstration trajectories for all configurations, evaluating predictions every 16 environment steps. These selected trajectories come from the held-out demonstrations and serve as illustrative diagnostics. For Eq. 4, we evaluate the first $H_a = 16$ actions of each 20-step prediction over $d_a = 8$ action dimensions, using the same action normalization across configurations. At each query, we generate four predictions using independent noise draws shared across methods. Thin curves show the mean NMAE across these predictions; thick curves apply a centered three-point moving average. Shaded regions show the unsmoothed mean $\pm$ one standard deviation, clipped at zero, and therefore describe action-sampling variability. Colored ticks indicate the nominal fixed-segment intervals.

Task-Level Results. Figure 7 extends the main-text comparison to seven tasks. With $L = 2$, success rates increase from 18% to 30% on PickXTimes and from 16% to 58% on SwingXTimes. However, the global RMT retains the highest success rates on ButtonUnmask and BinFill, while $L = 4$ performs best on StopCube and PickHighlight. These results show that the benefit of segmentation is task-dependent: finer temporal partitioning does not uniformly improve performance under a fixed memory budget.

Trajectory-Level Results. Figures 8–10 illustrate how the relative performance of memory configurations varies within individual trajectories. On PickHighlight episode 866, $L = 2$ substantially reduces the elevated error in the middle of the trajectory; on episode 874, both $L = 8$ and $L = 2$ maintain lower error over much of this interval. On PickXTimes episode 584, however, $L = 8$ yields lower error near the end, while finer segmentation exhibits localized error spikes. These examples complement the task-level results by showing that the preferred fixed granularity can also vary across trajectory phases.

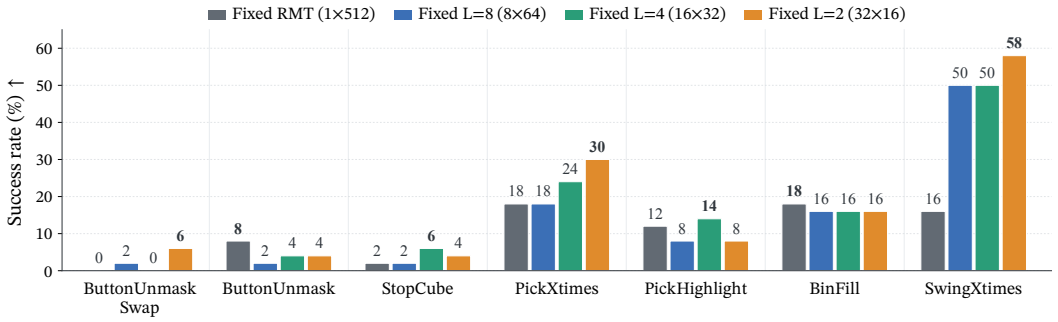


Figure 7: **Task-level evaluation of fixed segmentation granularity on RoboMME.** Success rates (%) across seven tasks for a global RMT and fixed-length segmented variants ($L \in \{2, 4, 8\}$) under a matched budget of 512 memory tokens. Each allocation denotes the number of segments $\times$ tokens per segment.

A.2 Training and Evaluation Protocol

Memory and Boundary Configurations. Table 4 summarizes the principal hyperparameters, using the notation of the main text. Each segment contains $m = 32$ memory tokens of dimension $d = 512$. During random-segmentation pretraining, we allocate $J = 16$ segments and uniformly

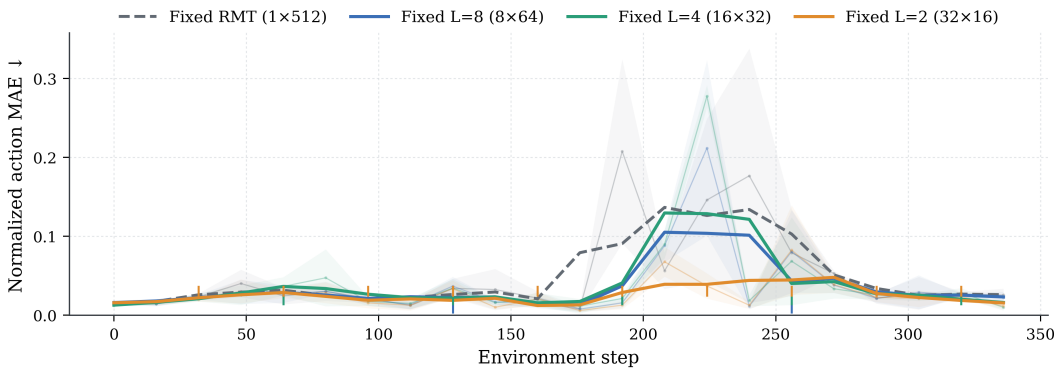


Figure 8: Normalized action MAE along a PICKHIGHLIGHT demonstration trajectory (episode 866) for a global RMT and fixed-length segmented variants.

