FLEXMEMWAM: FLEXIBLE TEMPORAL COMPRESSION FOR WORLD ACTION MODELS

Anonymous authors
Paper under double-blind review

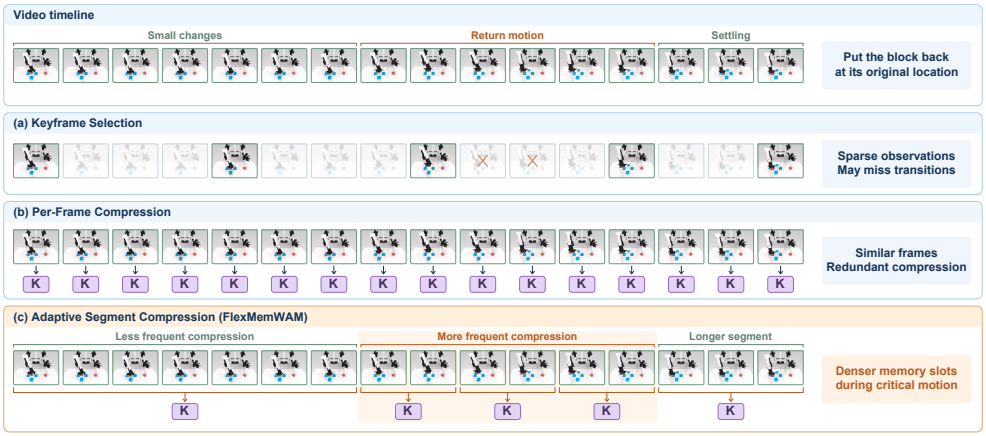


Figure 1: Schematic comparison of historical memory allocation using a Put Back Block trajectory from RMBench. Top: a relatively stable interval, return motion, and settling. (a) Sparse selection may miss intermediate changes. (b) Per-frame compression assigns separate K-slot units even to similar observations. (c) FlexMemWAM groups consecutive observations into variable-length segments, reducing redundant compression in stable intervals and allocating memory slots more densely during critical motion. All purple blocks have the same capacity of K slots. RGB snapshots are sampled uniformly for illustration; their counts do not specify latent-frame segment lengths, and selection and boundaries are schematic. Anchor and retained uncompressed context are omitted.

ABSTRACT

Long-horizon robotic manipulation requires world action models (WAM) to retain historical evidence while controlling memory costs. Task-relevant changes are unevenly distributed over time: many consecutive observations are redundant, whereas brief interactions can introduce important new evidence. Keyframe selection may discard such evidence, while fixed per-frame compression assigns the same memory capacity to every observation. We propose **FlexMemWAM**, which uses online robot-state cues to organize history into variable-length segments, each sharing a fixed set of learnable slots. Once a boundary is causally confirmed, a shared set of learnable slots jointly reads the segment and prior context within the WAM's video Transformer layers, producing layer-wise key–value memories for subsequent action prediction. The memory slots are optimized through joint video and action prediction. FlexMemWAM achieves average success rates of 84.1% on nine history-dependent RMBench tasks and 71.7% on three real-world memory-required tasks, outperforming other strong competitors. We also show that FlexMemWAM effectively reduces the historical KV story compared with per-frame allocation. The code will be made publicly available.

1 INTRODUCTION

Long-horizon robotic manipulation often requires information that is no longer visible. Returning a displaced object to its original location, for example, requires retaining an earlier scene through-

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.

out subsequent actions. World–action models connect visual dynamics learning with action prediction (Ye et al., 2026; Li et al., 2026b; Yuan et al., 2026), but must also organize accumulating observations for history-dependent control. A recent window can discard early cues, whereas retaining every observation incurs growing cache costs (Yang et al., 2026). This history contains both repeated scene content and changes in object and robot states: successive observations during a relatively stable interval may be similar, while a motion or transition introduces new evidence. In this work, we study how to organize and compress this history while retaining evidence useful for subsequent actions.

Keyframe selection and per-frame compression address this problem in different ways. Keyframe selection reduces stored history by retaining representative observations (Zeng et al., 2026), but omitted observations may contain brief cues or intermediate changes. Per-frame compression lets each observation contribute a compact representation (Yang et al., 2026; Wang et al., 2026b), yet assigns separate capacity even to adjacent observations with similar content. These representations may incorporate historical context, but their allocation unit remains an individual observation. Figure 1 illustrates the resulting design question: **which consecutive observations should share one compressed memory unit?**

Our key idea is to let consecutive observations within an interaction segment share compression capacity. Sharing capacity across observations requires deciding where one segment ends and the next begins. Fixed-length grouping may split a continuous process or combine distinct interaction phases as execution speed and pauses vary. Robot-state changes provide online cues for separating these phases, allowing observations within a phase to share a memory unit. We therefore use motion stabilization, gripper transitions, and deceleration to determine variable-length segments.

We propose **FlexMemWAM**, an adaptive segment-memory method within a native world–action model. Once a segment boundary is causally confirmed, a fixed set of learnable slots jointly reads its observations, the initial anchor, and prior memories to produce layer-wise key–value memory. Action prediction reads confirmed memories together with the anchor, recent observations, and the uncompressed open segment. Joint video and action prediction objectives train the slots to retain control-relevant information.

We evaluate FlexMemWAM on nine history-dependent RMBench tasks (Chen et al., 2026) and three real-world robot tasks, achieving average success rates of **84.1%** and **71.7%**, respectively. The RMBench average is the highest among the compared methods.

Our contributions are threefold:

1. **State-guided adaptive temporal grouping.** We use causally confirmed robot-state cues to determine which consecutive observations share a memory unit.
2. **Joint segment compression within WAM.** We construct fixed-capacity, layer-wise segment memory using shared learnable slots, and combine it with the initial anchor and recent/open observations for action prediction.
3. **Strong performance on memory-dependent robot tasks.** FlexMemWAM achieves the highest average success rate among the compared methods on RMBench and demonstrates strong performance on real-world tasks requiring memory.

