LTM: LATENT TRANSITION BASED IN-CONTEXT MEMORY FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Robotic manipulation requires remembering interactions that are no longer observable in the current scene. Existing memory mechanisms face a trade-off between retaining detailed histories and maintaining a compact policy context. We introduce **Latent Transition Memory (LTM)**, which represents interaction history as an ordered sequence of compact latent transitions between consecutive observations. A frozen latent action encoder extracts each transition, and a lightweight adapter maps it to a single memory token, preserving a record of observed changes at one token per transition. We instantiate LTM on $\pi_{0.5}$ as **LTM-π**, using causal attention and persistent KV caching for incremental inference, with sparse visual anchors to ground video demonstrations. Across two pretrained latent encoders, LTM-π achieves up to 58.54% success on RoboMME and 80.5% on LIBERO-Mem, exceeding the reported memory-free $\pi_{0.5}$ baseline by 40.61 percentage points and the strongest compared LIBERO-Mem baseline by 38.0 points, respectively. On a real robot, it improves average success rate from 11.25% to 82.50% in a *pick-and-place x times* task. Ablations support the contribution of transition content beyond temporal cues, while a controlled swap diagnostic shows that relevant motion information can remain recoverable even when downstream state tracking fails. These findings establish latent transitions as an effective, compact memory representation and highlight temporal composition as a remaining challenge.

1 INTRODUCTION

Vision-language-action (VLA) models combine pretrained visual and linguistic representations with action generation, enabling robots to follow diverse instructions and generalize across manipulation settings (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; Physical Intelligence et al., 2025). Yet successful manipulation often depends on information that is absent from the current observation. A robot asked to place three objects into a bag must remember how many transfers it has completed; a robot tracking an occluded object must retain the effects of earlier interactions. Similar current scenes can therefore require different actions depending on the preceding history. Addressing this partial observability requires a policy to retain and use past experience, motivating explicit memory mechanisms and benchmarks for history-dependent robotic control (Dai et al., 2026; Chung et al., 2026b).

Existing memory designs retain sampled visual observations, summarize experience in language, or update a compact recurrent state (Shi et al., 2025; Qu et al., 2026; Torne et al., 2026; Cherepanov et al., 2026). These choices expose different trade-offs. Dense visual histories consume substantial context, whereas sparse sampling can omit intermediate interactions. Language summaries provide semantic abstraction, but remain limited in complex environments, fine-grained interactions, and manipulation processes that are difficult to faithfully express in natural language. Recurrent representations support compact updates, but must continually decide which details to preserve in a bounded state. Beyond how memory is stored or retrieved, these trade-offs raise a basic representational question: *what should a VLA remember about its past interactions?*

For many manipulation tasks, the relevant history is the sequence of changes induced by robot–environment interactions. Repeated views contain substantial appearance redundancy, while the differences between consecutive observations describe how an episode unfolds. This motivates rep-

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.

resenting history through *latent transitions*. Latent action models learn such representations by encoding an observation pair into a compact code from which the later observation can be reconstructed conditioned on the earlier one (Bruce et al., 2024; Ye et al., 2024). This conditional reconstruction objective encourages the bottleneck to encode temporal changes. While latent actions have been used for policy pretraining and world-model control (Ye et al., 2024; Bu et al., 2025; Gao et al., 2026a), we study their use as an ordered episodic memory of observed interactions for subsequent action generation.

We introduce **Latent Transition Memory (LTM)**, which encodes each pair of consecutive observations with a frozen latent action model and projects the resulting code into a single memory token. The policy receives a delayed transition prefix alongside the instruction and current observation. This design retains a compact record up to a configurable cutoff, without requiring historical image tokens at every timestep or repeatedly compressing the entire episode into a recurrent state. Each code depends only on an adjacent observation pair, allowing transition encoding to overlap action execution before the next decision. We instantiate this approach on $\pi_{0.5}$ as **LTM-π**. A lightweight adapter maps latent transitions into the VLA backbone, and causal attention over the language–memory prefix enables persistent KV caching during online inference. For video-conditioned tasks, sparse initial-frame anchors complement transition memory with visual grounding across demonstration and execution scenes. The policy and adapter are trained from robot demonstrations without language summaries or intermediate semantic supervision.

We evaluate LTM-π with two frozen latent encoders, CD-LAM and DreamDojo, on RoboMME, LIBERO-Mem, and a real-world counting task. The DreamDojo variant achieves 58.54% average success on RoboMME, compared with 17.93% for the reported memory-free $\pi_{0.5}$ baseline, and 80.5% on LIBERO-Mem, compared with 42.5% for the strongest baseline in our comparison. In terms of inference efficiency, the average latency of a single LTM-π policy inference is only 67 ms. On a Galbot G1 robot, LTM-π increases average success rate from 11.25% to 82.50%. Ablations show that the ordered memory sequence itself provides useful temporal-progress cues, while observation-dependent latent transitions provide additional gains and visual anchors offer complementary grounding. Replacing transition codes with fixed random representations reduces average success from 57.67% to 44.50%, supporting the value of transition-specific content beyond timing cues.

Our results also reveal a limitation: preserving local transitions does not ensure that a policy can compose them into a persistent estimate of object state. LTM-π struggles on tasks involving repeated object swaps. In a controlled Cup Swap diagnostic, however, a process-supervised model recovers the final state from CD-LAM transitions with 97.93% accuracy. This demonstrates that the latent sequence retains information needed for swap tracking in this setting, while pointing to downstream temporal composition as a remaining challenge.

Our contributions are threefold:

- We introduce LTM, an episodic memory representation that encodes interaction history as one compact token per observed transition.
- We develop LTM-π with lightweight memory adaptation, cache-compatible causal attention, and delayed memory for efficient online inference.
- We demonstrate its effectiveness and efficiency across two simulation benchmarks and real-world manipulation, and use ablations and a controlled diagnostic to distinguish the benefits of transition content from temporal cues and the challenges of state tracking.

2 Related Work

Vision-language-action models. Vision-language-action (VLA) models adapt pretrained vision-language representations to predict robot actions from visual observations and language instructions. RT-2 (Brohan et al., 2023) tokenizes actions to transfer web-scale semantic knowledge to robotic control, while OpenVLA (Kim et al., 2024) provides an open-source generalist manipulation model. π_0 (Black et al., 2024) further introduces flow matching for continuous action generation, with subsequent models improving generalization through heterogeneous co-training and hierarchical prediction in $\pi_{0.5}$ (Physical Intelligence et al., 2025) and large-scale multi-embodiment training in LingBot-VLA (Wu et al., 2026). Despite these advances, standard VLAs remain limited under

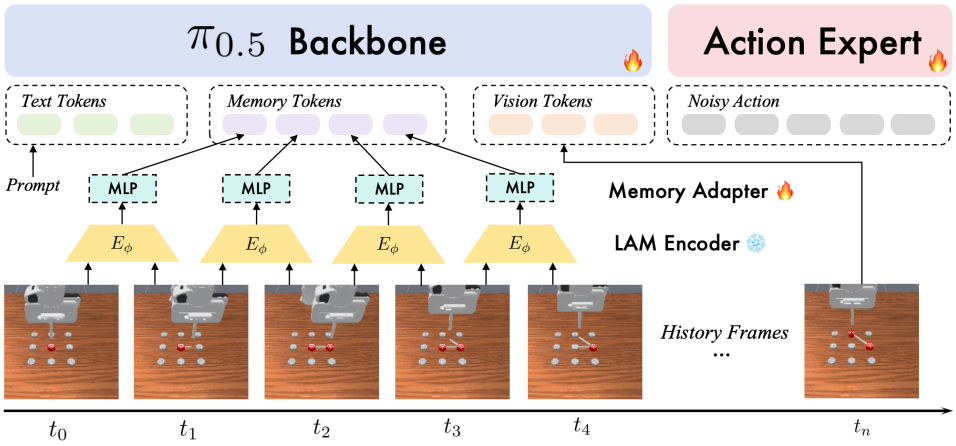


Figure 1: Our method LTM-π.

partial observability, where identical current observations may require different actions depending on past events, motivating explicit execution memory (Dai et al., 2026).