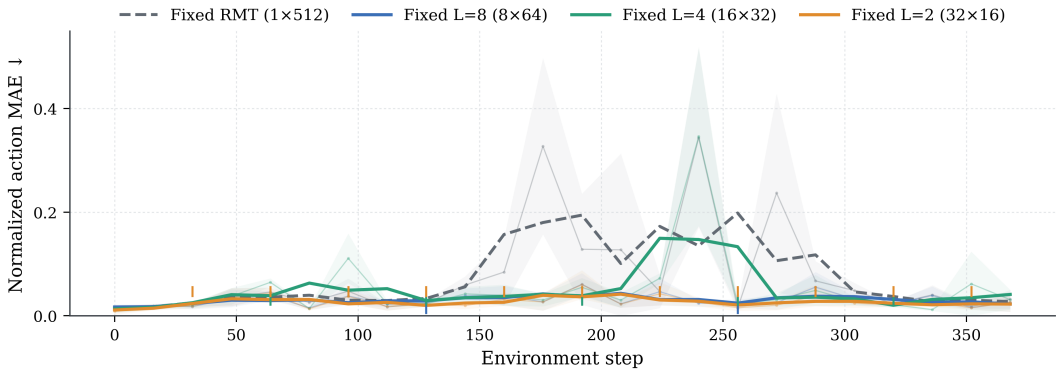


Figure 9: Normalized action MAE along a PICKHIGHLIGHT demonstration trajectory (episode 874) for a global RMT and fixed-length segmented variants.

sample internal boundaries from the valid history positions, using fewer nonempty segments when the history is short. For the $B = 512$ configuration, the policy reads the most recent $B/m = 16$ segment states, including the active segment. We update memory every δ environment steps, with $\delta = 16$ for RoboMME and $\delta = 12$ for LIBERO-Mem. Segment ages and query offsets are measured in memory updates. The query set $\mathcal{Q}_t$ uses offsets $\{0, 1, 2, 4\}$ relative to the candidate boundary, where offset zero denotes the immediate action query.

Training Schedule. We initialize the policy from the pretrained $\pi_{0.5}$ checkpoint and train a separate SEGMEM model for each task. Table 5 summarizes the three-stage optimization schedule. Stage 1 jointly optimizes the recurrent compressor, memory projection, and policy for 15,000 updates using randomly segmented histories. Stage 2 trains only the boundary gate while keeping the Stage 1 model fixed. We perform two rounds of gate learning, first using teacher-induced trajectories and then recollecting supervision along trajectories induced by the learned gate. The second round is initialized from the first-round gate and samples 60% of its training examples from the recollected data and 40% from the first round. In Stage 3, we freeze the visual encoder, recurrent compressor, and gate, and adapt the policy and memory projection for 10,000 updates using gate-induced memories. Both policy-training stages use the original flow-matching action loss.

Gate Architecture. Figure 11 illustrates the boundary gate G_ω. The gate combines three paths: attention pooling of the active memory $\mathbf{M}_{t-1}^{(j_t)}$, memory-conditioned encoding of the frame pair $(\mathbf{X}_{t-1}, \mathbf{X}_t)$, and an encoding of the segment age r_t. In the frame-pair path, visual tokens attend to the active memory through cross-attention, followed by a feed-forward network and attention pooling. The resulting summaries, their visual change and memory–current interaction features, and the age encoding are fused and passed to an MLP with a sigmoid output to predict p_t. All inputs

