EMAG: EVENT MEMORY RETRIEVAL AND SPATIAL AFFORDANCE GROUNDING FOR NON-MARKOVIAN ROBOT MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Many real-world manipulation tasks are non-Markovian: the target of the current action may be determined by an event observed much earlier and therefore cannot be inferred from the current observation alone. Existing memory-augmented policies compress past observations into compact visual representations or textual summaries that track task progress, but do not jointly retain explicit event semantics and the visual evidence needed to re-ground a history-dependent target in the current scene. To address this issue, we introduce EMAG, which uses memory for semantic retrieval to identify which past events are relevant and for visual grounding to determine where the corresponding target is. Specifically, EMAG stores memory as captioned keyframes, pairing RGB snapshots of salient events with concise textual descriptions. Each caption serves as a semantic handle for memory retrieval, while the corresponding keyframe preserves the visual evidence needed for grounding. Given the task goal, progress summary, and recent observations, EMAG queries the stored captions to retrieve relevant memory entries, re-grounds the corresponding target in the current observation, and predicts an affordance map that specifies the target region to act on. The predicted affordance map modulates the visual features of a low-level vision-language-action (VLA) policy to guide action generation. Across all 16 RoboMME tasks, EMAG achieves an average success rate of 73.4%, exceeding the strongest official RoboMME baseline by 28.9 percentage points. On three real-robot tasks requiring object counting, object permanence, and sequence imitation, EMAG achieves an average success rate of 77.2%.

1 INTRODUCTION

Many real-world manipulation tasks are history-dependent: the correct action may rely on information from past events that is no longer available in the current observation. For example, a robot watches several cubes being covered by cups and must later lift the cup hiding a specified cube (Dai et al., 2026). Once the cubes are covered, the current image shows the cups but does not reveal which one conceals the target. The correct decision depends on the earlier covering event, violating the assumption that the current observation and task instruction form a Markov state sufficient for action selection (Kaelbling et al., 1998). Such tasks are therefore non-Markovian with respect to the current observation, requiring the robot to recall past events to guide task execution.

Existing methods address this need by incorporating memory into manipulation policies, allowing past experience to inform current decisions. These approaches retain historical information through visual representations, textual descriptions, or a combination of the two. Visual history can be compressed into compact contexts or encoded in recurrent memory states (Jang et al., 2025; Li et al., 2026). Textual summaries can complement short-horizon visual memory to track task progress over longer horizons (Torne et al., 2026). However, compact visual representations may lose event-specific visual details, while textual summaries may omit the appearance and spatial cues needed to distinguish the objects involved. Keyframe-based approaches preserve these cues by retaining selected historical observations as visual evidence (Sridhar et al., 2025; Yang et al., 2026a). In MemER, however, the policy revisits all selected keyframes at each decision, even when the relevant evidence may be confined to a small region in only a few frames. This can introduce redundant visual

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.

computation and requires the model to isolate subtle target cues from task-irrelevant visual content. More importantly, preserving or retrieving historical evidence does not by itself identify where the corresponding target lies in the current observation.

We introduce **EMAG**, a hierarchical policy for event-memory retrieval and spatial affordance grounding in history-dependent manipulation.[1] EMAG first identifies the past events relevant to the current decision, retrieves their associated image–caption pairs, and then combines these pairs with the predicted language subgoal and current observation to ground the corresponding target in the current scene. The grounded target is represented as a target-affordance map, a dense current-view spatial representation that marks the object, receptacle, interaction region, or route relevant to the current subgoal. This map provides an explicit spatial interface between high-level memory reasoning and low-level VLA control.

EMAG realizes this retrieve-then-ground process through two rounds of a shared high-level VLM. In Round 1, the policy reasons over the task goal, recent observations, a running progress summary, and stored event captions to update memory, predict the current language subgoal, and retrieve relevant events. In Round 2, only the retrieved image–caption pairs are combined with the predicted language subgoal and current observation to ground the historically determined target as a target-affordance map. Representative affordance-conditioned policies derive spatial guidance primarily from task instructions and initial or current observations (Nasiriany et al., 2024; Huang et al., 2025). EMAG instead conditions current-scene spatial grounding on retrieved event evidence. To translate the grounded target into control, EMAG uses the predicted target-affordance map to modulate the front-view visual features of the low-level VLA policy. This preserves the original visual context while providing spatial guidance for action generation.