2 Related Work

2.1 World–Action Models

RT-1, RT-2, and OpenVLA have advanced large-scale visuomotor control and vision-language-action learning (Brohan et al., 2023; Zitkovich et al., 2023; Kim et al., 2025), while UniPi, GR-1/2, and VPP explore visual prediction for robot control (Du et al., 2023; Wu et al., 2024; Cheang et al., 2024; Hu et al., 2025). World–action models connect visual dynamics learning with action generation. DreamZero jointly models visual dynamics and actions with a pretrained video diffusion model, while LingBot-VA supports continuous control through causal video–action modeling and history caching (Ye et al., 2026; Li et al., 2026b). Fast-WAM uses video supervision during training without generating future video at inference (Yuan et al., 2026). Building on this line of

work, we study how world–action models can organize and compress accumulating visual history for memory-dependent action prediction.

2.2 Memory for Long-Horizon Robotic Manipulation

Robot memory methods must decide what history to retain and how to make it available to current decisions. MemoryVLA retrieves and integrates history through a perception–cognition memory bank (Shi et al., 2026); ReMem-VLA propagates information through frame-level and temporal-chunk recurrent queries (Li et al., 2026a); and KEMO combines robot kinematics with visual filtering to select event-related keyframes (Zeng et al., 2026).

Compression-based methods reduce the representation cost of retained observations. MemoryWAM generates history-conditioned gist tokens for each observation and caches their layer-wise keys and values, while NativeMEM compresses each historical frame–view pair into a single token using the policy's vision encoder (Yang et al., 2026; Wang et al., 2026b). These methods allocate compressed capacity primarily per observation.

Other methods organize memory around skills, events, or plans. HiMem-WAM uses predicted skill boundaries to gate writes of task states formed from current representations, predicted skills, and pooled low-level latent-action chunks (Sun et al., 2026). DiM-WAM maintains multiple banks of historical event memory through similarity-based merging, while MaP-WAM converts episodic history into compact plans for subsequent execution (Wang et al., 2026a; Zhao et al., 2026).

FlexMemWAM focuses on the *source span for joint compression*. Following KEMO's use of robot-state changes as interaction cues, we use these cues to define segment boundaries rather than select individual observations to retain. A fixed set of shared slots jointly integrates all observations in each closed segment into layer-wise K/V memory, making the variable-length segment the unit of compression.

2.3 Context Compression and Memory Management

Learnable latent representations and summary tokens compress variable-length inputs into compact context (Jaegle et al., 2022; Alayrac et al., 2022; Mu et al., 2023; Chevalier et al., 2023). Long-sequence methods complement compression with state reuse, recurrent memory, or selective cache retention (Dai et al., 2019; Bulatov et al., 2022; Rae et al., 2020; Xiao et al., 2024). Building on these mechanisms, FlexMemWAM studies which consecutive robot observations should share a fixed-capacity memory unit, using state cues to adapt the source span for joint compression.

3 FlexMemWAM: Adaptive Memory Compression

Inputs and backbone. At each planning step t, the policy receives visual observations, the current robot state, and a task instruction c, and predicts a chunk of H control actions. As shown in Figure 2, the backbone contains a video expert and an action expert. A causal VAE encodes the observations into visual tokens, which are processed by the video expert's Transformer layers. We denote the tokenized observation at step t by z_t; one planning observation is one causal-VAE latent frame (16 control steps per planning interval in RMBench; Appendix A.1). The action expert encodes a noisy action chunk, processes it conditioned on the robot state and visual history, and maps its output back to action space. At each layer, action queries attend to visual-history keys and values together with the action branch's own keys and values. Both experts receive instruction features through text cross-attention. At inference, the action chunk is initialized with Gaussian noise and iteratively denoised into predicted controls.

Historical context. Past observations are made available through cached, layer-wise key–value (K/V) representations. Keys determine attention weights for the current queries, and values supply the corresponding historical information. FlexMemWAM organizes this context into the initial-observation anchor, compressed *segment memories*, and uncompressed recent/open observations. The open segment contains observations whose compression boundary has not yet been confirmed. The anchor is denoted by A_0 in Figure 2 and by $\mathcal{Z}_0^{(\ell)}$ for its K/V at layer ℓ. These historical K/V representations are reused across action-denoising steps.

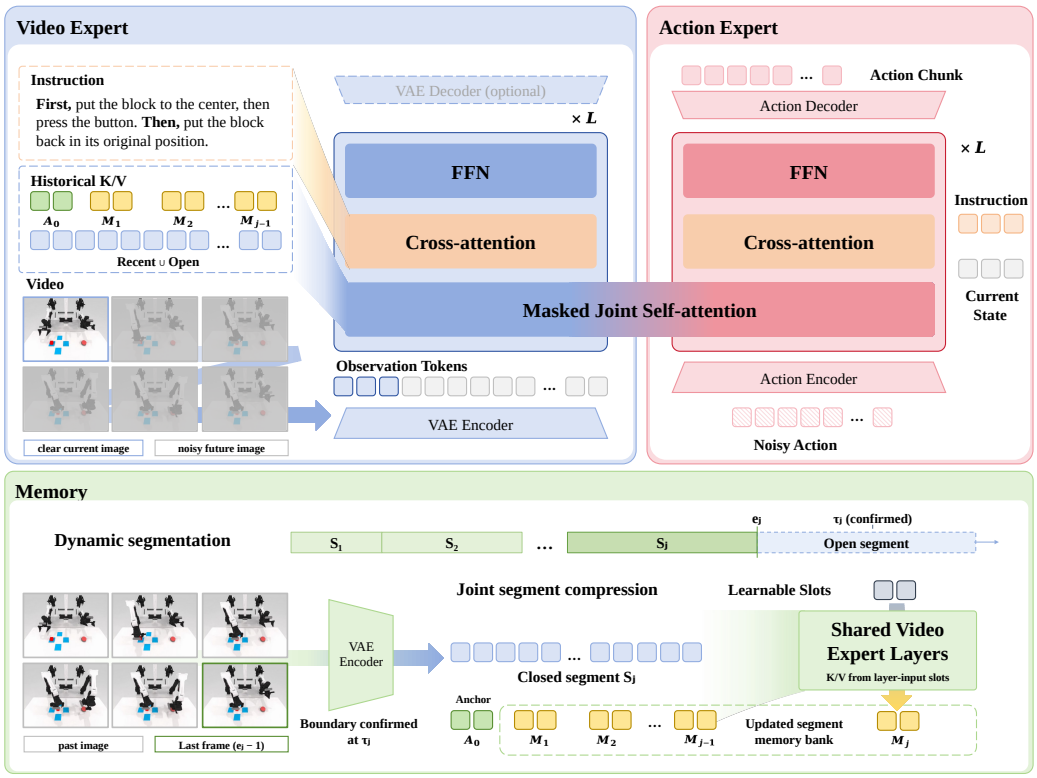


Figure 2: Overview of FlexMemWAM. Top: the video and action experts with historical K/V conditioning. Bottom: (1) robot-state cues define causally confirmed, variable-length segments; (2) shared slots jointly integrate each closed segment with the anchor and prior memory in native WAM layers, with K/V projected from each layer's input slot states; (3) new segment memory is appended without modifying prior memories. Action prediction reads this memory together with the anchor and retained recent/open observations, conditioned on the instruction and current robot state.