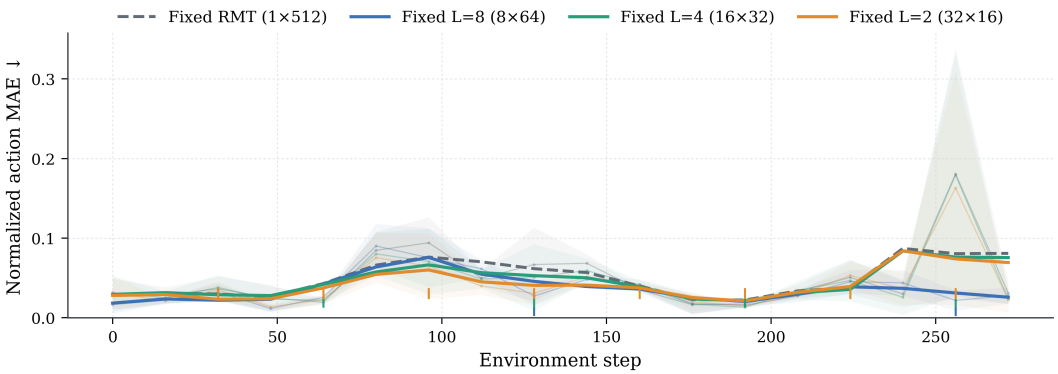


Figure 10: Normalized action MAE along a PICKXTIMES demonstration trajectory (episode 584) for a global RMT and fixed-length segmented variants.

Table 4: Principal hyperparameters of SEGMEM. Values are shared across benchmarks unless specified otherwise.

Symbol	Description	RoboMME	LIBERO-Mem
B	Policy-facing memory-token budget	512	512
m	Memory tokens per segment	32	32
J	Segment allocation during pretraining	16	16
d	Recurrent memory dimension	512	512
δ	Memory-update interval (environment steps)	16	12
$r_{\min}$	Minimum segment age before a split is allowed	2	2
$r_{\max}$	Segment age at which a split is forced	16	16
$\{q-t : q \in \mathcal{Q}_t\}$	Action-query offsets (memory updates)	$\{0,1,2,4\}$	$\{0,1,2,4\}$
λ_{sp}	Split penalty in Eq. equation 6	0	0
κ	Evidence clipping threshold	4	4
τ_g	Soft-target temperature	1	1
ϵ	Stabilizer for evidence normalization	10^{-8}	10^{-8}

are available before the current observation is written into memory, preserving causal boundary decisions.

Gate Supervision and Calibration. Boundary decisions determine how observations are grouped and consolidated into segment memories. Since the gate takes the active segment memory as input, earlier boundary decisions can change its later inputs even when the observation sequence is unchanged. We train the gate through two rounds of demonstration replay: the first uses boundaries determined by counterfactual comparisons, and the second recollects supervision under boundaries predicted by the learned gate. The visual encoder, recurrent compressor, memory projection, and VLA policy obtained from memory pretraining remain frozen throughout both rounds.

Round 1: collection using counterfactual comparisons. We replay each demonstration and evaluate paired Merge and Split continuations at eligible candidate boundaries. Both branches receive the same subsequent observations and follow merge-only updates. At each query in $\mathcal{Q}_t$, their action losses are computed using the same target, flow noise, and flow time, with one flow-matching sample per query. The resulting split advantage Δ_t determines the boundary decision used to advance the replay: we select Split when $\Delta_t > 0$ and Merge otherwise, subject to the segment-age constraints. At each candidate, we store the gate inputs and paired losses, construct the soft target y_t according to Eq. equation 7, and use the collected examples to train the first-round gate with Eq. equation 8.

Round 2: gate-guided recollection and refinement. We replay the same demonstrations again, now determining boundaries using the first-round gate's predictions and calibrated decision threshold. At each eligible candidate, we recompute the paired counterfactual losses using the active segment memory produced by the preceding gate decisions, and obtain a new soft target y_t. The learned

Table 5: Optimization settings. Batch sizes are global; Stage 2 batches contain candidate boundaries. Gate-training epochs are specified per round.

Setting	Stage 1	Stage 2	Stage 3
Training duration	15,000 updates	2×100 epochs	10,000 updates
Batch size	32	32	32
Optimizer	AdamW	AdamW	AdamW
Peak learning rate	5×10^{-5}	3×10^{-4}	10^{-5}
Final learning rate	5×10^{-5}	3×10^{-5}	10^{-6}
Warmup	100 updates	5% of updates	100 updates
Schedule after warmup	Constant	Cosine decay	Cosine decay