We evaluate EMAG on all 16 RoboMME tasks, spanning the Counting, Permanence, Reference, and Imitation suites (Dai et al., 2026). EMAG achieves an average success rate of **73.4%**, outperforming FrameSamp+Modul, the strongest official RoboMME baseline, by **28.9** percentage points. We further evaluate EMAG on three real-world non-Markovian manipulation tasks: COUNT PLACE, BLOCK UNMASK, and BUTTON REPLAY, where it achieves an average success rate of **77.2%**. Controlled ablations show that selective memory retrieval and affordance-based feature modulation each improve success by more than 10 percentage points over their respective alternatives.

Our contributions are threefold:

- We introduce EMAG, a hierarchical memory-augmented policy for non-Markovian manipulation that explicitly connects past event evidence with spatial grounding of the current interaction target.
- We propose a retrieve-then-ground architecture that stores salient events as captioned keyframes, selectively retrieves task-relevant event evidence, and re-grounds the corresponding target as an affordance map for low-level VLA control.
- We evaluate EMAG on all 16 RoboMME tasks and three real-robot history-dependent manipulation tasks, achieving 73.4% average success on RoboMME and improving over its strongest official baseline by 28.9 percentage points, while reaching 77.2% success on the real robot.

2 RELATED WORK

2.1 MEMORY-AUGMENTED ROBOT POLICIES

Partially observable, long-horizon manipulation requires robot policies to reason beyond the current observation. One line of work encodes history into compact context tokens (Jang et al., 2025; Koo et al., 2026) or accumulates it through recurrent states (Li et al., 2026). Gated Memory Policy learns when and what to recall (Gao et al., 2026), while MemoryWAM combines recent observations, event-boundary frames, and compressed gist tokens (Yang et al., 2026b). PonderPounce instead reuses the native causal context of a pretrained multimodal large language model (MLLM) as episodic memory and asynchronously routes continuous cognition tokens to a low-level action model (Choi et al., 2026). SimpleMemVLA (Yin et al., 2026) retains sampled visual history in the

[1] Project website: `https://emag-anon.github.io/`.

pretrained backbone's native timestamped-video format and relies on native self-attention rather than a dedicated memory module. These approaches efficiently extend temporal context, but generally do not expose individual historical evidence through an explicit, interpretable memory interface.

Another line of work maintains explicit visual or semantic memory. MemER selects task-relevant historical frames and generates language subgoals (Sridhar et al., 2025); MEM combines short-horizon video memory with long-horizon language memory (Torne et al., 2026); MemoryVLA constructs a perceptual-cognitive memory bank (Shi et al., 2026); and Mem-0 couples short-term buffers, keyframe memory, and task-phase estimation (Chen et al., 2026b). HiMe and Chameleon further explore hierarchical memory management and prospective episodic recall (Ji et al., 2026; Guo et al., 2026).

To reduce redundancy in long histories, SAM2Act+ introduces a memory bank and memory attention (Fang et al., 2025), while EventVLA, Keyframe-Chaining VLA, and KEMO organize history around sparse event keyframes (Yang et al., 2026a; Chen et al., 2026d; Zeng et al., 2026). HALO learns to retrieve control-relevant evidence through auxiliary supervision (Shah et al., 2026), whereas HistRISE retains motion and state changes using object-centric point tracks (Chen et al., 2026a). EMAG jointly predicts an event caption for each selected keyframe. Within each resulting memory entry, the caption serves as a semantic retrieval cue, while the paired RGB keyframe provides visual evidence for subsequent spatial grounding.

2.2 Visual Grounding and Affordance-Conditioned Policies

Visual grounding and affordance learning aim to translate semantic intent into spatial signals that can directly guide robot control. Where2Act (Mo et al., 2021) predicts pixel-level actionability for articulated objects, while CLIPort (Shridhar et al., 2021) separates semantic "what" reasoning from spatial "where" prediction. MOO (Stone et al., 2023) connects open-vocabulary object grounding to manipulation policies, and KITE (Sundaresan et al., 2023), MOKA (Liu et al., 2024), and RoboPoint (Yuan et al., 2024) use keypoints or affordance points to bridge language understanding and skill execution. VoxPoser (Huang et al., 2023) grounds language-derived constraints into composable 3D value maps. VRB (Bahl et al., 2023), RAM (Kuang et al., 2024), and Robo-ABC (Ju et al., 2024) instead learn, retrieve, or transfer contact regions and motion patterns from human videos or affordance memories.