Adaptive segment memory. Online robot-state cues group consecutive observations into variable-length segments. Once a boundary is causally confirmed, shared learnable slots jointly read the closed segment, the anchor, and prior memories within the video expert's layers. Their layer-wise K/V form a new segment memory, appended without modifying earlier memories. Action prediction reads the updated context together with the retained observations. Adaptivity concerns the temporal span compressed into each memory unit; the number of slots per unit remains fixed. Segment lengths and recent-window sizes count planning observations, with their temporal representation specified in Appendix A.

3.1 Interaction-Aware Causal Temporal Grouping

Interaction changes as grouping cues. We use three robot-state signals for grouping: post-motion stabilization, gripper endpoint transitions, and local deceleration. Unlike keyframe selection, these cues determine which consecutive observations are compressed together and when a new segment memory is written. A proposed boundary is committed only after its confirmation evidence has arrived.

Online accumulation and segment closure. History after the initial observation is organized into consecutive segments $S_j = [s_j, e_j)$, where $s_1 = 1$, $s_{j+1} = e_j$, and $L_j = e_j - s_j$. Observations accumulate from s until a causally confirmed boundary satisfies $L_{\min} \le b - s \le L_{\max}$, or the maximum length is reached. Let $b_t^\star$ denote the selected valid candidate at time t; candidate ordering is specified in Appendix A. The closure rule is

$$\mathrm{Close}(s,t) = \begin{cases} b_t^\star, & \text{if a confirmed valid candidate exists,} \\ s + L_{\max}, & \text{if none exists and } t = s + L_{\max}, \\ \perp, & \text{otherwise.} \end{cases} \tag{1}$$

Here, $\perp$ means that observation accumulation continues. The minimum length prevents overly frequent writes, while the maximum length bounds the temporal span that a single memory unit must integrate.

The boundary location e_j and confirmation time τ_j may differ: a segment covers $[s_j, e_j)$, but its memory becomes available only from τ_j onward. For example, if a segment starts at step 1 and a boundary at step 6 is confirmed at step 8, observations 1–5 form the closed segment; observations from step 6 onward remain in the open tail. Until confirmation, the source observations remain accessible in full resolution. Appendix A details the signals and online procedure.

3.2 Fixed-Capacity Joint Segment Writing

A shared slot set for the entire segment. For a closed segment containing L_j observations, each observation contributes P visual tokens. We use K learnable slots to jointly integrate all $L_j P$ source positions, with $K = 8$ in our experiments. Memory capacity is fixed *per segment*: longer segments share the same number of slots among more observations, allocating K/L_j persistent positions per observation on average.

We use learnable memory slots for aggregation (Mu et al., 2023; Yang et al., 2026), with the dynamically selected segment defining their joint input span. Every write starts from the same learned parameters $E \in \mathbb{R}^{K \times d_v}$, where d_v is the visual hidden dimension. At each native WAM layer, the K slot queries jointly attend to all $L_j P$ source positions in the segment, the initial anchor, earlier segment memories, and the slots themselves. Thus, the segment shares one query set rather than producing separate per-frame summaries for later merging.

Shared capacity across observations. Similar observations within a segment share one slot set instead of each receiving a separate memory allocation. Joint writing lets the slots aggregate repeated content and changes within this shared capacity, while prediction objectives determine what information is retained. Source observations preserve their spatiotemporal positions and causal within-segment context, so joint compression can also use evidence about changes across observations.

Let $\mathcal{Z}_i^{(\ell)}$ denote the source keys and values of observation i at layer ℓ, $\mathcal{Z}_{S_j}^{(\ell)}$ their concatenation over the segment, and $\mathcal{M}_{<j}^{(\ell)}$ the concatenation of earlier segment memories. Slot states are updated as

$$\begin{aligned} U_j^{(0)} &= E, \\ U_j^{(\ell)} &= \mathcal{B}_\theta^{(\ell)}\left(U_j^{(\ell-1)}; \mathcal{Z}_0^{(\ell)}, \mathcal{M}_{<j}^{(\ell)}, \mathcal{Z}_{S_j}^{(\ell)}, c\right). \end{aligned} \tag{2}$$

Here, $\mathcal{B}_\theta^{(\ell)}$ comprises the native attention, residual, instruction cross-attention, and feed-forward operations. Writing parameters are shared across segments, while the resulting slot states depend on each segment's observations and historical context.

Layer-wise readable memory. At layer ℓ, the cached K key–value pairs are projected from the input slot states $U_j^{(\ell-1)}$, before that layer's update. The first layer therefore uses the shared initialization, while later layers receive segment information integrated by preceding layers. These per-layer caches together form M_j, which is appended to persistent memory after writing. The available memory is $\mathcal{M}_t = \{M_j : \tau_j \le t\}$. Appendix B.2 specifies positional encoding, and Appendix B.1 gives the attention masks.

3.3 Long-Term Memory and Recent Context

A source observation remains directly accessible while it belongs to the recent window or the open segment. Except for the initial anchor, it is evicted only after its segment has been compressed and

it leaves the retained context. Closing a segment therefore need not immediately remove its source observations: recent sources and their segment memory may temporarily coexist.

Let $\mathcal{R}_t$ index the most recent R planning observations and $\mathcal{U}_t$ all arrived observations in the open segment. Together with the initial anchor, they define the retained observation set and layer-wise historical context:

$$\mathcal{F}_t = \{0\} \cup \mathcal{R}_t \cup \mathcal{U}_t, \qquad \mathcal{C}_t^{(\ell)} = \left[\mathcal{Z}_{\mathcal{F}_t}^{(\ell)}; \mathcal{M}_t^{(\ell)}\right]. \tag{3}$$

Here, $\mathcal{Z}_{\mathcal{F}_t}^{(\ell)}$ and $\mathcal{M}_t^{(\ell)}$ concatenate the keys and values of retained observations and confirmed memories, respectively. The union stores overlapping observations once and preserves the entire open tail, even when it extends beyond the recent window.