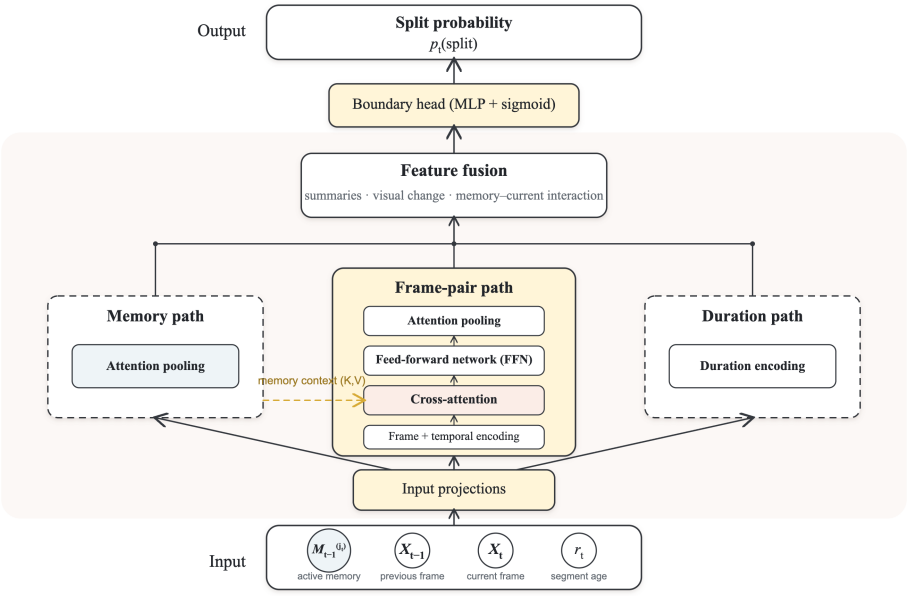


Figure 11: **Architecture of the boundary gate G_ω.** The memory, frame-pair, and duration paths provide complementary features for predicting the split probability p_t before the current memory update.

gate thus determines the segmentation followed during replay, while the counterfactual comparisons provide supervision for refining its predictions. We initialize the second-round gate from the first-round parameters and train it with a mixture of 60% recollected examples and 40% first-round examples. Each round uses 100 training epochs.

In both rounds, we set $\lambda_{\text{sp}} = 0$ and compute the confidence weights as $w_t \propto 0.5 + 0.5 \min(|e_t|/2, 1)$, where e_t is the unclipped evidence in Equation 7; the common normalization factor cancels in Equation 8. After each round, we calibrate the split-probability threshold on held-out validation trajectories by minimizing normalized decision regret. The first-round threshold is used during recollection, whereas the final threshold governs boundary decisions when constructing gate-induced memories for task-specific policy adaptation and during online inference. The segment-age constraints are enforced during both collection rounds and deployment.

Evaluation. We follow the task definitions and evaluation settings of RoboMME (Dai et al., 2026) and LIBERO-Mem (Chung et al., 2025). We report full-horizon task success rate (%), using 50 evaluation episodes for each RoboMME task and 20 for each LIBERO-Mem task. The policy predicts 20-step action chunks on RoboMME and 10-step action chunks on LIBERO-Mem. All model parameters remain fixed during evaluation, and the recurrent memory and segment archive are reset

Table 6: **Simulation tasks used in our evaluation.** Descriptions specify the required behavior and the historical information relevant to task completion.

Task	Description
RoboMME	
Pick×Times	Repeatedly lift and replace a cube of the requested color, tracking completed repetitions to stop at the specified count.
PickHighlight	Pick the cubes indicated by earlier brief highlights, retaining their identities after the visual cues disappear.
BinFill	Deposit the requested number of cubes of a specified color into a bin as cubes appear, tracking cumulative placements.
Swing×Times	Move a cube back and forth between two targets, keeping count of the requested motion cycles.
RouteStick	Reproduce a demonstrated route around vertical obstacles, recalling the target order and circling directions.
PatternLock	Trace a demonstrated pattern across a target grid, remembering the ordered trajectory and execution progress.
LIBERO-Mem	
T1: Bowl $\times 1$	Lift a bowl and return it to its plate once, distinguishing completion from the initial state.
T2: Bottle $\times 1$	Lift a bottle and return it to its plate once, remembering whether the operation has been completed.
T3: Bowl $\times 3$	Complete three bowl lift-and-return cycles, tracking the count despite repeated scene configurations.
T4: Bottle $\times 3$	Complete three bottle lift-and-return cycles and stop after the third return.
T5: Bowl $\times 5$	Complete five bowl lift-and-return cycles, maintaining progress over a longer sequence.
T6: Bowl $\times 7$	Complete seven bowl lift-and-return cycles, retaining the count throughout extended repetition.
T7: Two-bowl swap	Exchange two bowls between their plates using a third, empty plate as temporary storage; track identity and placement history.
T8: Three-bowl swap	Cyclically rearrange three bowls using a fourth, empty plate as temporary storage; preserve the required transfer order.

at the beginning of each episode. Benchmark-level results are the unweighted mean of the per-task success rates.