A complementary line of work treats spatial predictions as intermediate representations for hierarchical control. RT-Affordance (Nasiriany et al., 2024) conditions a low-level policy on affordance plans, CoA-VLA (Li et al., 2025) generates sequential chains of visual and textual affordances, and AffordDP (Wu et al., 2025) guides a diffusion policy with 3D contact points and post-contact trajectories. Other methods adopt structured visual primitives: Point-VLA (Yu et al., 2026a), Pointing-VLA (Chen et al., 2026c), and VP-VLA (Wang et al., 2026b) use points, heatmaps, trajectories, crosshairs, or boxes as interfaces between reasoning and control. Tsai et al. (2026) directly inject a grounded 3D point into the action head.

Recent systems more tightly couple dense grounding or affordance representations with action generation. RoboGround (Huang et al., 2025) uses grounding masks as intermediate guidance, while Yang et al. (2025) study object-centric, pose-aware, and depth-aware mid-level representations. PEEK (Zhang et al., 2025) overlays task-relevant masks and end-effector paths on policy observations, while MaskWAM (Yu et al., 2026b) uses masks as both input prompts and prediction targets. Afford-VLA (Wang et al., 2026a), AffordVLA (Kong et al., 2026), AffordanceVLA (Yu et al., 2026c), and VLAff (Oh et al., 2026) further explore explicit affordance decoding, implicit feature alignment, structured Which2Act–Where2Act–How2Act prediction, and unified heatmap–grasp–trajectory modeling, respectively.

A few studies explicitly connect temporal information with current spatial grounding. TraceVLA (Zheng et al., 2025) renders past state–action traces as visual prompts, Spatial Traces (Patratskiy et al., 2025) combines point trajectories with depth, and history-aware point tracking (Chen et al., 2026a) constructs compact object-centric history representations. In contrast, EMAG targets non-Markovian manipulation tasks in which past events determine the current interaction target. It grounds the current target from selectively retrieved captioned-keyframe evidence and uses the resulting affordance map to guide low-level VLA control.

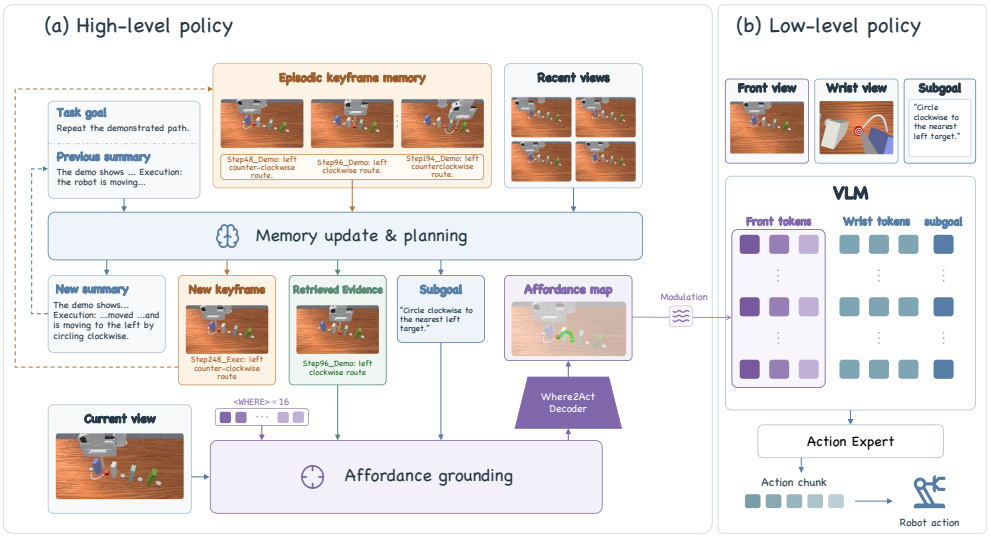

Figure 1: Overview of EMAG. The high-level policy updates the progress summary and captioned keyframe memory, retrieves task-relevant event evidence, predicts the current language subgoal, and grounds the corresponding target as an affordance map in the current view. The low-level VLA policy uses this map to modulate front-view visual features and generate an action chunk conditioned on the language subgoal.