Memory mechanisms for VLAs. Existing memory-augmented VLAs differ mainly in how historical information is represented and injected into the policy. RoboFlamingo (Li et al., 2023) models history through a sequential policy head, while HAMLET (Koo et al., 2025) and μVLA (Cherepanov et al., 2026) compress history into learned memory tokens. MemoryVLA (Shi et al., 2025) and LaMem-VLA (Qu et al., 2026) maintain explicit memory banks and retrieve relevant perceptual or semantic evidence. Other methods exploit temporal hierarchy: MEM (Torne et al., 2026) combines recent video with long-term textual memory, HiMe (Ji et al., 2026) separates working memory from long-term planning, and WeaveLA (Zhu et al., 2026) stores compact segment representations at subtask boundaries. HiF-VLA (Lin et al., 2025) further incorporates historical motion priors, while MemoryVLA++ (Shi et al., 2026) combines retrieved history with latent future imagination. RoboMME (Dai et al., 2026) shows that different memory designs benefit different task types, motivating representations that preserve task-relevant interaction dynamics rather than merely increasing memory capacity.

Latent action models. Latent action models (LAMs) learn compact transition representations from videos without action labels, typically by inferring a latent action between observations and reconstructing the resulting transition (Bruce et al., 2024; Ye et al., 2024). Genie (Bruce et al., 2024) learns discrete latent actions for controllable video environments, while LAPA (Ye et al., 2024) uses them as supervision for VLA pretraining. Later work extends this idea to continuous control and transfer: CLAM (Liang et al., 2025) learns continuous latent actions, UniVLA (Bu et al., 2025) learns task-centric latent actions for cross-embodiment policy learning, and DreamDojo (Gao et al., 2026a) uses continuous latent actions as proxy controls for world-model pretraining. In contrast, we use latent actions inferred from historical observation pairs as episodic memory, providing the VLA with a compact representation of past interactions for subsequent action generation.

3 Method

3.1 Latent-Transition-Based In-Context Memory for VLAs

We represent interaction history using compact latent transitions between consecutive observations. A latent action model (LAM) encodes each transition from o_{i-1} to o_i into a latent code z_i, from which a decoder reconstructs o_i conditioned on o_{i-1}:

$$z_i = E_\phi(o_{i-1}, o_i), \qquad \hat{o}_i = D_\psi(o_{i-1}, z_i), \qquad \mathcal{L}_{\text{LAM}} = d(\hat{o}_i, o_i),$$

where E_ϕ and D_ψ are the transition encoder and decoder, respectively, and d is a reconstruction distance. Conditioning the decoder on the preceding observation encourages the compact latent bottleneck to capture temporal changes rather than redundant appearance information.

Two properties make these representations useful for VLA memory: **(i) Compactness.** Each transition is represented by a low-dimensional vector, reducing the context budget required to retain interaction history. **(ii) Locality.** Each code depends only on two consecutive observations, allowing incremental memory construction without re-encoding earlier frames.

Let $\mathbf{A}_t$ denote an action chunk, s_t the robot state, and ℓ the language instruction. A current-observation policy predicts $\mathbf{A}_t \sim \pi_\theta(\cdot \mid o_t, s_t, \ell)$. Existing memory mechanisms augment this policy in different ways: frame-based memory uses sampled observations, $\mathbf{A}_t \sim \pi_\theta(\cdot \mid o_t, s_t, \ell, \{o_i\}_{i\in\mathcal{I}_t})$, where $\mathcal{I}_t \subseteq \{0, \dots, t-1\}$; language-based memory uses a summary $m_t^{\text{lang}} = f_{\text{LLM}}(o_{<t})$, giving $\mathbf{A}_t \sim \pi_\theta(\cdot \mid o_t, s_t, \ell, m_t^{\text{lang}})$; and recurrent memory updates $h_t = f_{\text{rec}}(h_{t-1}, o_t)$, giving $\mathbf{A}_t \sim \pi_\theta(\cdot \mid o_t, s_t, \ell, h_t)$.

Our approach instead conditions the policy on an ordered sequence of latent transitions:

$$\mathcal{M}_t^{\text{LAM}} = (z_i)_{1\le i\le \tau_t}, \qquad \mathbf{A}_t \sim \pi_\theta(\cdot \mid o_t, s_t, \ell, \mathcal{M}_t^{\text{LAM}}).$$

Here, $\tau_t = \max(0, t-k)$ excludes the latest k transitions, while o_t remains the current visual input. This representation retains transition-level information without historical RGB tokens or recurrent compression.

3.2 LTM-π

We instantiate this memory mechanism on $\pi_{0.5}$ and refer to the resulting model as **LTM-$\boldsymbol{\pi}$**. The policy and memory projector are trained on demonstration data using a frozen LAM encoder, without requiring language summaries or intermediate semantic supervision. The architecture of **LTM-$\boldsymbol{\pi}$** is shown in Figure 1.

Training Phase. For each training sample, we use a fixed lag k, shared with inference, and set $\tau_t = \max(0, t-k)$. Frozen DreamDojo (Gao et al., 2026a) or CD-LAM (Wei et al., 2026) encodes the retained observation pairs into 32-dimensional transition vectors:

$$z_i = E_\phi(o_{i-1}, o_i) \in \mathbb{R}^{32}, \qquad \tilde{z}_i = \texttt{stopgrad}\left(\frac{z_i - \mu_{\mathcal{D}}}{\sigma_{\mathcal{D}}}\right), \quad 1 \le i \le \tau_t,$$

where E_ϕ denotes the LAM encoder, and $\mu_{\mathcal{D}}$ and $\sigma_{\mathcal{D}}$ are dataset-level normalization statistics.

We use a lightweight memory adapter to align latent actions with the semantic space of the VLM. A two-layer MLP projector maps each normalized transition into a memory token in the 2048-dimensional hidden space of the PaliGemma-3B backbone of $\pi_{0.5}$:

$$m_i = \text{LN}(W_2\,\text{GELU}(W_1\tilde{z}_i + b_1) + b_2), \qquad W_1 \in \mathbb{R}^{512\times 32}, \quad W_2 \in \mathbb{R}^{2048\times 512}.$$

We insert these tokens between the language and current visual tokens:

$$X_t = \left[X^{\text{lang}}(\ell); m_1, \dots, m_{\tau_t}; X^{\text{vis}}(o_t)\right].$$

The resulting context conditions the action expert. We apply causal attention within the language–memory prefix and allow the current visual tokens to attend to this prefix. Each prefix token therefore depends only on itself and preceding prefix tokens, enabling its KV representations to be reused when new memory tokens and visual inputs arrive.