A.3 Benchmarks and Baseline Configurations

A.3.1 Benchmarks and Task Selection

We evaluate on six tasks from RoboMME (Dai et al., 2026) and eight tasks (T1–T8) from LIBERO-Mem (Chung et al., 2025). Table 6 summarizes the tasks and their memory requirements.

We select the six RoboMME tasks for their substantial demands on procedural memory. At critical decision points, the current observation alone is insufficient to determine the appropriate action: the policy must retain previously observed cues, track completed operations, or recall a demonstrated motion sequence. These tasks primarily involve relatively simple manipulation primitives, such as picking, placing, and following a prescribed trajectory. The selection is intended to reduce confounding from fine-grained motor control, allowing performance to more directly reflect the ability to retain and use procedural information.

LIBERO-Mem provides a complementary evaluation of object-level memory. T1–T2 require a single lift-and-return operation, whereas T3–T6 require tracking repeated operations and terminating after the requested count. T7–T8 require maintaining object identities and placement history while rearranging bowls using an empty plate. We group T1–T2 as short-horizon tasks and T3–T8 as long-horizon tasks. Together, these tasks test whether policies can distinguish visually similar states associated with different stages of execution.

A.3.2 Baseline Methods and Configurations

We organize the baselines according to how they make historical information available for action prediction, following the three categories in the main text.

No Persistent History. $\pi_{0.5}$ (Intelligence et al., 2025b) predicts actions from the current observation and task instruction, without an additional memory mechanism that carries information across decision steps. **Qwen-OFT** (Community, 2026) combines a Qwen vision–language backbone with an OFT-style action head for parallel continuous action prediction. We evaluate it without persistent historical context.

Explicit Historical Evidence. **Mem0** (Chen et al., 2026) combines anchor and sliding-window memories for action execution with a keyframe-based memory for subtask planning. A subtask-completion classifier coordinates planning and execution. **EventVLA** (Yang et al., 2026) retains the initial observation and recent frames as visual anchors and augments them with event-dependent keyframes. Its Keyframe Evidence Memory module predicts which upcoming observations should be stored for later decisions. **Perceptual Sampling (Context)**, abbreviated as **PSC**, uniformly samples historical frames and concatenates their visual tokens with the current policy inputs, following the FrameSamp memory representation and Memory-as-Context integration in RoboMME (Dai et al., 2026).

Latent Historical State. **MemoryVLA** (Shi et al., 2026) maintains a perceptual–cognitive memory bank containing visual details and semantic representations. Current features retrieve and fuse relevant memories, while similar adjacent entries are merged to control memory capacity. **Global Recurrent** follows the Recurrent Memory Transformer with Memory-as-Context integration in RoboMME (Dai et al., 2026). It summarizes incoming observations by repeatedly updating a shared set of latent memory slots throughout the episode.

Backbone and Adaptation Details. We use $\pi_{0.5}$ as the shared policy backbone for SegMem, Mem0, PSC, and Global Recurrent. Qwen-OFT uses its Qwen-based architecture, and our EventVLA and MemoryVLA reproductions are adapted to Qwen-OFT. These two methods are therefore reported as adapted variants, with their backbone indicated in the result tables. PSC and Global Recurrent follow the corresponding RoboMME implementations described above. This setup provides comparisons across memory mechanisms within the $\pi_{0.5}$ family and complementary comparisons against the Qwen-OFT-based variants.

A.4 Additional Real-World Experimental Details

Hardware and Data Collection. We conduct the real-world experiments on an AgiBot G1 robot. Demonstrations are collected through teleoperation using a PICO 4 VR headset and handheld controllers. We collect 50 demonstrations for each task, yielding 100 demonstrations in total. For *Bread Lift*, the dataset contains 10 demonstrations for each requested cycle count $n \in \{1, \ldots, 5\}$. For *Cup Transfer*, it contains 16, 17, and 17 demonstrations for $n = 1, 2, 3$, respectively.