The action branch reads this historical context and its own target block at every layer to denoise an action chunk, conditioned on the task instruction and current robot state. Action and video targets each read the available historical context and their own target block, but do not attend to each other. Figure 5 in Appendix B.1 provides the complete attention masks; Appendix B details cache maintenance.

Context updates after confirmation. After a boundary is confirmed, already-arrived observations beyond it remain in the open tail. Their source K/V pairs are recomputed under the new segment start before the current observation is encoded; Appendix B details this cache update.

If J_t segments have been confirmed, the number of historical cache positions per layer is

$$N_t = P|\mathcal{F}_t| + KJ_t. \tag{4}$$

The first term preserves full-resolution context for current control; the second retains compressed history. Capacity is fixed per segment, but the total cache grows with the number of segments.

3.4 Memory Training and Action Inference

During training, the video expert predicts future visual observations and the action expert predicts the corresponding action chunk. Each target branch receives independently noised targets and reads the available historical context, but the two branches cannot attend to each other's target tokens. Video prediction thus provides auxiliary supervision for learning representations useful to control.

Training follows the same confirmation-time visibility rule as online prediction: both target branches read only memories with $\tau_j \leq t$ and the retained observations in Equation 3. Older source observations outside the retained full-resolution context are accessible to subsequent predictions only through confirmed segment memory. Training reconstructs memory in a differentiable forward pass, without detaching it at segment boundaries; Appendix C.1 details gradient flow and trainable modules. We use joint video and action flow-matching objectives (Yuan et al., 2026):

$$\mathcal{L} = \lambda_a \mathcal{L}_{\text{action}} + \lambda_v \mathcal{L}_{\text{video}}. \tag{5}$$

Boundaries follow the causal grouping rule and are not optimized by gradient descent; the prediction objectives learn the memory content. At inference, the model encodes observed history and updates memory, then denoises only the action chunk using the cached context; future video generation is not required. Appendix C.1 details the flow-matching targets and historical-conditioning augmentation, and Appendix B.1 specifies the attention masks.

4 Experiments

We evaluate memory-dependent control on nine RMBench tasks, examine temporal compression and memory components through ablations, and report real-world results. A separate cache-capacity analysis measures the storage required by different history-organization strategies.

Table 1: Success rates (%) on nine history-dependent RMBench tasks. Official baseline results are taken from RMBench (Chen et al., 2026); the per-task results for Fast-WAM, LingBot-VA, and MemoryWAM are taken from Table 1 of MemoryWAM (Yang et al., 2026), not from new evaluations in this work. FlexMemWAM is evaluated over 100 trials per task. Group averages are computed from the per-task rates; All is the unweighted mean across nine tasks. Observe & Pick abbreviates Observe and Pick Up. Bold and underlined entries denote the best and second-best result in each column, respectively.

	$M^{(1)}$: single critical memory					$M^{(n)}$: multiple historical memories				Group average		
Method	Observe & Pick	Rearrange Blocks	Put Back Block	Swap Blocks	Swap T	Battery Try	Blocks Ranking	Cover Blocks	Press Button	$M^{(1)}$	$M^{(n)}$	All
Official RMBench baselines												
DP	1%	0%	0%	11%	20%	10%	10%	0%	0%	6.4%	5.0%	5.8%
ACT	1%	29%	0%	2%	2%	19%	0%	0%	0%	6.8%	4.8%	5.9%
$\pi_{0.5}$	9%	13%	11%	24%	15%	16%	6%	0%	0%	14.4%	5.5%	10.4%
X-VLA	9%	13%	18%	16%	3%	26%	1%	2%	0%	11.8%	7.3%	9.8%
Mem-0	4%	<u>89%</u>	<u>90%</u>	67%	14%	28%	18%	68%	0%	52.8%	28.5%	42.0%
World–action models												
Fast-WAM	0%	0%	0%	0%	7%	20%	26%	0%	0%	1.4%	11.5%	5.9%
LingBot-VA	13%	**100%**	**100%**	<u>99%</u>	<u>88%</u>*	<u>41%</u>	**100%**	79%	<u>84%</u>	80.0%	76.0%	78.2%
MemoryWAM	<u>27%</u>	**100%**	**100%**	**100%**	**94%***	<u>41%</u>	**100%**	<u>98%</u>	**87%**	**84.2%**	<u>81.5%</u>	<u>83.0%</u>
FlexMemWAM (Ours)	**31%**	**100%**	**100%**	**100%**	76%	**83%**	**100%**	**100%**	67%	<u>81.4%</u>	**87.5%**	**84.1%**

*Swap T results use action-history conditioning, as described in Appendix A.1 of MemoryWAM (Yang et al., 2026). FlexMemWAM does not use action-history conditioning on Swap T.

4.1 Implementation Details

Model configuration. The model uses the pretrained video DiT, text encoder, and 3D causal VAE of Wan2.2-TI2V-5B (Team Wan et al., 2025; Peebles & Xie, 2023), together with a separate action expert (Yuan et al., 2026). The three camera views are concatenated into one input. For RMBench, robot states and actions use 14-dimensional dual-arm joint vectors, and each action chunk spans 16 control steps. We set the segment length range to 3–8 planning observations, the memory capacity to eight slots per segment, and the recent window to four planning observations. Action inference uses 10 denoising steps. Appendix C provides the remaining configurations.

Training configuration. We train a separate model for each RMBench task using joint video and action flow matching (Lipman et al., 2023). Historical conditions receive Gaussian noise with varying strength during training. The optimizer, noise schedule, and loss weights are specified in Appendix C.1.

4.2 Main Results on RMBench

We organize results by the Task Memory Complexity (TMC) defined in RMBench. $M^{(1)}$ tasks depend on a single critical historical state, whereas $M^{(n)}$ tasks require history to be continually retained and updated over multiple interactions. We compare with Fast-WAM (Yuan et al., 2026), LingBot-VA (Li et al., 2026b), and MemoryWAM (Yang et al., 2026); these citations identify the methods, while the numerical sources are specified in the table caption. The table also includes DP (Chi et al., 2023), ACT (Zhao et al., 2023), $\pi_{0.5}$ (Black et al., 2025), X-VLA (Zheng et al., 2026), and Mem-0 (Chen et al., 2026) from the official RMBench evaluation for a broader comparison.