Inference Phase. Algorithm 1 in Appendix A summarizes online inference. At decision step t, the policy consumes transitions only through $\tau_t = \max(0, t-k)$. Each transition is encoded once, asynchronously after its observation pair becomes available, and stored in a latent buffer; the lag delays only when it is admitted to policy memory. Thus, a transition has up to k observation steps of computation slack before it is required. At each decision, newly eligible latents are appended to the persistent language–memory KV cache, after which o_t is processed and an action chunk is generated. This overlaps LAM computation with action execution and reduces critical-path blocking.

For n cached prefix tokens and b newly processed memory and visual tokens, caching reduces the per-chunk backbone self-attention cost from $\mathcal{O}((n+b)^2)$ to $\mathcal{O}(nb + b^2)$. For fixed b, this cost is linear in the cached history length.

Visual Anchors for Video Demonstrations. For tasks conditioned on video demonstrations, the initial scene layout may differ between the demonstration and execution environments. To improve visual–semantic alignment, we complement latent transitions with sparse visual features from both sequences, inspired by frame-based memory sampling. A minimal choice uses the first frame of each sequence as a visual anchor:

$$X_t = \big[X^{\text{lang}}(\ell); X^{\text{vis}}(o_0^{\text{demo}}); m_1^{\text{demo}}, \ldots, m_{T_{\text{demo}}}^{\text{demo}}; X^{\text{vis}}(o_0); m_1, \ldots, m_{\tau_t}; X^{\text{vis}}(o_t)\big].$$

Here, o_0^{demo} and o_0 are the initial demonstration and execution observations, respectively, and m_i^{demo} encodes the transition from o_{i-1}^{demo} to o_i^{demo}. We add distinct learnable type embeddings to the demonstration and execution segments to distinguish their roles; these embeddings are omitted from the formula for clarity. The fixed visual anchors and demonstration tokens are retained in the persistent KV cache alongside the language and execution memory tokens.

4 Experiment

4.1 Evaluation in Simulation

4.1.1 Settings

Benchmarks. We evaluate memory-dependent robotic control on two complementary benchmarks, RoboMME (Dai et al., 2026) and LIBERO-Mem (Chung et al., 2026b). RoboMME contains 16 history-dependent manipulation tasks organized into four suites: Counting, Permanence, Reference, and Imitation, targeting temporal, spatial, object, and procedural memory, respectively. These tasks require the policy to retain information from earlier observations or demonstrations that may no longer be recoverable from the current observation. LIBERO-Mem further evaluates long-horizon memory under object-level partial observability. It contains 10 tasks spanning four object-centric memory dimensions: Object Motion (OM), Object Sequence (OS), Object Relations (OR), and Object Occlusion (OO). The benchmark introduces temporally dependent subgoals and visually ambiguous objects, requiring the policy to track object identities, interactions, and state changes over trajectories of up to several hundred frames.

Training and Evaluation Configuration. We train the model on 8 H100 GPUs for 40K steps on RoboMME and 30K steps on LIBERO-Mem, with a batch size of 128. The learning rate decays from 2.5×10^{-5} to 2.5×10^{-6}, with 1K warm-up steps. Unless otherwise specified in the lag ablations, we use the same fixed lag $k = 5$ during training and inference on both benchmarks. The policy predicts action chunks of 50 steps on RoboMME and 10 steps on LIBERO-Mem, and replans after executing the first 10 and 5 steps, respectively. We do not use proprioceptive inputs on RoboMME. Following the official evaluation protocols, we evaluate each task on 20 layouts in LIBERO-Mem and over 50 rollouts in RoboMME. For RoboMME, we evaluate with three random seeds and report the mean success rates for the main results, while the ablations are evaluated with a single seed.

Baselines. For RoboMME, we compare against DP-Unet, OpenVLA-OFT, $\pi_{0.5}$ with and without past actions, and its symbolic, perceptual, and recurrent memory variants (Dai et al., 2026). We also include MemoryVLA, SAM2Act+, and MemER as reported in RoboMME, alongside WeaveLA (Zhu et al., 2026) and PonderPounce (9B, $1\times$ data) (Choi et al., 2026). All baseline results are taken directly from published reports, with oracle-assisted methods and human performance listed separately for reference. For LIBERO-Mem, we compare against $\pi_{0.5}$, SlotVLA, and Naive E-SlotSSM (Chung et al., 2026a), MemoryVLA (Shi et al., 2025), GMP (Gao et al., 2026b), MemoryVAM with UNet and DiT backbones (Jiang et al., 2026), and 2AM together with its π_0 reimplementation (Hu et al., 2026).

4.1.2 Main Results

Results on RoboMME. As shown in Table 1, LTM-π achieves average success rates of 57.67% with CD-LAM and 58.54% with DreamDojo, improving over $\pi_{0.5}$ by 39.74 and 40.61 percentage points, respectively. Both variants also outperform FrameSamp+Modul (44.51%), suggesting that compact transition representations provide an effective alternative to sampled-frame memory. The DreamDojo variant is particularly strong on Counting, achieving 90.33% on average and solving

Table 1: Success rates (%, ↑) on all 16 RoboMME tasks. Avg. denotes the overall success rate. Privileged-information results are listed separately.