3 EMAG

3.1 Problem Setup and Method Overview

We consider history-dependent manipulation tasks specified by a task goal G. At the h-th high-level inference step, the system receives recent frames $\mathcal{C}_h$, whose latest front-view image is I_h, together with the previous progress summary S_{h-1} and episodic memory $\mathcal{M}_{h-1}$. Each memory entry is represented as $m_i = (K_i, I_{K_i}, E_i)$, where K_i is the keyframe identifier, I_{K_i} is the corresponding RGB keyframe, and E_i is its event caption. We denote the number of stored memory entries by $N_{h-1} = |\mathcal{M}_{h-1}|$. During execution, given this history, the policy must infer the current language subgoal ℓ_h and ground the historically determined target as an actionable region in the current scene.

As illustrated in Figure 1, EMAG follows a retrieve-then-ground-then-act process using two rounds of a shared high-level VLM. Round 1 maintains task progress and event memory, predicts the current language subgoal, and retrieves relevant event identifiers. Round 2 combines the retrieved image–caption pairs with the subgoal and current observation to predict a target-affordance map A_h, which modulates the front-view visual features of the low-level VLA for action generation.

3.2 Event Memory Retrieval and History-Conditioned Spatial Grounding

Central to EMAG's high-level policy is an event memory composed of captioned keyframes. Each memory entry combines two complementary forms of information: the event caption E_i provides a compact semantic description for identifying relevant history, while the corresponding RGB keyframe I_{K_i} preserves the appearance and spatial cues required for subsequent grounding. This design allows EMAG to first select relevant past events through their semantics and access their visual evidence only after retrieval.

Event Memory Update and Retrieval. At execution step h, Round 1 maps the task goal G, recent frames $\mathcal{C}_h$, previous summary S_{h-1}, and stored identifier–caption pairs $\{(K_i, E_i)\}_{i=0}^{N_{h-1}-1}$ to the updated summary S_h, an optional new captioned keyframe m_h^{new}, the current language subgoal

ℓ_h, and ordered relevant-keyframe identifiers $\mathcal{J}_h = [K_{j_1}, \ldots, K_{j_r}]$, where $r = |\mathcal{J}_h|$:

$$(S_h,\, m_h^{\text{new}},\, \ell_h,\, \mathcal{J}_h) = \pi_{\text{HL}}^{\text{R1}}\left(G,\, \mathcal{C}_h,\, S_{h-1},\, \{(K_i, E_i)\}_{i=0}^{N_{h-1}-1}\right). \tag{1}$$

When a salient event is observed, m_h^{new} contains a representative recent frame and its generated caption; otherwise, $m_h^{\text{new}} = \varnothing$. The event memory is updated as

$$\mathcal{M}_h = \begin{cases} \mathcal{M}_{h-1} \cup \{m_h^{\text{new}}\}, & m_h^{\text{new}} \neq \varnothing, \\ \mathcal{M}_{h-1}, & m_h^{\text{new}} = \varnothing. \end{cases} \tag{2}$$

The evidence retrieved by $\mathcal{J}_h$ is

$$\mathcal{R}_h = \left[\left(I_{K_{j_k}}, E_{j_k}\right)\right]_{k=1}^{r}, \tag{3}$$

which is loaded and passed to Round 2 in retrieval order. When $r = 0$, we set $\mathcal{R}_h = []$. Event retrieval identifies which past events are relevant to the current decision, but does not specify where that target lies in the current view.

History-Conditioned Spatial Grounding. Round 2 bridges this gap by predicting a target-affordance map $A_h \in [0,1]^{H \times W}$, where H and W denote the height and width of the affordance map, and $A_h(u)$ measures the relevance of pixel u to the target. The target may be an object, receptacle, interaction region, or spatial route. Round 2 processes the current observation I_h, the Round 1-predicted subgoal ℓ_h, and the retrieved evidence $\mathcal{R}_h$ together with learnable <WHERE> tokens $\mathcal{T}_{\text{where}}$. The high-level VLM's final-layer hidden embeddings at <WHERE> token positions form the history-conditioned target representation:

$$\mathbf{H}_h^{\text{where}} = \left[\pi_{\text{HL}}^{\text{R2}}\left(\ell_h,\, \mathcal{R}_h,\, I_h,\, \mathcal{T}_{\text{where}}\right)\right]_{\texttt{<WHERE>}}, \tag{4}$$