Table 1 compares FlexMemWAM with representative robot policies and world–action models on RMBench. FlexMemWAM achieves the highest average success rate of **84.1%**. It also achieves the highest average on $M^{(n)}$ tasks, which require retaining and updating history across multiple interactions.

4.3 Ablation Study

We ablate temporal compression and memory components on Swap T and Cover Blocks, representing single-critical-state and multiple-history memory tasks, respectively. Frame-wise independently compresses each planning observation into eight slots. Fixed-4 jointly compresses every four planning observations into eight slots. Both retain the anchor and recent window. The w/o Anchor

Table 2: Ablations of temporal compression, the initial-frame anchor, and segment memory. Each variant is trained independently and evaluated over 100 trials per task. We report success rates (%); bold and underlined entries denote the best and second-best results, respectively.

Method	Swap T	Cover Blocks
FlexMemWAM (Ours)	**76%**	**100%**
Frame-wise	<u>66%</u>	<u>98%</u>
Fixed-4	65%	92%
w/o Anchor	16%	61%
w/o Segment Memory	23%	69%

variant removes the initial-frame anchor from both writing and reading; w/o Segment Memory removes only compressed segment memories while retaining the same initial anchor and deduplicated recent/open observations ($\mathcal{R}_t \cup \mathcal{U}_t$) as the full model.

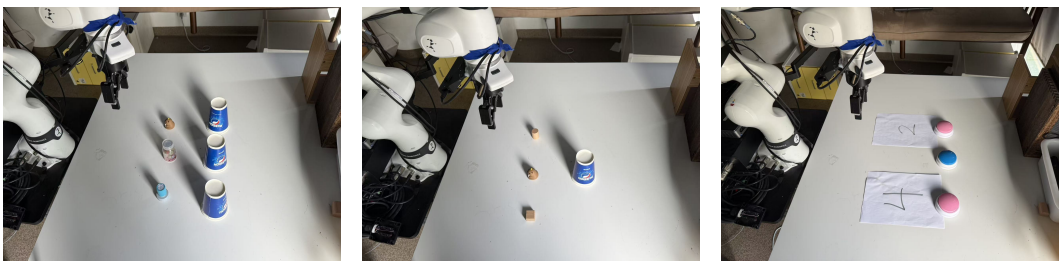
Effect of temporal compression. FlexMemWAM achieves success rates 10 and 2 percentage points higher than frame-wise compression on Swap T and Cover Blocks, respectively (Table 2). Compared with Fixed-4, adaptive grouping improves success by 11 and 8 percentage points. The observed gains support segment-level compression and adaptive grouping under the tested settings, although frame-wise compression performs similarly on Cover Blocks (98% versus 100%).

Effect of memory components. Removing the anchor reduces success from 76% to 16% on Swap T and from 100% to 61% on Cover Blocks, supporting the role of an initial-scene reference. Removing segment memory reduces success to 23% and 69%, respectively, despite retaining the same anchor and recent/open context as the full model. This supports the contribution of compressed segment memory beyond the retained uncompressed context. Both components contribute to performance on the two tested tasks.

4.4 Real-World Experiments

Table 3: Real-world success rates (%) over 20 trials per task for each method. Average is the unweighted mean across the three tasks. Bold and underlined entries denote the best and second-best results, respectively.

Method	Sequential Uncovering	Hidden-Object Matching	Press Button	Average
Fast-WAM	0%	15%	0%	5.0%
LingBot-VA	<u>60%</u>	<u>70%</u>	<u>50%</u>	<u>60.0%</u>
FlexMemWAM (Ours)	**75%**	**80%**	**60%**	**71.7%**



(a) Sequential Uncovering (b) Hidden-Object Matching (c) Press Button

Figure 3: Real-world task setups: (a) Sequential Uncovering, (b) Hidden-Object Matching, and (c) Press Button.

Figure 3 shows the setups for our three real-world tasks. Sequential Uncovering tests memory of object locations under occlusion and execution order; Hidden-Object Matching tests matching after

re-occlusion; and Press Button tests counting followed by confirmation. We use a single-arm Franka Research 3 (FR3) robot equipped with left, right, and wrist cameras, with 50 training demonstrations per task. Appendix C.2 provides task instructions and success criteria.

FlexMemWAM achieves the highest full-task success rate on all three real-world tasks (Table 3). Compared with LingBot-VA, success improves by 15 percentage points on Sequential Uncovering and 10 points each on Hidden-Object Matching and Press Button. Average success increases from 60.0% to 71.7%, a gain of 11.7 percentage points; Fast-WAM achieves 5.0%. These improvements span tasks requiring object-location and order memory, matching after re-occlusion, and counting.

4.5 Historical KV-Cache Scaling

We compare historical K/V storage across all layers of the same BF16 model using synthesized layouts (Figure 4). This analysis measures cache capacity and excludes the runtime costs of memory writing, suffix recomputation, and action inference.

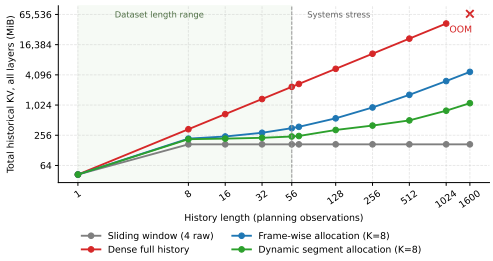


Figure 4: Historical KV-cache capacity of synthesized layouts. Shading: dataset history-length range; crosses: analytical dense-history demand after OOM.

Sliding window retains only the four most recent planning observations, while dense history retains all observations. Per-frame allocation assigns eight slots per evicted observation; dynamic allocation assigns eight per closed segment. Both retain the initial observation and the four most recent observations, and dynamic allocation additionally retains the open tail. Dynamic layouts use empirical segment lengths and confirmation delays from Battery Try; Appendix C.1 details their construction.

At 56 planning observations, the maximum history length in the Battery Try data, dynamic allocation requires 241.9 MiB of historical K/V cache, compared with 354.4 MiB for per-frame allocation: a reduction of 31.7%. These totals include both compressed memory and retained full-resolution observations.