Method	Counting				Permanence				Reference				Imitation				Avg.
	BF	PX	SX	SC	VU	BU	VUS	BUS	PH	VR	VPB	VPO	MC	IP	PL	RS	
Privileged-information references																	
SimpleSG+Oracle	85.78	99.78	100.00	44.67	33.11	22.00	15.56	15.56	44.00	27.78	31.33	26.00	87.33	10.00	95.33	55.11	49.58
GroundSG+Oracle	85.78	100.00	100.00	49.67	98.78	95.00	99.22	80.22	83.33	97.33	100.00	100.00	87.78	15.56	97.00	55.56	84.08
WeaveLA (oracle)	28.00	34.00	50.00	12.00	24.00	28.00	22.00	4.00	12.00	2.00	22.00	14.00	16.00	2.00	8.00	12.00	18.12
Human	96.00	100.00	80.00	78.00	90.00	92.00	92.00	90.00	92.00	92.00	98.00	90.00	90.00	98.00	84.00	86.00	90.50
Without long-term visual memory																	
DP-Unet	32.00	2.67	6.67	0.00	16.00	4.00	4.00	0.00	0.00	0.00	22.00	5.67	8.67	0.00	2.00	10.00	7.11
OpenVLA-OFT	8.00	8.00	2.00	2.00	16.00	0.00	18.00	0.00	10.00	0.00	28.00	20.00	18.00	4.00	6.00	8.00	9.25
$\pi_{0.5}$	30.00	42.89	35.56	6.67	20.44	22.22	18.67	6.67	11.33	0.44	31.11	25.78	26.00	1.56	2.89	4.67	17.93
$\pi_{0.5}$ w/ past actions	26.67	58.33	26.67	4.67	30.67	23.67	20.67	16.00	12.33	8.67	24.00	18.67	34.00	1.00	4.00	5.67	19.73
Symbolic memory on $\pi_{0.5}$																	
SimpleSG+Gemini	46.00	63.00	45.00	2.00	29.00	9.00	14.00	2.00	20.00	15.00	26.00	29.00	61.00	4.00	7.00	0.00	23.25
SimpleSG+QwenVL	**77.56**	95.33	5.11	0.44	34.22	19.33	15.33	9.56	17.11	25.33	33.33	25.11	82.00	3.78	12.67	7.78	29.00
GroundSG+Gemini	26.00	18.00	4.00	3.00	36.00	14.00	13.00	0.00	9.00	17.00	12.00	7.00	17.00	0.00	7.00	2.00	11.56
GroundSG+QwenVL	52.00	92.67	7.33	0.00	88.67	24.00	30.67	14.00	15.11	25.33	54.00	31.78	71.56	3.33	6.67	6.00	32.70
Perceptual memory on $\pi_{0.5}$																	
TokenDrop+Context	48.67	85.11	94.67	3.11	33.78	31.56	26.22	16.00	20.67	17.78	31.11	25.33	81.33	4.00	12.67	20.00	34.50
TokenDrop+Modul	34.44	83.56	86.00	5.33	28.22	29.33	28.44	21.33	21.33	22.00	59.56	36.00	62.00	7.11	32.44	51.56	38.04
TokenDrop+Expert	54.22	87.56	91.78	4.22	26.67	30.44	18.44	18.89	19.33	20.89	36.44	24.67	**87.56**	2.22	16.22	18.22	34.86
FrameSamp+Context	41.22	72.00	73.67	13.67	26.89	30.22	20.89	15.22	17.67	15.22	30.00	20.89	77.22	1.22	15.22	19.67	30.68
FrameSamp+Modul	39.56	87.33	92.00	42.00	32.67	25.11	24.44	18.22	22.89	30.44	60.00	32.00	77.78	7.56	53.56	66.67	44.51
FrameSamp+Expert	57.33	86.22	94.67	28.89	31.78	25.78	22.89	20.22	19.11	23.11	30.00	24.22	83.11	2.00	13.56	17.11	36.25
Recurrent memory on $\pi_{0.5}$																	
TTT+Context	35.56	62.89	42.44	3.33	29.78	22.89	18.44	14.44	20.44	13.11	34.22	20.22	32.44	1.11	1.56	3.56	22.28
TTT+Modul	34.22	65.11	36.67	2.11	27.22	22.11	25.22	14.11	14.56	12.11	32.67	22.33	31.22	1.11	3.56	7.00	21.96
TTT+Expert	34.89	63.78	41.33	4.00	31.78	22.44	19.56	18.00	12.22	9.56	34.00	22.89	33.56	0.89	3.11	5.56	22.35
RMT+Context	32.44	56.89	33.56	5.78	31.33	10.89	17.33	2.00	14.00	3.78	32.00	29.11	25.78	2.00	5.56	8.89	19.46
RMT+Modul	33.33	60.78	37.78	4.67	31.11	11.78	17.78	2.44	17.11	4.22	32.00	31.11	24.67	2.21	3.78	8.00	20.17
RMT+Expert	35.78	60.22	36.00	5.56	28.00	17.11	15.78	2.00	11.78	0.22	24.22	22.67	20.00	1.89	4.22	4.89	18.15
Other memory policies																	
MemoryVLA	10.00	17.33	1.33	0.00	16.00	8.00	4.67	4.67	7.33	0.00	13.33	4.00	14.00	0.00	12.00	1.33	7.12
SAM2Act+	40.00	76.00	25.33	0.00	27.33	32.00	18.00	26.67	17.33	5.33	24.67	20.00	29.33	0.00	0.00	0.00	21.37
MemER	56.67	79.33	59.33	0.00	81.33	72.00	**38.00**	21.33	70.67	25.33	30.00	26.00	82.67	6.67	16.67	12.00	42.38
WeaveLA	26.00	44.00	46.00	2.00	28.00	14.00	14.00	4.00	10.00	4.00	26.00	18.00	22.00	4.00	8.00	8.00	17.38
PonderPounce (9B)	50.67	93.33	72.00	82.67	**94.67**	97.33	25.33	34.00	**85.33**	40.67	**83.33**	**79.33**	81.33	6.00	12.67	34.67	**60.83**
LTM-π (Ours)																	
CD-LAM	64.67	**100.00**	93.33	87.33	68.67	**98.00**	30.67	31.33	22.67	30.00	38.67	22.67	80.00	**20.67**	**64.00**	**70.00**	57.67
DreamDojo	66.00	99.33	**96.00**	**100.00**	72.67	86.67	34.00	**36.00**	38.67	**42.67**	38.00	30.67	77.33	20.00	60.00	38.67	58.54

BF: BinFill; PX: PickXtimes; SX: SwingXtimes; SC: StopCube; VU/BU: Video/ButtonUnmask; VUS/BUS: Video/ButtonUnmaskSwap; PH: PickHighlight; VR: VideoRepick; VPB/VPO: VideoPlaceButton/Order; MC: MoveCube; IP: InsertPeg; PL: PatternLock; RS: RouteStick.

StopCube with 100% success. The CD-LAM variant performs best on InsertPeg, PatternLock, and RouteStick among non-oracle policies, highlighting its utility for retaining procedural information. However, LTM-π remains slightly below PonderPounce (60.83%) overall, with substantial gaps on several Reference tasks. These results suggest that latent transitions effectively capture interaction history, while preserving fine-grained object identity and visual correspondence remains a limitation. In terms of inference efficiency, the average latency of a LTM-π policy inference is only 66.71 ms.

Table 2: Success rates (SR, %) on LIBERO-Mem. Bold indicates the best result in each column.

Method	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	Avg.
$\pi_{0.5}$ (original)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SlotVLA ($h = 8$)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Naive E-SlotSSM (oracle)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
MemoryVLA	55.0	0.0	35.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	9.5
GMP	85.0	45.0	20.0	5.0	40.0	0.0	0.0	0.0	0.0	0.0	19.5
MemoryVAM (Pathway 1 only)	85.0	40.0	35.0	10.0	15.0	0.0	5.0	0.0	30.0	65.0	28.5
MemoryVAM (Pathway 2 only)	45.0	10.0	5.0	0.0	5.0	5.0	0.0	0.0	5.0	0.0	7.5
MemoryVAM (UNet)	85.0	65.0	90.0	10.0	35.0	0.0	25.0	5.0	40.0	70.0	42.5
MemoryVAM (DiT)	75.0	65.0	65.0	15.0	55.0	35.0	10.0	10.0	25.0	50.0	40.5
π_0 (2AM reimplementation)†	30.83	13.33	14.17	21.67	11.67	10.00	6.67	8.33	0.00	5.83	12.25
2AM†	1.67	5.00	5.00	6.67	6.67	2.50	7.50	10.00	47.50	25.83	11.83
LTM-π (Ours)											
CD-LAM	**95.0**	**75.0**	90.0	70.0	70.0	45.0	**80.0**	20.0	**100.0**	**100.0**	74.5
DreamDojo	**95.0**	**75.0**	**100.0**	**90.0**	**90.0**	**65.0**	65.0	**25.0**	**100.0**	**100.0**	**80.5**

T0: Bowl back on plate; T1: Bottle back on plate; T2: Bowl $\times 3$; T3: Bottle $\times 3$; T4: Bowl $\times 5$; T5: Bowl $\times 7$; T6: Swap 2 bowls; T7: Rotate 3 bowls; T8: Cheese + basket center; T9: Cheese + empty basket center.