where $[\cdot]_{\texttt{<WHERE>}}$ selects the output states corresponding to the <WHERE> tokens. To ground the history-conditioned target representation in the current scene, we introduce a lightweight Where2Act decoder that combines it with spatial visual features from the current front-view image. The current-view features serve as spatial queries, while the target representation provides cross-attention keys and values, enabling dense target-affordance prediction. Let F_h^{cur} denote the spatial visual features extracted from I_h. We map F_h^{cur} to spatial decoder queries using Φ_{img} and project the history-conditioned target representation into the cross-attention memory using Φ_{where}. D_{where} denotes the complete Where2Act decoder, including spatial decoding and dense mask prediction. We compute the target-affordance map as

$$A_h = \sigma\left(D_{\text{where}}\left(\underbrace{\Phi_{\text{img}}(F_h^{\text{cur}})}_{\mathbf{Q}}, \underbrace{\Phi_{\text{where}}\left(\mathbf{H}_h^{\text{where}}\right)}_{\mathbf{K}=\mathbf{V}}\right)\right). \tag{5}$$

Each query corresponds to a spatial location in the current view, while the keys and values encode the history-conditioned target representation formed from the language subgoal, retrieved event evidence, and current observation. The decoder evaluates current-scene locations against this representation to produce the target-affordance map A_h.

3.3 Affordance-Conditioned Low-Level Modulation

To incorporate the grounded target into low-level control, EMAG aligns the predicted affordance map with the spatial token grid of the front-view visual stream and injects the resulting spatial signal through feature modulation. Let $\bar{A}_h \in \{0,1\}^{H \times W}$ denote the binary map obtained by thresholding A_h, and let $\mathcal{P}_j$ denote the image region corresponding to visual token j. We compute the affordance occupancy of each token and convert it into a soft spatial gate:

$$o_j = \frac{1}{|\mathcal{P}_j|} \sum_{u \in \mathcal{P}_j} \bar{A}_h(u), \qquad g_j = 1 - \exp\left(-\frac{o_j}{\tau}\right), \tag{6}$$

where τ controls the sensitivity of the gate to partial target coverage. The resulting gates associate the predicted actionable region with the corresponding front-view visual tokens.

At each selected visual layer $k \in \Omega_{\text{mod}}$, EMAG modulates token j as

$$\widetilde{\mathbf{x}}_j^{(k)} = \mathbf{x}_j^{(k)} + g_j \boldsymbol{\beta}^{(k)}, \tag{7}$$

where $\mathbf{x}_j^{(k)} \in \mathbb{R}^d$ denotes the front-view token representation at layer k and $\boldsymbol{\beta}^{(k)} \in \mathbb{R}^d$ is a learnable layer-specific additive vector initialized to zero. The same layer-specific vector is shared across spatial locations, while the token-specific gate g_j determines the strength of the injected target conditioning. Tokens with greater target overlap therefore receive stronger modulation, whereas tokens with no target overlap remain unchanged.

We apply the modulation only to the front-view visual stream; wrist-view and language tokens are not directly modulated. The modulated visual features are then consumed by the low-level VLA through its native action-generation pathway together with the language subgoal ℓ_h. When the thresholded affordance map $\bar{A}_h$ is all zero, $g_j = 0$ for all tokens and Eq. 7 reduces to the unmodulated low-level representation.

3.4 Training Objective

To jointly optimize language generation, event-memory retrieval, and spatial grounding, we apply a next-token prediction loss $\mathcal{L}_{\text{LM}}$ to all structured textual outputs, including the progress summary, event caption, language subgoal, and retrieved memory identifiers.

Because the retrieved memory identifiers occupy only a small fraction of the output sequence, we additionally apply a cross-entropy loss $\mathcal{L}_{\text{ret}}$ at the retrieval-token positions. This provides stronger supervision for event-memory retrieval without introducing a separate retrieval head.

For the target-affordance map predicted in Round 2, we use a class-balanced binary cross-entropy loss together with a Dice loss:

$$\mathcal{L}_{\text{where}} = \mathcal{L}_{\text{bal-BCE}} + \mathcal{L}_{\text{Dice}}, \tag{8}$$

where $\mathcal{L}_{\text{bal-BCE}}$ accounts for the foreground–background imbalance in the affordance mask, while $\mathcal{L}_{\text{Dice}}$ directly supervises the spatial overlap between the predicted affordance region and the ground-truth target region. Detailed definitions are provided in Appendix C.1.