Task Definitions and Success Criteria. Both tasks require the robot to execute the number of cycles specified in the instruction. In *Bread Lift*, the robot repeatedly grasps and lifts the bread, places it on the pink plate, releases it, and raises the gripper before the next grasp. A lift is counted as valid only if the grasped bread reaches a height of at least 20 cm above the tabletop. In *Cup Transfer*, the robot grasps the cup and moves it alternately between the blue and pink plates. One cycle corresponds to a complete round trip from the blue plate to the pink plate and back to the blue plate. A transfer to either plate is counted as valid only when the cup reaches a position above that plate. A trial is successful only if the robot completes exactly the instructed number of valid cycles. Each trial has a time limit of 5 minutes; trials that do not satisfy the success criteria within this limit are counted as failures.

Training and Evaluation Protocol. We compare SegMem with the history-free $\pi_{0.5}$ policy and PSC, using the same $\pi_{0.5}$ backbone and demonstration datasets. All methods are trained for 30,000 optimization steps with a batch size of 128. We evaluate each method over 20 trials per task. For

Table 7: **Real-world results by requested cycle count.** Demonstration counts are shared across methods, and evaluation counts are per method. Results are success rates (%), with successful/total trials in parentheses. Task-level totals pool trials across all cycle counts. Best results in each row are bold, including ties.

Cycles n	Demonstrations	Eval. trials	$\pi_{0.5}$	PSC	SEGMEM
Bread Lift					
1	10	4	**100.0 (4/4)**	**100.0 (4/4)**	**100.0 (4/4)**
2	10	4	50.0 (2/4)	**75.0 (3/4)**	**75.0 (3/4)**
3	10	4	25.0 (1/4)	25.0 (1/4)	**75.0 (3/4)**
4	10	4	0.0 (0/4)	0.0 (0/4)	**50.0 (2/4)**
5	10	4	**0.0 (0/4)**	**0.0 (0/4)**	**0.0 (0/4)**
Total	50	20	35.0 (7/20)	40.0 (8/20)	**60.0 (12/20)**
Cup Transfer					
1	16	6	**100.0 (6/6)**	83.3 (5/6)	**100.0 (6/6)**
2	17	7	14.3 (1/7)	**100.0 (7/7)**	**100.0 (7/7)**
3	17	7	0.0 (0/7)	**100.0 (7/7)**	**100.0 (7/7)**
Total	50	20	35.0 (7/20)	95.0 (19/20)	**100.0 (20/20)**

Bread Lift, we conduct four trials for each $n \in \{1, \ldots, 5\}$. For *Cup Transfer*, we conduct six, seven, and seven trials for $n = 1, 2, 3$, respectively. Task-level success rates are computed over all 20 trials, accounting for the unequal numbers of trials across cycle counts in *Cup Transfer*.

Results by Requested Cycle Count. Table 7 shows that SEGMEM better sustains successful execution as the required number of cycles increases. On *Bread Lift*, it achieves 75% and 50% success at $n = 3$ and $n = 4$, respectively, compared with 25% and 0% for both baselines. This advantage at larger repetition counts suggests that SEGMEM can maintain task progress over longer sequences of repeated interactions. On *Cup Transfer*, it achieves 100% success across all tested cycle counts, matching PSC at $n = 2$ and $n = 3$ and substantially outperforming the history-free policy at these settings. Together, these results support the benefit of organizing extended histories into segment-level memories for longer-horizon real-world execution.

B ADDITIONAL RESULTS AND ANALYSES

B.1 COMPLETE PER-TASK RESULTS

Tables 8 and 9 report the complete per-task full-horizon success rates on RoboMME and LIBERO-Mem, respectively. These results complement the aggregate comparisons in the main text by showing how the gains are distributed across individual tasks.

On RoboMME, SEGMEM achieves the best or joint-best performance on four of the six tasks and ranks second on the remaining two. Beyond the gains on PickXTimes and SwingXTimes discussed in the main text, it also achieves the highest success rate on BinFill (24%, compared with 18% for the strongest baselines) and matches the best result on RouteStick. Moreover, SEGMEM outperforms Global Recurrent on every RoboMME task, indicating that the advantage of segment-wise memory organization extends across the evaluated tasks.

On LIBERO-Mem, SEGMEM ranks among the top two methods on seven of the eight tasks. Its advantage is particularly evident on T5 and T6, where it achieves success rates of 10% and 20%, respectively, while all baselines remain at or below 5%. It also attains joint-best performance on T8. Although Global Recurrent leads on T2 and T3, SEGMEM outperforms it on five other tasks and matches it on T7, achieving the highest overall average. Together, these results suggest that the aggregate gains reflect broadly competitive performance across tasks, with additional benefits on several challenging long-horizon tasks.