† Their π_0 reimplementation uses Qwen3-VL-4B; the number of rollouts is unspecified and does not follow the official 20-rollout setting.

Results on LIBERO-Mem. As shown in Table 2, LTM-π achieves average success rates of 74.5% with CD-LAM and 80.5% with DreamDojo, exceeding the strongest reported baseline, Memory-VAM (UNet), by 32.0 and 38.0 percentage points, respectively. DreamDojo improves over CD-LAM by 6.0 points overall, with particularly large gains on repeated-interaction tasks T3–T5, while both variants achieve 100% success on T8 and T9. These results support the effectiveness of latent-transition memory for history-dependent manipulation across different LAM encoders. The encoder choice nevertheless affects individual tasks: CD-LAM performs better on swapping two bowls (T6), achieving 80.0% versus 65.0%. Rotating three bowls (T7) remains challenging for both variants (20.0% and 25.0%), suggesting that tracking multiple objects through sequential interactions remains a limitation.

4.1.3 Ablations

Effect of Latent Transitions and Visual Anchors. In Table 3, we ablate the two sources of historical information in LTM-π: latent-transition memory and sparse visual anchors. We compare the full model (CD-LAM version) with **LT-only**, which removes visual anchors; **Anchor-only**, which removes latent transitions; and **Fixed Random LT**, which replaces observation-dependent latent transitions with fixed random representations while preserving the temporal structure of the memory sequence.

Table 3: Ablation study on the effect of latent transitions and visual anchors on RoboMME. Success rates (%, ↑) are reported on all 16 tasks. Fixed Random LT removes transition-specific semantics while retaining temporal information from the memory sequence.

Method	Counting				Permanence				Reference				Imitation				Avg.
	BF	PX	SX	SC	VU	BU	VUS	BUS	PH	VR	VPB	VPO	MC	IP	PL	RS	
Full	64.67	100.00	93.33	87.33	68.67	98.00	30.67	31.33	22.67	30.00	38.67	22.67	80.00	20.67	64.00	70.00	**57.67**
LT-only	66.00	100.00	94.00	90.00	24.00	44.00	30.00	16.00	26.00	30.00	40.00	36.00	90.00	14.00	46.00	40.00	49.13
Anchor-only	34.00	32.00	42.00	0.00	66.00	58.00	24.00	8.00	12.00	2.00	26.00	10.00	32.00	2.00	12.00	10.00	23.13
Fixed Random LT	72.00	74.00	98.00	58.00	82.00	94.00	30.00	38.00	18.00	26.00	26.00	20.00	60.00	2.00	10.00	4.00	44.50

Removing latent transitions reduces average success from 57.67% to 23.13%, while LT-only retains 49.13%, showing that the transition sequence is the stronger historical component and anchors mainly provide complementary visual grounding.

Fixed Random LT still reaches 44.50%, indicating that the ordered memory sequence itself provides useful cues about elapsed time and trajectory progress. Replacing these random tokens with observation-dependent latent transitions further improves performance to 57.67%, demonstrating that transition-specific content contributes beyond temporal cues. This gain is limited on swap-dependent tasks, motivating the diagnostic study below.

Table 4: Ablation study on lag k on RoboMME. Success rates (%, ↑) are reported on all 16 tasks. Reported LAM blocking is an estimated critical-path cost, $\max(0, T_{\text{LAM}} - T_{\text{overlap}})$.

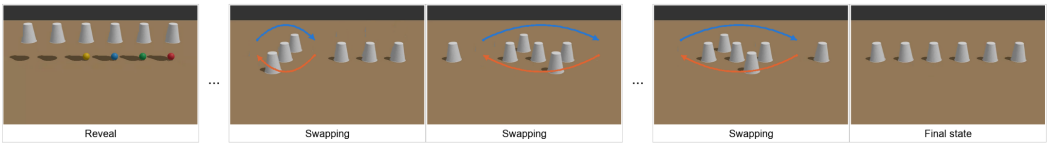
k	Counting				Permanence				Reference				Imitation				Avg.	Runtime (ms, ↓)			
	BF	PX	SX	SC	VU	BU	VUS	BUS	PH	VR	VPB	VPO	MC	IP	PL	RS		Policy	Sim.	LAM update	LAM block
0	62.00	98.00	94.00	90.00	70.00	94.00	30.00	34.00	28.00	26.00	38.00	24.00	82.00	20.00	66.00	70.00	57.88	66.71	117.01	114.25 (10p, sync.)	114.25
5	64.67	100.00	93.33	87.33	68.67	98.00	30.67	31.33	22.67	30.00	38.67	22.67	80.00	20.67	64.00	70.00	57.67	66.71	117.01	67.97 (5p, async.)	9.47
10	58.00	98.00	90.00	86.00	68.00	96.00	28.00	32.00	20.00	30.00	36.00	24.00	76.00	24.00	64.00	72.00	56.38	66.71	117.01	114.25 (10p, async.)	0.00

Effect of Lag k. On RoboMME, we replan every 10 execution steps and use the same k for training and inference. As shown in Table 4, lag creates computation slack between observing a transition and admitting it to policy memory. Increasing k from 0 to 5 changes average success only from 57.88% to 57.67%, while reducing the estimated LAM blocking cost from 114.25 ms to 9.47 ms. At $k = 10$, the measured LAM update fits within the available execution overlap, giving an estimated zero blocking cost, with 56.38% success. Thus, a small non-zero lag substantially reduces critical-path overhead with little performance loss.

4.1.4 Diagnostic Study: Can Latent Transitions Represent "Swap"?

Problem. The RoboMME results reveal a clear weakness on swap-dependent tasks. For example, both LTM-π variants perform substantially better on Video/ButtonUnmask than on their swap coun-

terparts (Table 1), and our qualitative inspection of VideoRepick failures shows that many errors also occur when the demonstration contains swap operations. This raises a more specific question: *do latent transitions fail to encode swap motion, or does the downstream model fail to compose the encoded transitions into a persistent object state?*



Input: There are 6 cups numbered 1 through 6. Initially, balls are under cups 2, 3, 4 and 5. The latent actions encode only motion after that reveal. <LT_0><LT_1><LT_1> ... <LT_N>

(a) **Vanilla supervision** (final state only)

Output: **0, 2, 3, 4**

(b) **Process supervision** (CoT-like)

Output: NO.1: [SWAP(0, 2), CUP(0, 3, 4, 5)]
NO.2: [SWAP(1, 5), CUP(0, 1, 3, 4)] ...
FINAL: CUP(1, 2, 3, 6)

Figure 2: Our Cup Swap diagnostic benchmark and two supervision methods. The arrows are for illustration only and are not included in the actual dataset. <LT_i> denotes the i-th latent transition.

Cup Swap Test. To isolate these two factors, we construct a controlled Cup Swap diagnostic benchmark as shown in Figure 2. Each sequence contains six cups and one to five hidden balls, followed by 5–20 randomly generated cup swaps. The initial ball locations are provided explicitly, while only the post-reveal motion history is represented by CD-LAM latent transitions. We inject these transitions into a Qwen3-VL-2B backbone and compare two supervision schemes: **Vanilla**, which directly predicts the final cup occupancy without intermediate supervision, and **Process supervision (CoT-like)**, which explicitly supervises swap tracking and the resulting intermediate/final state. Full experimental details are provided in Appendix B.