The final high-level training objective is

$$\mathcal{L}_{\text{HL}} = \mathcal{L}_{\text{LM}} + \lambda_1 \mathcal{L}_{\text{ret}} + \lambda_2 \mathcal{L}_{\text{where}}. \tag{9}$$

Here, λ_1 and λ_2 weight the retrieval and spatial-grounding objectives, respectively. The low-level VLA is trained separately with its standard flow-matching action objective. Implementation details are provided in Appendix C.1.

4 Experiments

We evaluate EMAG on RoboMME and three real-robot tasks, addressing the following questions:

1. How effectively can EMAG solve diverse history-dependent manipulation tasks compared with existing memory-augmented policies?
2. Does EMAG's retrieve-then-ground mechanism remain effective in real-world manipulation under visual and execution uncertainty?
3. How do selective memory retrieval and affordance-conditioned feature modulation contribute to the final performance?

4.1 Experimental Setup

RoboMME. We evaluate EMAG on all 16 history-dependent manipulation tasks in RoboMME (Dai et al., 2026). The benchmark contains four task suites—Counting, Permanence, Reference, and Imitation—which evaluate temporal, spatial, object, and procedural memory, respectively. The primary metric is episode success rate. We evaluate EMAG using three independently trained policy seeds, with 50 episodes per task and seed, for a total of 150 episodes per task.

Table 1: Success rates (%) on all 16 RoboMME tasks. The best learned-policy result in each column is shown in bold. Within the oracle references, only the best average is bold. † Results reported by RoboMME (Dai et al., 2026). ‡ Results reported by PonderPounce (Choi et al., 2026).

	Counting				Permanence				Reference				Imitation				Average
Method	Bin Fill	Pick Xtimes	Swing Xtimes	Stop Cube	Video Unmask	Button Unmask	Video UnmaskS	Button UnmaskS	Pick Highlight	Video Repick	Video PlaceBtn	Video PlaceOrd	Move Cube	Insert Peg	Pattern Lock	Route Stick	
Oracle 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
AffordSG+Oracle (ours)	88.67	99.33	98.67	43.33	100.00	97.33	99.33	96.00	90.00	97.33	100.00	99.33	70.67	33.33	100.00	96.67	**88.13**
Learned policies																	
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
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
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
EMAG (ours)	**72.67**	89.33	**96.00**	52.00	**97.33**	93.33	34.00	**56.00**	**86.00**	**62.00**	**85.33**	**85.33**	79.33	**21.33**	**74.00**	**90.67**	**73.42**

Compared methods. We compare EMAG with SAM2ACT+ (Fang et al., 2025), MEMER (Sridhar et al., 2025), FRAMESAMP+MODUL (Dai et al., 2026), and PONDERPOUNCE (Choi et al., 2026). Results for SAM2ACT+, MEMER, and FRAMESAMP+MODUL are reported by RoboMME, while the 9B PONDERPOUNCE result at the base training-data scale is taken from its original paper. We additionally report SIMPLESG+ORACLE, GROUNDSG+ORACLE, and AFFORDSG+ORACLE as oracle references. The first two use oracle simple and grounded language subgoals, respectively. AFFORDSG+ORACLE measures low-level performance when both the simple language subgoal and spatial affordance are provided by an oracle, and is averaged over three independently trained policy seeds.

Implementation. In both the RoboMME and real-robot settings, EMAG uses Qwen3.5-4B (Team, 2026) as its high-level policy and initializes its low-level policy from the public $\pi_{0.5}$ base checkpoint (Intelligence et al., 2025). Detailed training and evaluation protocols for EMAG are provided in Appendix C.

4.2 MAIN RESULTS ON ROBOMME

Table 1 first highlights the benefit of explicit spatial guidance. AffordSG+Oracle achieves 88.13% average success, compared with 84.08% for GroundSG+Oracle and 49.58% for SimpleSG+Oracle. The advantage over grounded language is especially clear on INSERTPEG and ROUTESTICK, where AffordSG+Oracle improves from 15.56% to 33.33% and from 55.56% to 96.67%, respectively. This suggests that dense affordance maps provide complementary spatial guidance for where to act.