Table 8: Complete per-task RoboMME success rates (%). Ties are marked equally.

Method	PickXTimes	PickHighlight	BinFill	SwingXTimes	RouteStick	PatternLock	Avg.
No Persistent History							
$\pi_{0.5}$	28	8	14	32	6	10	16.3
Qwen-OFT	0	0	0	6	2	10	3.0
Explicit Historical Evidence							
Mem0	0	4	2	0	2	8	2.7
Perceptual Sampling (context)	34	24	18	44	**10**	**14**	24.0
EventVLA	18	**36**	2	4	0	8	11.3
Latent Historical State							
MemoryVLA	14	4	10	6	6	**14**	9.0
Global Recurrent	18	12	18	16	2	6	12.0
SEGMEM (Ours)	**36**	26	**24**	**50**	**10**	12	**26.3**

Table 9: Complete per-task LIBERO-Mem full-horizon success rates (%). Each task contains 20 evaluation episodes, so a five-point difference corresponds to one episode. Ties are marked equally.

Method	T1	T2	T3	T4	T5	T6	T7	T8	Avg.
No Persistent History									
$\pi_{0.5}$	35	25	15	15	5	0	5	10	13.8
Qwen-OFT	25	0	5	0	0	5	0	0	4.4
Explicit Historical Evidence									
Mem0	40	10	0	5	0	5	0	0	7.5
Perceptual Sampling (context)	5	5	40	**55**	0	0	15	**25**	18.1
EventVLA†	**65**	15	40	15	0	5	**20**	0	20.0
Latent Historical State									
MemoryVLA†	15	5	10	5	5	5	5	0	6.3
Global Recurrent	35	**90**	**70**	30	0	0	10	15	31.3
SEGMEM (Ours)	60	55	45	40	**10**	**20**	10	**25**	**33.1**

B.2 Failure Analysis and Case Studies

We examine two complementary demands on memory: retaining a transient target cue and tracking progress through repeated manipulation. Figures 12 and 13 show matched scenes selected from archived diagnostic runs in which SEGMEM succeeds and $\pi_{0.5}$, PSC, and Global Recurrent fail.

Acting on a cue after it disappears. In PICKHIGHLIGHT, a brief white highlight identifies the green cube before the subsequent pickup decision. SEGMEM approaches and lifts the indicated cube after the cue has disappeared (Figure 12). In the same scene, the three baselines instead manipulate blue distractors and eventually time out. The difference concerns which object to manipulate, despite all policies being able to execute a pickup. This example illustrates the successful use of earlier task-relevant visual evidence when the current image no longer identifies the target.

Completing the required count without continuing past it. LIBERO-Mem T5 requires five bowl lift-and-return cycles, repeatedly revisiting a similar scene configuration. In Figure 13, SEGMEM completes all five cycles and maintains the final goal. The current-observation policy records no completed cycle, Global Recurrent completes four, and PSC completes five but then lifts the bowl again, triggering overshoot. These trajectories separate insufficient progress from unnecessary manipulation after completion.

Error distribution on extended repetition. Table 10 includes every unseen episode of T5 and T6 (five and seven cycles). SEGMEM succeeds in 9/40 episodes, compared with 3/40 for $\pi_{0.5}$, 0/40 for Global Recurrent, and 1/40 for PSC. PSC records all required cycles in all 40 episodes but subsequently overshoots in 39; Global Recurrent remains incomplete in all 40. SEGMEM reduces both bottlenecks relative to these respective baselines, although its 22 incomplete episodes show that sustained progress remains a substantial source of failure.