As shown in Table 5, the process-supervised model recognizes individual swaps with 99.78% accuracy and recovers the final state with 97.93% accuracy, despite using a frozen VLM backbone in this setting. This provides strong evidence that the **CD-LAM latent sequence preserves the motion and temporal-order information required to track swaps**. In contrast, directly decoding the final state without process supervision reaches only 2.40% best exact accuracy on the same six-cup, multi-ball task. The gap therefore suggests that the dominant difficulty is not simply the absence of swap information in the latent transitions, but the downstream model's ability to *compose* many local transitions into a persistent state estimate without **intermediate supervision**.

Table 5: Diagnostic results on the six-cup, multi-ball Cup Swap task.

Supervision	Individual Swap Acc.	Final-state Acc.
Vanilla supervision (final state only)	–	2.40
Process supervision (CoT-like)	99.78	97.93

4.2 Real-World Evaluation

4.2.1 Settings

Platform. We use the Galbot G1 humanoid robot as the physical platform for real-world evaluation. During all experiments, only the two arms and grippers are actuated, while the remaining joints and body components are kept fixed.

Task and Data. We consider a *pick-and-place x times* task shown in Figure 3, where the robot is required to sequentially pick up x objects and place them into a packaging bag. We collect teleoperated demonstrations for $x \in \{1, 2, 3, 4\}$, with 50 trajectories for each setting, resulting in 200 trajectories in total. The dataset contains 222,834 frames recorded at 30 FPS, corresponding to 2.06 hours of real-world data.

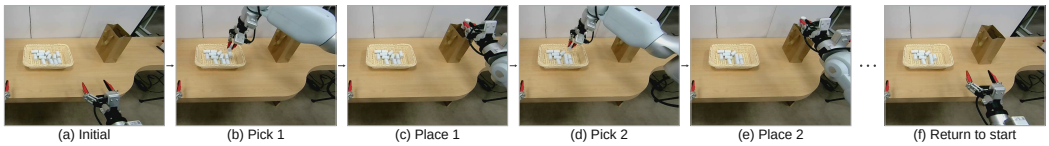


Figure 3: Task and environment for real-world experiments.

Implementation Details. We run inference on a single RTX 4090 GPU. The robot is controlled with a control interval of $\Delta t = 0.05\,\mathrm{s}$ per step. For real-world test, we do not use anchor frames. A trial is considered successful only if the robot transfers exactly the instructed number of objects, returns to its initial pose, and remains stationary for at least 10 s.

4.2.2 Results

Table 6: Real-world success rate (%, ↑) for picking and placing x objects. Avg. is the unweighted mean across the four target quantities.

Method	$x = 1$	$x = 2$	$x = 3$	$x = 4$	Avg.
$\pi_{0.5}$ (no memory)	5.0	20.0	20.0	0.0	11.25
LTM-π (ours)	**95.0**	**90.0**	**70.0**	**75.0**	**82.50**

Table 6 shows that LTM-π improves success rate across all four target quantities, achieving a success rate of 82.50% compared with 11.25% for the memory-free $\pi_{0.5}$, a gain of 71.25 percentage points. Qualitatively, the memory-free policy tends to continue picking until approximately N objects remain in the source box, where N is the mean remaining-object count observed during data collection, suggesting reliance on a familiar scene configuration rather than tracking completed transfers. By retaining interaction history, LTM-π more reliably executes the requested number of transfers and stops accordingly. Per-trial raw results and videos are provided in Appendix C.

Additional Findings. During real-robot deployment, we observe an additional benefit of policies equipped with our LTM method. When data collection contains many static frames at the beginning of a trajectory, e.g., due to recording being started before robot motion, the original $\pi_{0.5}$ policy can have difficulty initiating execution. Specifically, the robot may remain stationary at its initial pose for an extended period and only begin moving stochastically with a small probability. In contrast, the LTM-augmented policy consistently initiates motion after a short delay and exhibits much more stable execution onset. We attribute this behavior to the temporal information implicitly injected by the latent-transition memory, which provides the policy with a stronger notion of execution progress and constitutes an additional practical advantage of our method.

5 Conclusion

We introduced **Latent Transition Memory (LTM)**, a compact memory representation that summarizes interaction history as an ordered sequence of latent transitions, using one token per observed transition. Instantiated on $\pi_{0.5}$, LTM-π integrates frozen latent action encoders with a lightweight adapter and cache-compatible causal attention, enabling efficient incremental memory construction during online control. Across RoboMME, LIBERO-Mem, and a real-world counting task, LTM-π consistently improves history-dependent manipulation performance, demonstrating that **latent transitions provide an effective alternative to storing dense visual histories**.

Our ablations further show that the transition sequence provides useful temporal-progress cues, while observation-dependent transition content contributes additional gains beyond these cues. At the same time, the swap diagnostic highlights an important limitation: preserving informative local transitions does not by itself guarantee that a downstream policy can reliably compose them into persistent object-state estimates over long horizons. These results suggest that latent transitions are a promising primitive for compact robotic memory, while stronger mechanisms for temporal composition and state tracking remain an important direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools for the following purposes:

1. **Partial generation and review of experimental code.** Generative AI was used to translate algorithms designed by the authors into executable code, conduct code reviews, and assist with debugging when implementation errors occurred.
2. **Manuscript refinement.** Generative AI was used to improve the manuscript at both the grammatical and structural levels.

Generative AI was **not** involved in: (1) formulating the core research ideas or insights, (2) designing the experiments, or (3) analyzing the experimental results or conducting further scientific investigation based on those results. These aspects of the work were carried out entirely by the human authors.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024. doi: 10.48550/arXiv.2410.24164. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. *arXiv preprint arXiv:2307.15818*, 2023. doi: 10.48550/arXiv.2307.15818. URL https://arxiv.org/abs/2307.15818.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative Interactive Environments. *arXiv preprint arXiv:2402.15391*, 2024. doi: 10.48550/arXiv.2402.15391. URL https://arxiv.org/abs/2402.15391.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. UniVLA: Learning to Act Anywhere with Task-centric Latent Actions. *arXiv preprint arXiv:2505.06111*, 2025. doi: 10.48550/arXiv.2505.06111. URL https://arxiv.org/abs/2505.06111.

Egor Cherepanov, Nikita Kachaev, Daniil Zelezetsky, Aydar Bulatov, Artem Pshenitsyn, Yuri Kuratov, Alexey Skrynnik, Aleksandr I. Panov, and Alexey K. Kovalev. μVLA: On Recurrent Memory for Partially Observable Manipulation in VLA Models. *arXiv preprint arXiv:2606.12497*, 2026. doi: 10.48550/arXiv.2606.12497. URL https://arxiv.org/abs/2606.12497.

Suhwan Choi, Jaeyoon Jung, Sungkyung Kim, Yunsung Lee, and Youngjae Yu. Ponderpounce: A pretrained mllm as an episode context engine for robot control. *arXiv preprint arXiv:2608.24115*, 2026.

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

In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 3407–3415, 2026a. doi: 10.1609/aaai.v40i5.37337. URL https://ojs.aaai.org/index.php/AAAI/article/view/37337.