Beyond the dataset's history-length range, the stress test shows a widening storage gap between segment-level and per-frame allocation. Segment memory retains compressed information from older history, while its cache grows with the number of closed segments; the sliding window maintains constant storage by discarding older observations.

5 Conclusion and Limitations

We presented FlexMemWAM, which uses robot-state cues to organize visual history into variable-length segments and jointly compresses each segment into a fixed set of learnable slots within a native world–action model. This design adapts the temporal span of each memory unit while keeping its slot capacity fixed. FlexMemWAM achieves an average success rate of 84.1% across nine history-dependent RMBench tasks and 71.7% across three real-world tasks. Ablations on Swap T and Cover Blocks support adaptive segment compression and the contributions of both the initial anchor and segment memory under the tested settings, while a separate analysis of synthesized cache layouts shows lower historical K/V storage requirements than per-frame allocation. These results support temporal grouping as a useful design choice for compressed robot memory.

FlexMemWAM currently relies on predefined robot-state signals and segment-length constraints, and these signals may not capture all task-relevant visual changes. Its persistent cache also grows with the number of closed segments, while fixed-capacity compression may discard fine-grained information within a segment. Future work includes learning task-aware segment boundaries and managing accumulated memories under a fixed total budget. End-to-end runtime evaluation is also needed to account for memory writing and suffix recomputation in addition to cache storage.

STATEMENT ON THE USE OF AI TOOLS

We used generative AI tools to assist with manuscript drafting, translation and language editing, literature search, preparation of schematic figures, development of analysis scripts, and LaTeX typesetting. The authors take responsibility for the final content of the paper, including all AI-assisted text, code, and figures.

REPRODUCIBILITY STATEMENT

Section 3 describes the temporal grouping, segment-memory writing, and memory-reading mechanisms. Appendix A provides the grouping signals, boundary confirmation rules, and related parameters; Appendix B describes attention visibility, positional encoding, and cache maintenance. Section 4.1 and Appendix C.1 provide the model configuration, training objectives, optimization settings, and evaluation protocol. Appendix C.2 describes the real-world robot task setups and success criteria. Table 1 identifies the sources of the baseline results. Appendix C.1 also describes the synthesized K/V-cache measurement procedure and explicitly distinguishes this capacity measurement from policy evaluation and end-to-end runtime measurement.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: A visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2204.14198.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert 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. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/black25a.html.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil 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 S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H. Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems XIX*, 2023. URL https://roboticsproceedings.org/rss19/p025.html.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/47e288629a6996a17ce50b90a056a0e1-Abstract.html.

Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, Hanbo Zhang, and Minzhao Zhu. GR-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024. URL https://arxiv.org/abs/2410.06158.

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

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://aclanthology.org/2023.emnlp-main.232/.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin C. M. Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems XIX*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://roboticsproceedings.org/rss19/p026.html.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019. URL https://aclanthology.org/P19-1285/.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 9156–9172. Curran Associates, Inc., 2023. doi: 10.52202/075280-0403. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1d5b9233ad716a43be5c0d3023cb82d0-Paper-Conference.pdf.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://proceedings.mlr.press/v267/hu25g.html.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=fILj7WpI-g.

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 *Proceedings of The 8th Conference on Robot Learning*, 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. *arXiv preprint arXiv:2603.12942*, 2026a. URL https://arxiv.org/abs/2603.12942.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Luyao Zhang, Mingrui Yu, Zelin Gao, Nan Xue, Boyu Zhou, Xing Zhu, Mingyu Ding, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2026b. doi: 10.15607/RSS.2026.XXII.016. URL https://www.roboticsproceedings.org/rss22/p016.html.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/3d77c6dcc7f143aa2154e7f4d5e22d68-Abstract.html.

Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, Lunkai Lin, Zhiqiang Xie, Mingyu Ding, and Ping Luo. RoboTwin: Dual-arm robot benchmark with generative digital twins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27649–27660, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Mu_RoboTwin_Dual-Arm_Robot_Benchmark_with_Generative_Digital_Twins_CVPR_2025_paper.html.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Peebles_Scalable_Diffusion_Models_with_Transformers_ICCV_2023_paper.html.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SylKikSYDH.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=54U3XHf7qq.

Xiaoquan Sun, Ruijian Zhang, Chen Cao, Yihan Sun, Jiahui Chen, Zetian Xu, Bo Chen, Haijier Chen, Zhen Yang, Jiarun Zhu, Yijun Hong, JingZhe Xu, Jingrui Pang, Mingqi Yuan, and Jiayu Chen. HiMem-WAM: Hierarchical memory-gated world action models for robotic manipulation. *arXiv preprint arXiv:2606.10363*, 2026. URL https://arxiv.org/abs/2606.10363.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. URL https://arxiv.org/abs/2503.20314.

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

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

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NxoFmGgWC9.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.17453.

Sizhe Yang, Juncheng Mu, Tianming Wei, Chenhao Lu, Xiaofan Li, Linning Xu, Zhengrong Xue, Zhecheng Yuan, Dahua Lin, Jiangmiao Pang, and Huazhe Xu. MemoryWAM: Efficient world action modeling with persistent memory. *arXiv preprint arXiv:2606.20562*, 2026. URL https://arxiv.org/abs/2606.20562.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi Jim Fan, and Joel Jang. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026. URL https://arxiv.org/abs/2602.15922.

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

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

Sizhe Zhao, Haozhe Xie, Weiyu Zhao, Chenchu Zhang, Huan Wang, Chenyang Wang, Qinglin Liu, and Shengping Zhang. Memory as plans: World-action modeling with memory-grounded

planning. *arXiv preprint arXiv:2609.11561*, 2026. URL https://arxiv.org/abs/2609.11561.

Tony Z. Zhao, Vikash Kumar, Sergey 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. URL https://roboticsproceedings.org/rss19/p016.html.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Tai Wang, Ya-Qin Zhang, Jingjing Liu, and Xianyuan Zhan. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 60580–60606, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/630d293833e09e1ecd892a898a20b074-Paper-Conference.pdf.