Among learned policies, EMAG achieves an average success rate of **73.42%**, exceeding the reported FrameSamp+Modul and PonderPounce averages by 28.91 and 12.59 percentage points, respectively. At the suite level, EMAG achieves 77.50%, 70.17%, 79.67%, and 66.33% on Counting, Permanence, Reference, and Imitation, respectively, outperforming the reported PonderPounce results across all four suites. These results demonstrate the effectiveness of EMAG across diverse history-dependent manipulation settings that require both memory reasoning and spatially precise control.

4.3 REAL-WORLD EXPERIMENTS

We further evaluate EMAG on three physical-robot tasks covering distinct forms of history dependence. As shown in Figure 2, **Count Place** requires the robot to place a specified number of blocks of the target color into a receptacle and then press a button to signal task completion. **Block Unmask** presents multiple blocks before they are covered by opaque cups. Given one or more target blocks specified by the task goal, the robot must later lift the cups hiding those targets. **Button Replay** requires the robot to observe and reproduce a three- to six-step button-press sequence. The language supervision provides neither button color names nor other symbolic identity labels. The policy must therefore establish button correspondences from the visual demonstration and re-ground the demonstrated button at each step in the current scene. These tasks respectively evaluate accumulated progress tracking, target memory under occlusion, and ordered event memory with visual grounding. The robot platform, task randomization, and detailed task designs are provided in Appendix D.

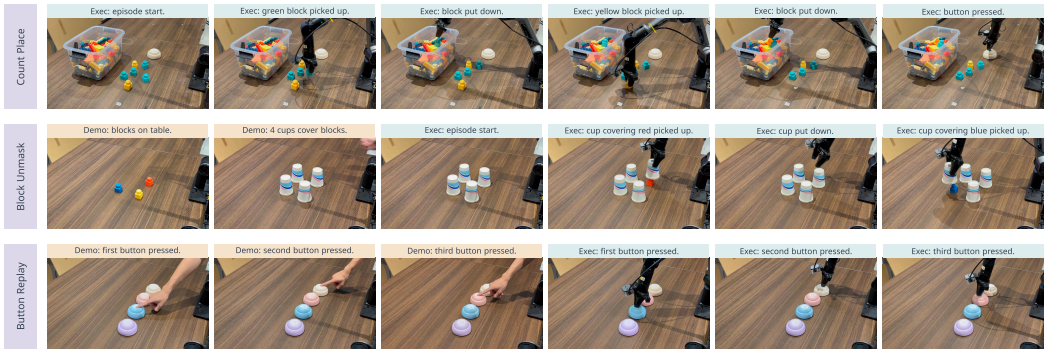

Figure 2: Real-world task setups for COUNT PLACE, BLOCK UNMASK, and BUTTON REPLAY.

We compare a fine-tuned $\pi_{0.5}$ baseline, FrameSamp+Modul, and EMAG. As a reference, we also report **Human Performance**, obtained by having a human operator directly teleoperate the robot on the same tasks. All evaluations use the same task configurations and initial-state assignments. For each learned policy, we evaluate three independently trained policy seeds. Each seed is evaluated on the same 20 episode configurations, resulting in 60 trials per method and task. Human Performance is evaluated over three repeated rounds by the same operator on the same 20 configurations, also resulting in 60 trials per task. We report the average success rate and sample standard deviation across policy seeds for learned methods and across repeated rounds for Human Performance.

As shown in Figure 3, EMAG achieves an average success rate of **77.2%** across the three tasks, substantially outperforming FrameSamp+Modul at **9.4%** and the fine-tuned $\pi_{0.5}$ at **3.3%**. Specifically, EMAG achieves $88.3 \pm 2.9\%$ on Count Place, $63.3 \pm 7.6\%$ on Block Unmask, and $80.0 \pm 8.7\%$ on Button Replay. In contrast, the fine-tuned $\pi_{0.5}$ obtains zero success on both Block Unmask and Button Replay. FrameSamp+Modul reaches 20.0% on Count Place and 8.3% on Block Unmask, but likewise fails to complete Button Replay. Human Performance reaches **98.3%**, confirming that the tasks are largely feasible under direct teleoperation while leaving a clear gap between EMAG and the human teleoperation reference.