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

Yinpei Dai, Hongze Fu, Jayjun Lee, Yuejiang Liu, Haoran Zhang, Jianing Yang, Chelsea Finn, Nima Fazeli, and Joyce Chai. RoboMME: Benchmarking and Understanding Memory for Robotic Generalist Policies. *arXiv preprint arXiv:2603.04639*, 2026. doi: 10.48550/arXiv.2603.04639. URL https://arxiv.org/abs/2603.04639.

Shenyuan Gao, William Liang, Kaiyuan Zheng, Ayaan Malik, Seonghyeon Ye, Sihyun Yu, Wei-Cheng Tseng, Yuzhu Dong, Kaichun Mo, Chen-Hsuan Lin, Qianli Ma, Seungjun Nah, Loic Magne, Jiannan Xiang, Yuqi Xie, Ruijie Zheng, Dantong Niu, You Liang Tan, K. R. Zentner, George Kurian, Suneel Indupuru, Pooya Jannaty, Jinwei Gu, Jun Zhang, Jitendra Malik, Pieter Abbeel, Ming-Yu Liu, Yuke Zhu, Joel Jang, and Linxi "Jim" Fan. DreamDojo: A Generalist Robot World Model from Large-Scale Human Videos. *arXiv preprint arXiv:2602.06949*, 2026a. doi: 10.48550/arXiv.2602.06949. URL https://arxiv.org/abs/2602.06949.

Yihuai Gao, Jeff Jinyun Liu, Shuang Li, and Shuran Song. Gated memory policy: In-context memorization and adaptation. *arXiv preprint arXiv:2604.18933*, 2026b. URL https://arxiv.org/abs/2604.18933.

Yutong Hu, Fengjiao Chen, Xuezhi Cao, and Renaud Detry. 2AM: Grounding agent-side memory as guidance for steerable action models in long-horizon manipulation. *arXiv preprint arXiv:2609.11308*, 2026. URL https://arxiv.org/abs/2609.11308.

Li Ji, Siyin Wang, Pengfang Qian, Xiaopeng Yu, Yihai Tian, Zhaoye Fei, Jingjing Gong, and Xipeng Qiu. HiMe: Hierarchical Embodied Memory for Long-Horizon Vision-Language-Action Control. *arXiv preprint arXiv:2607.03449*, 2026. doi: 10.48550/arXiv.2607.03449. URL https://arxiv.org/abs/2607.03449.

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. *arXiv preprint arXiv:2606.20679*, 2026. URL https://arxiv.org/abs/2606.20679.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model. *arXiv preprint arXiv:2406.09246*, 2024. doi: 10.48550/arXiv.2406.09246. URL https://arxiv.org/abs/2406.09246.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. HAMLET: Switch your Vision-Language-Action Model into a History-Aware Policy. *arXiv preprint arXiv:2510.00695*, 2025. doi: 10.48550/arXiv.2510.00695. URL https://arxiv.org/abs/2510.00695.

Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, Hang Li, and Tao Kong. Vision-Language Foundation Models as Effective Robot Imitators. *arXiv preprint arXiv:2311.01378*, 2023. doi: 10.48550/arXiv.2311.01378. URL https://arxiv.org/abs/2311.01378.

Anthony Liang, Pavel Czempin, Matthew M. Hong, Yutai Zhou, Jingzhen Wang, Erdem Biyik, and Stephen Tu. CLAM: Continuous Latent Action Models for Robot Learning from Unlabeled Demonstrations. *arXiv preprint arXiv:2505.04999*, 2025. doi: 10.48550/arXiv.2505.04999. URL https://arxiv.org/abs/2505.04999.

Minghui Lin, Pengxiang Ding, Shu Wang, Zifeng Zhuang, Yang Liu, Xinyang Tong, Wenxuan Song, Shangke Lyu, Siteng Huang, and Donglin Wang. HiF-VLA: Hindsight, Insight and Foresight through Motion Representation for Vision-Language-Action Models. *arXiv preprint arXiv:2512.09928*, 2025. doi: 10.48550/arXiv.2512.09928. URL https://arxiv.org/abs/2512.09928.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*, 2025. doi: 10.48550/arXiv.2504.16054. URL https://arxiv.org/abs/2504.16054.

Hongyu Qu, Jianzhe Gao, Xiaobin Hu, Shaohuan Yang, Xinlei Yu, Rui Yan, Wenguan Wang, Xiangbo Shu, and Shuicheng Yan. Dual Latent Memory in Vision-Language-Action Models for Robotic Manipulation. *arXiv preprint arXiv:2607.07608*, 2026. doi: 10.48550/arXiv.2607.07608. URL https://arxiv.org/abs/2607.07608.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-Cognitive Memory in Vision-Language-Action Models for Robotic Manipulation. *arXiv preprint arXiv:2508.19236*, 2025. doi: 10.48550/arXiv.2508.19236. URL https://arxiv.org/abs/2508.19236.

Hao Shi, Weiye Li, Bin Xie, Yulin Wang, Renping Zhou, Tiancai Wang, Xiangyu Zhang, Ping Luo, and Gao Huang. MemoryVLA++: Temporal Modeling via Memory and Imagination in Vision-Language-Action Models. *arXiv preprint arXiv:2606.09827*, 2026. doi: 10.48550/arXiv.2606.09827. URL https://arxiv.org/abs/2606.09827.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, Karan Dhabalia, Michael Equi, Quan Vuong, Jost Tobias Springenberg, Sergey Levine, Chelsea Finn, and Danny Driess. MEM: Multi-Scale Embodied Memory for Vision Language Action Models. *arXiv preprint arXiv:2603.03596*, 2026. doi: 10.48550/arXiv.2603.03596. URL https://arxiv.org/abs/2603.03596.

Yufan Wei, Kun Zhou, Lingjun Mao, Zijun Zhang, Ziming Xu, Ziqiao Xi, Shuang Liang, Ruobing Han, Yuchen Yan, Xinyue Wang, Fan Feng, and Biwei Huang. Causally Debiased Latent Action Model for Embodied Action Conditioned World Models. *arXiv preprint arXiv:2607.09185*, 2026. doi: 10.48550/arXiv.2607.09185. URL https://arxiv.org/abs/2607.09185.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, Yiyu Ren, Kejia Zhang, Hui Yu, Jingmei Zhao, Shuai Zhou, Zhenqi Qiu, Houlong Xiong, Ziyu Wang, Zechen Wang, Ran Cheng, Yong-Lu Li, Yongtao Huang, Xing Zhu, Yujun Shen, and Kecheng Zheng. A Pragmatic VLA Foundation Model. *arXiv preprint arXiv:2601.18692*, 2026. doi: 10.48550/arXiv.2601.18692. URL https://arxiv.org/abs/2601.18692.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent Action Pretraining from Videos. *arXiv preprint arXiv:2410.11758*, 2024. doi: 10.48550/arXiv.2410.11758. URL https://arxiv.org/abs/2410.11758.

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. *arXiv preprint arXiv:2606.17463*, 2026. doi: 10.48550/arXiv.2606.17463. URL https://arxiv.org/abs/2606.17463.