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 *Proceedings of The 7th Conference on Robot Learning*, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A Additional Details on Adaptive Temporal Grouping

A.1 State Signals and Causal Confirmation

Temporal units and observation alignment. We distinguish raw control steps, sampled RGB observations, and planning observations. In RMBench, the three camera views are combined into one image at each sampled time. Images are sampled every four control steps and encoded as a continuous stream by the frozen causal VAE, whose fourfold temporal compression produces one latent frame per 16 control steps after the initial frame. Thus, planning observation z_t is aligned with control step $16t$: z_0 is the initial anchor, and each subsequent latent summarizes the next four sampled images using the preceding causal VAE context. The VAE stream is not restarted at action-chunk boundaries. At decision t, history includes $z_0, \ldots, z_t$, and the policy predicts the next $H = 16$ control actions before replanning. Segment lengths and recent-window sizes count these latent frames, rather than RGB images or raw control steps. Here, *raw* observations mean latent representations before segment compression, not RGB pixels; the RGB snapshots in Figure 1 only illustrate the trajectory.

Boundary coordinates. Robot-state cues are evaluated at raw control-step resolution. Candidate u has change location r_u, evidence-availability time a_u, and latent boundary $b_u = \lceil r_u/16 \rceil$. At planning step t, it can be considered only if $a_u \le 16t$. A closed segment $[s_j, e_j)$ contains $e_j - s_j$ planning observations; the boundary observation z_{e_j} belongs to the next segment. We use three state signals:

- **Post-motion stabilization.** We normalize non-gripper joint differences by their historical per-dimension median scales and smooth the root-mean-square velocity to obtain v_r. Using the maximum velocity $v_r^\star$ and maximum velocity percentile $p_r^\star$ in the preceding motion window, we define

$$d_r = \max\left(0, \log \frac{v_r^\star + 10^{-6}}{v_r + 10^{-6}}\right), \qquad h_r = \sqrt{\rho_r(d)\, p_r^\star}.$$

  Here, $\rho_r(d)$ denotes the empirical percentile of the current value d_r relative to at most 128 preceding raw control steps. A local peak above the threshold becomes a valid candidate after its right-hand confirmation window has arrived.
- **Gripper endpoint transition.** A candidate is generated when a gripper moves from one open/closed endpoint interval into the other; the intermediate interval does not update the endpoint state. This candidate is available as soon as the transition occurs.
- **Local deceleration.** We apply trailing smoothing to the displacement norm of full-state differences, obtaining $\bar{\delta}_r$, and detect deceleration as local peaks of $(1 + \bar{\delta}_r)^{-1}$. A candidate becomes available only after its right-hand confirmation window has arrived.

The stabilization signal uses an 8-step smoothing window and a 32-step motion-reference window, while the deceleration signal uses 10-step smoothing. Their peak-confirmation radii are 4 and 30 raw steps, and their minimum candidate spacings are 16 and 8 steps, respectively. The first arrival at a gripper endpoint only initializes its state.

A.2 Key Parameters and Online Updates

Table 4: Key parameters for adaptive temporal grouping and fixed-capacity segment memory.

Parameter	Value
Segment length range	3–8 latent frames
Memory slots per segment K	8
Initial anchor	First latent frame
Recent uncompressed window R	4 latent frames
Stabilization candidate threshold	0.95
Gripper open/closed endpoint thresholds	0.2, 0.8
Candidate-confirmation delay bound	2 latent-frame steps

The confirmation delay is $\lceil a_u/16 \rceil - b_u$, bounded by two latent-frame steps by the 30-control-step maximum confirmation window; it is not a separate timeout parameter.

Starting from $s = 1$, each planning step t proceeds as follows:

1. Read arrived robot states and retain candidates with $a_u \leq 16t$ and $L_{\min} \leq b_u - s \leq L_{\max}$.
2. Sort valid candidates by evidence-availability time, boundary location, and signal type. Close at the first candidate; if none exists, close at $s + L_{\max}$ when the maximum length is reached. Otherwise, continue accumulating the open segment.
3. If a segment closes at e, write its memory, set $s \leftarrow e$, and recompute the already-arrived latent-frame suffix $[e, t)$.
4. Encode the current latent frame for action prediction. It belongs to the open tail; at most one segment closes per step.

After any closure, recheck candidates against the new start; discard late candidates at or before it.

B Attention and Memory Caching

B.1 Attention Visibility

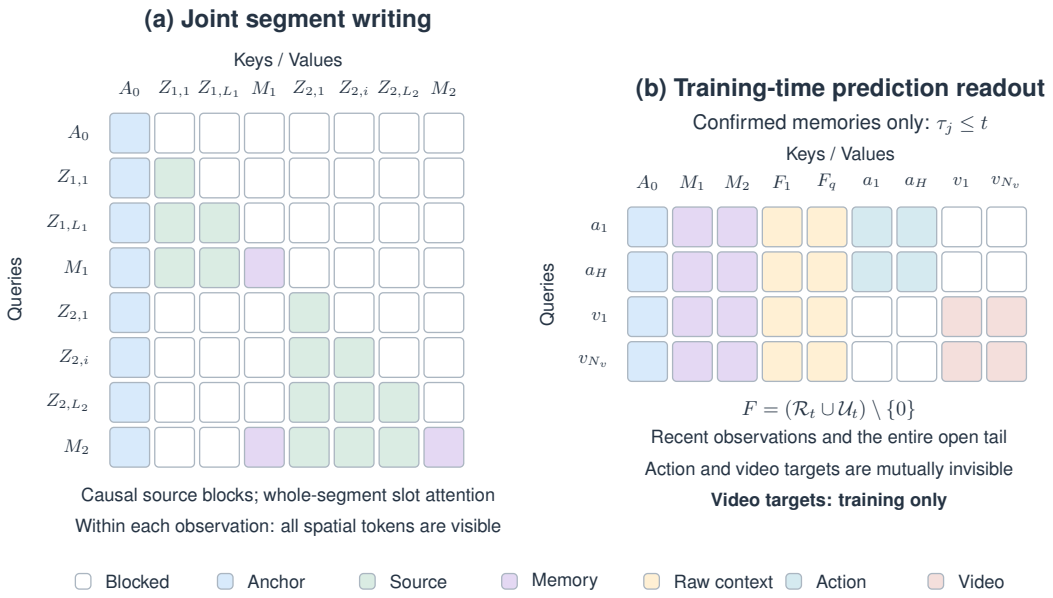


Figure 5: Block-level attention masks: rows are queries, columns are keys/values, and white cells are masked. (a) $Z_{j,i}$: source observation; M_j: segment slots. (b) $F_1, \ldots, F_q$: deduplicated non-anchor recent/open observations; a, v: action and video targets; H: action-chunk length; N_v: video-target token count.