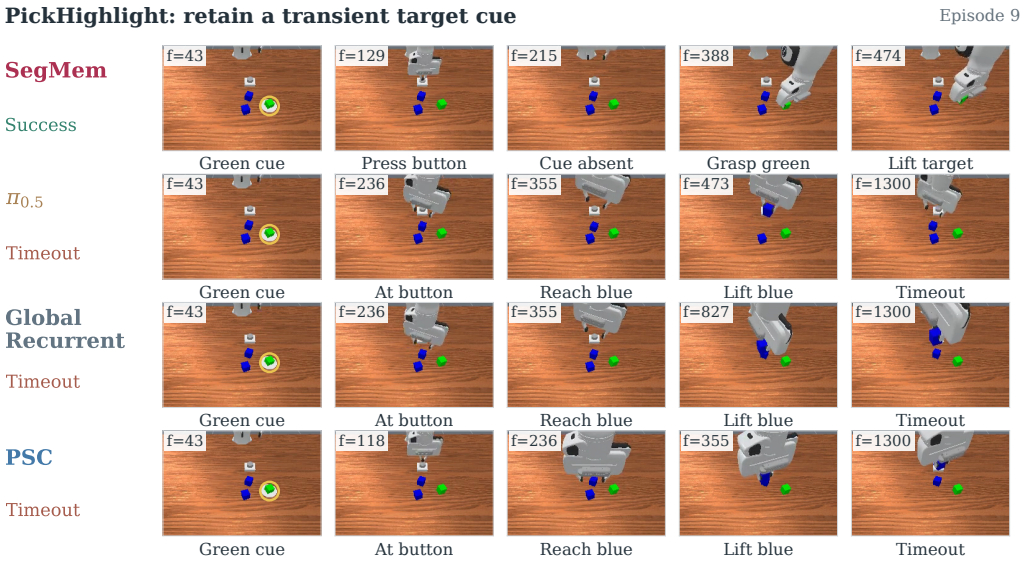


Figure 12: **Transient-cue retention on RoboMME.** In PICKHIGHLIGHT episode 9, SEGMEM lifts the previously highlighted green cube; the baselines manipulate blue distractors and time out. Yellow circles mark the original cue. Frames are selected independently for each row; f denotes the recorded video frame.

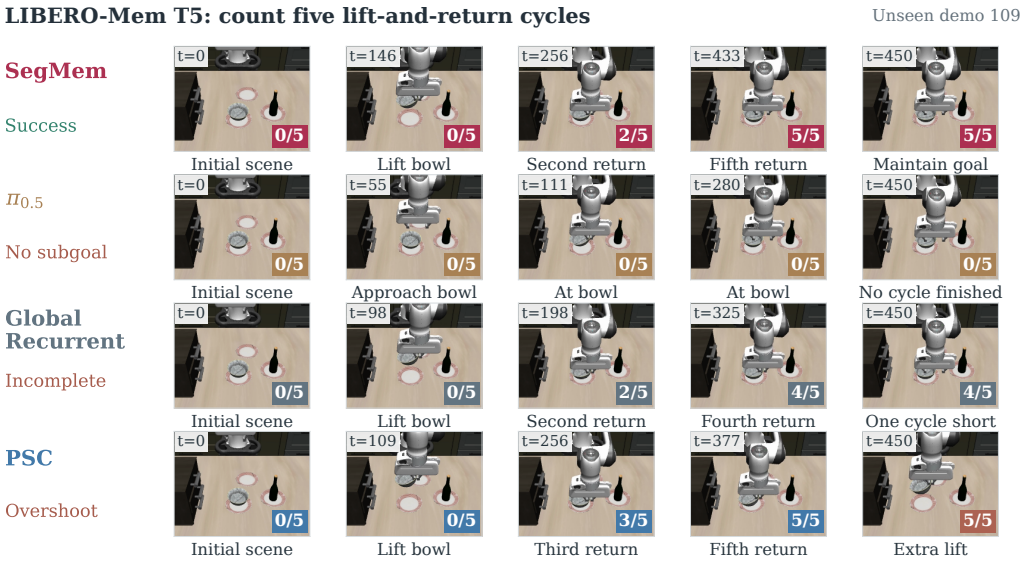


Figure 13: **Progress tracking and completion control on LIBERO-Mem.** T5, unseen demonstration 109. Badges show simulator-recorded completed cycles; t counts executed actions. Images replay archived actions, with all four terminal outcomes verified against the original records. Rows use independently selected action steps.

Table 10: **Outcomes on T5/T6 diagnostic rollouts.** Counts over 40 episodes per method (20 per task; training seed 42). Categories are mutually exclusive. *Incomplete* includes zero progress; *overshoot* is the evaluator's latched loss of the final goal after completion; *goal loss* denotes a false final predicate without overshoot.

Method	Incomplete	Overshoot	Goal loss	Success
$\pi_{0.5}$	36	1	0	3
PSC	0	39	0	1
Global Recurrent	40	0	0	0
SEGMEM	22	7	2	**9**