A Online Inference Algorithm

Algorithm 1 Online Inference of LTM-π

Require: Instruction ℓ, frozen encoder E_{LAM}, projector P, policy π_θ, statistics $(\mu_{\mathcal{D}}, \sigma_{\mathcal{D}})$, lag k

$j, c \leftarrow 0$; $\mathcal{C} \leftarrow \mathrm{Prefill}(X^{\mathrm{lang}}(\ell))$
Receive o_0
while task is not terminated **do**
  $t \leftarrow j$; $\tau_t \leftarrow \max(0, t-k)$; receive state s_t
  Wait only for unfinished z_i with $c < i \leq \tau_t$
  for $i = c+1, \ldots, \tau_t$ **do**
    $m_i \leftarrow P((z_i - \mu_{\mathcal{D}})/\sigma_{\mathcal{D}})$; $\mathcal{C} \leftarrow \mathrm{AppendKV}(\mathcal{C}, m_i)$; $c \leftarrow i$
  end for
  $\mathcal{C}_t \leftarrow \mathrm{AppendKV}(\mathcal{C}, X^{\mathrm{vis}}(o_t))$
  $\mathbf{A}_t \sim \pi_\theta(\cdot \mid \mathcal{C}_t, s_t)$
  while executing $\mathbf{A}_t$ **do**
    Receive o_{j+1}; $j \leftarrow j+1$
    LaunchAsync $z_j \leftarrow E_{\mathrm{LAM}}(o_{j-1}, o_j)$ ▷ store in latent buffer
  end while
  Discard temporary visual KV entries; retain $\mathcal{C}$
end while

B Cup Swap Diagnostic Experiment Details

Task and dataset. We construct a synthetic Cup Swap task to isolate motion-memory and temporal state-tracking ability from robotic control. Each video contains six cups and between one and five balls, with the initial balls placed under distinct cups. The initial configuration is revealed at frame 42, after which the sequence contains 5–20 randomly generated swaps. Videos are rendered at 30 FPS. The dataset contains 14,000 videos, split into 11,176 training, 1,375 validation, and 1,449 test examples using a fixed random seed of 42. The task is to recover the final ball occupancy after composing the complete swap sequence.

Latent-transition extraction. We use the same frozen CD-LAM encoder family as in LTM-π. Each adjacent frame pair is encoded into a 32-dimensional latent transition with stride 1. To prevent the reveal event itself from leaking the initial state into the motion representation, we retain only transitions whose source frame is at or after the reveal frame. Letting $r = 42$ denote the reveal-frame index, we keep a transition z_i only when its source-frame index satisfies $\mathrm{src}(z_i) \geq r$. Thus, the latent stream contains only the post-reveal motion history and does not directly provide the final answer.

Input and latent-memory fusion. To isolate motion reasoning from visual recognition of the reveal frame, the initial cup occupancy is provided as text (`reveal_mode=text`). The base model is Qwen3-VL-2B-Instruct. Every four consecutive CD-LAM latents are compressed into one memory token and directly projected to the Qwen hidden dimension. We add sinusoidal temporal embeddings, process the latent stream with 8 layers of latent self-attention (16 heads), and let the language states read the latent memory through 8 layers of cross-attention (16 heads). Attention dropout is set to zero. Training uses bfloat16 and gradient checkpointing.

Supervision variants. We distinguish two objectives. **Vanilla (final-state only)** uses a cup-occupancy classification target and directly predicts whether each of the six cups contains a ball; it receives no supervision on the swap sequence or intermediate states. **Process supervision (CoT-like)** uses a process-query-tracking objective that supervises recovery of the swap process and the evolving state before predicting the final state. Although this objective does not generate a natural-language chain of thought, it explicitly supervises intermediate reasoning and is therefore CoT-like in function.

Training configuration. For the process-supervised run, we train on 8 H100 80GB GPUs with a per-GPU batch size of 4 (global batch size 32). The main learning rate is 1×10^{-4}, the classification-head learning rate is 3×10^{-4}, weight decay is 1×10^{-2}, and gradient clipping is 1.0. The Qwen backbone is frozen.

Additional metrics. For the process-supervised model, token accuracy is 99.67%, swap accuracy is 99.78%, swap-count accuracy is 100.00%, final-state accuracy is 97.93%, and full exact accuracy is 97.52%. For the Vanilla six-cup objective, the best exact accuracy is 2.40% (1.75% at the final checkpoint), with training exact accuracy of approximately 12.5%. A separate, substantially simpler three-cup, single-ball, fixed-five-swap classifier reaches 99.40% exact accuracy without process supervision, but we do not compare it directly because its state space and task difficulty are much smaller.

C Real-World Evaluation

Raw Results. We report the actual number of objects transferred into the packaging bag in every trial in Table 7. A trial is counted as correct if this number exactly matches the instructed quantity x, and the robot subsequently returns to its initial pose and remains stationary for at least 10 s. Each model is evaluated for 20 trials per target quantity, except the memory-free $\pi_{0.5}$ at $x = 3$ and $x = 4$: we conducted only 10 trials each because the policy repeatedly failed to stop after the requested number of transfers and continued picking, making individual trials excessively long. Unperformed trials are marked as "–" and excluded from the accuracy denominator.

Table 7: Per-trial transferred-object counts in real-world evaluation. Bold entries match the instructed quantity; "–" denotes an unperformed trial. Trial indices are local to each model–quantity setting.

	$\pi_{0.5}$ (no memory)				LTM-π (ours)			
Trial	$x=1$	$x=2$	$x=3$	$x=4$	$x=1$	$x=2$	$x=3$	$x=4$
1	0	1	4	5	**1**	**2**	**3**	**4**
2	2	1	5	7	**1**	**2**	**3**	3
3	2	**2**	5	8	**1**	1	**3**	3
4	2	3	4	6	**1**	**2**	4	**4**
5	3	3	6	8	**1**	**2**	4	**4**
6	4	3	**3**	5	**1**	**2**	**3**	**4**
7	4	5	4	7	**1**	**2**	**3**	**4**
8	3	4	**3**	6	**1**	**2**	2	**4**
9	3	3	6	6	2	**2**	**3**	3
10	4	3	4	8	**1**	**2**	**3**	3
11	3	**2**	–	–	**1**	**2**	**3**	**4**
12	2	3	–	–	**1**	**2**	**3**	**4**
13	3	1	–	–	**1**	**2**	2	**4**
14	**1**	1	–	–	**1**	**2**	4	**4**
15	2	**2**	–	–	**1**	**2**	**3**	**4**
16	3	3	–	–	**1**	**2**	**3**	**4**
17	3	3	–	–	**1**	**2**	**3**	**4**
18	5	4	–	–	**1**	**2**	**3**	**4**
19	4	**2**	–	–	**1**	1	**3**	3
20	3	3	–	–	**1**	**2**	4	**4**
Correct / Trials	1/20	4/20	2/10	0/10	19/20	18/20	14/20	15/20
Acc. (%)	5.0	20.0	20.0	0.0	95.0	90.0	70.0	75.0

Representative Real-Robot Videos. We provide representative videos of our real-robot experiments in the supplementary materials. These include four successful pick-and-place trials using our proposed LTM method: `ltm_pick_1_success.mp4`, `ltm_pick_2_success.mp4`, `ltm_pick_3_success.mp4`, and `ltm_pick_4_success.mp4`, as well as one representative failure case without memory: `pi05_nomemory_pick_1_fail.mp4`.