Each source block contains mutually visible spatial tokens and reads the anchor and causal source blocks in its segment, but no memory slots. Slots read the anchor, prior memories, all segment sources, and themselves. Target branches read the historical context and their own target block, remaining mutually invisible; video targets are training-only. Instructions use separate cross-attention. Training and inference both follow the confirmation-time visibility rule in Section 3.4.

B.2 Positional Encoding and Cache Maintenance

Source observations preserve their own spatiotemporal positions. Slots use the temporal position of the segment's final source observation, $e_j - 1$, together with identity spatial rotation. Slot initialization, native-layer updates, and layer-wise K/V projection follow Section 3.2.

The cache retains the anchor, confirmed memories, and deduplicated recent/open observations as in Equation 3. On confirmation at τ_j, recompute suffix K/V pairs for $[e_j, \tau_j)$ under the new segment start before encoding the current observation. Historical K/V pairs remain fixed across action-denoising steps; action-side queries and keys are updated each step.

C Additional Experimental Details

C.1 Training and Evaluation

Both experts have 30 layers, with video/action hidden dimensions of 3072/1024 and approximately 6B parameters in total. The action expert is initialized by interpolating the pretrained video DiT along the hidden dimension. The three RMBench views form a 384×320 input, producing 120 visual tokens per latent frame and a 16-step action chunk. Each task is trained independently on official RMBench demonstrations (Chen et al., 2026) using eight GPUs. Table 5 lists optimization settings; noise times follow a shifted logit-normal distribution.

Table 5: Main training settings for RMBench.

Parameter	**Value**
Optimizer	AdamW
Peak learning rate	1×10^{-4}
Learning-rate schedule	Cosine decay
Warmup	First 5% of training steps
Weight decay	0.01
Optimizer β	$(0.9,\ 0.95)$
Per-GPU batch size	1
Training precision	BF16
Training epochs	60–200 (task-dependent)
Video noise-time shift	5.0
Action noise-time shift	1.0
Video and action loss weights	1.0 each

Training objective. The branches independently sample noise times σ and Gaussian noise ϵ, using input $y_\sigma = (1-\sigma)y + \sigma\epsilon$ and target $\epsilon - y$ with flow-scheduler-weighted MSE. Video loss applies only to future targets. History augmentation independently samples $\alpha \sim \mathcal{U}(0,1)$ per latent frame and forms $\widetilde{h} = (1-\alpha)h + \alpha\eta$, with standard Gaussian η. The mixing ratio is shared within each frame; history-branch temporal modulation remains zero.

Memory construction and gradient flow. A training sample corresponds to one planning decision and loads the cached causal-VAE history through that decision. Precomputed segment manifests supply only segments whose confirmation times have arrived. For each forward pass, the model packs the historical source tokens, a shared-initialization slot block for each confirmed segment, the retained observations, and the current video/action targets. Historical Transformer features and segment memories are recomputed with the current parameters; inference-time K/V caches are not carried between training samples. Source tokens remain in the training computation to construct memory, but the target attention mask exposes only the anchor, confirmed memories, and recent/open observations. Older sources outside this retained context cannot be read directly by the targets.

The packed forward pass is differentiable across segment boundaries: memory states and source K/V are not detached, and gradients from the prediction losses can propagate through memory reads, writes, and earlier-memory dependencies within the sampled history. There is no segment-wise truncated backpropagation or persistent computation graph between separate samples. Activation checkpointing recomputes intermediate activations while preserving these gradient paths. We jointly optimize the video and action Transformer experts, the shared learnable memory slots, and the proprioceptive projection. The VAE remains frozen, and instruction embeddings are precomputed and fixed. Grouping manifests are not differentiated. The supplied training configuration disables the optional closed-segment K/V distillation loss, so supervision uses the video and action objectives above. Historical-conditioning noise is applied to all historical latent frames before source encoding and memory construction, including the anchor and retained observations; it therefore also affects the sources from which memory is written.

Evaluation and ablations. All variants in Section 4.3 are independently retrained with the full model's task-specific data and training/evaluation settings, except for the ablated component. In w/o

Segment Memory, only compressed segment memories are removed. The initial anchor, temporal grouping rule, and deduplicated recent/open observation set $\mathcal{F}_t = \{0\} \cup \mathcal{R}_t \cup \mathcal{U}_t$ are retained as in the full model, including open-segment observations outside the recent window. Evaluation uses 10 action-denoising steps and 100 trials per task.

Task-specific pretraining. Only Observe and Pick Up uses additional RoboTwin pretraining (Mu et al., 2025) to reduce the influence of basic grasping limitations on memory-dependent performance. This introduces a training-data difference from the reported baselines; it does not establish a matched-data comparison.

Cache measurement. We allocate synthesized K/V layouts in BF16 at batch size 1 across 30 layers, each with 24 heads of dimension 128. For N historical positions per layer, storage in MiB is

$$C(N) = \frac{2 \times 30 \times N \times 3072 \times 2}{2^{20}},$$

where the factors of 2 count keys/values and bytes per element. Segment layouts use $N = 120|\mathcal{F}_t| + 8J_t$, with deduplication as in Equation 3. For the same closed history, segment memory uses $J_t / \sum_j L_j = 1/\bar{L}_t$ times the positions of per-frame memory, where $\bar{L}_t$ is mean segment length; retained source storage is additional.

Segment lengths and confirmation delays come from 387 Battery Try training segments, shuffled as pairs with seed 42 and repeated cyclically for longer histories. All layouts are synthesized; 56 latent frames marks the dataset maximum. At 1600 frames, dense history requires 1600×120 positions (67500 MiB analytically) and runs out of memory on an 80GB A800. Measurements exclude model weights and temporary runtime tensors.

C.2 Real-World Task Protocol

Under the setup in Section 4.4, all methods use the same 50 demonstrations per task and 20 evaluation trials. Success requires completing every stage below, including the specified order, object choice, count, and confirmation; the overall rate averages the three task success rates.

- **Sequential Uncovering.** Cover the three objects from left to right using the cups, then uncover the doll, the stamp, and the small bottle in that order.
- **Hidden-Object Matching.** Lift the cup and observe the object hidden underneath it. Put the cup back to cover the object. Then pick up the object on the table that matches the previously revealed object.
- **Press Button.** Press the button as many times as indicated by the number in front of you, then press the confirmation button.