Together, these results show that EMAG can solve physical-robot manipulation tasks requiring distinct forms of history dependence, including counting, target selection under occlusion, and sequence imitation. Appendix F provides a task-wise analysis of the characteristic failure modes of each learned policy.

4.4 ABLATION STUDIES

We ablate two key design choices in EMAG: selective memory retrieval and affordance-conditioned feature modulation. We conduct controlled ablations on four RoboMME tasks, one from each suite: BinFill, VideoUnmask, VideoPlaceButton, and RouteStick. All configurations use identical evaluation protocols. For efficiency, the high-level policies for Full EMAG-4 and w/o retrieval are trained on $4\times$-subsampled data, while affordance as context reuses the Full EMAG-4 high-level checkpoint. Absolute results are therefore not directly comparable to the full 16-task model, while comparisons among ablations remain controlled.

Full EMAG-4 denotes the complete model trained on the four-task subset. To isolate the effect of selective retrieval, **w/o retrieval** modifies only the high-level policy: Round 1 no longer selects relevant memory entries, and Round 2 instead receives all stored image–caption pairs. The low-level policy is kept identical to Full EMAG-4. To isolate the low-level affordance-conditioning mechanism, **affordance as context** keeps the Full EMAG-4 high-level policy fixed and replaces feature modulation with visual-context conditioning. The predicted affordance map is replicated to three channels and provided as a third visual input alongside the front and wrist views, and the low-level policy is retrained with the same budget.

As shown in Table 2, removing selective retrieval reduces average success from 77.67% to 62.17%. Since this variant retains the same captioned keyframe memory and instead provides all stored mem-

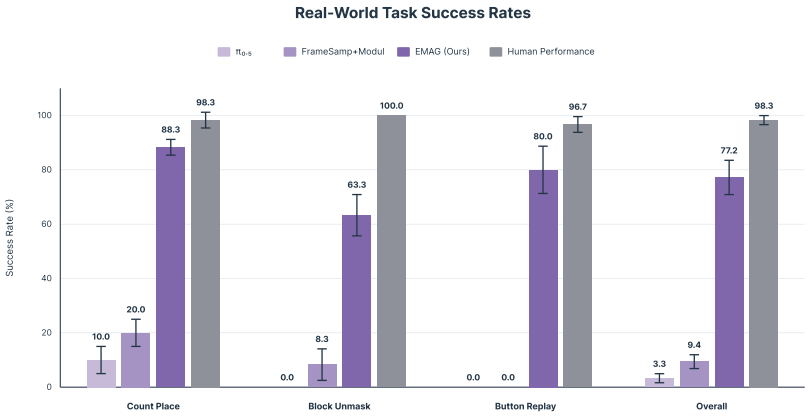

Figure 3: Real-world success rates on COUNT PLACE, BLOCK UNMASK, and BUTTON REPLAY. Results for learned policies are averaged over three independently trained policy seeds, with 20 trials per seed. Error bars denote the sample standard deviation across policy seeds for learned methods and across repeated rounds for Human Performance.

Table 2: Ablation results on four RoboMME tasks, with one task selected from each suite. Mean success rates and sample standard deviations across three independently trained seeds are reported in percent.

Method	BinFill	VideoUnmask	VideoPlaceButton	RouteStick	Average
Full EMAG-4	$\mathbf{69.33 \pm 3.06}$	87.33 ± 2.31	$\mathbf{81.33 \pm 2.31}$	$\mathbf{72.67 \pm 4.16}$	$\mathbf{77.67 \pm 1.53}$
w/o Retrieval	59.33 ± 3.06	88.67 ± 3.06	31.33 ± 6.43	69.33 ± 8.08	62.17 ± 0.58
Affordance as Context	52.00 ± 5.29	$\mathbf{89.33 \pm 2.31}$	66.00 ± 5.29	58.00 ± 7.21	66.33 ± 4.80

ories to Round 2, the result shows that memory availability alone is insufficient; selective retrieval is necessary to turn stored history into useful conditioning for spatial grounding. Replacing feature modulation with the affordance map as an additional visual input reduces average success from 77.67% to 66.33%. Because the high-level policy and affordance predictions are unchanged, this comparison isolates the low-level conditioning mechanism, showing that directly modulating spatial visual features provides a more effective interface than treating the affordance map as additional visual context.

5 CONCLUSION AND LIMITATIONS

We introduced EMAG, a retrieve-then-ground policy for history-dependent robot manipulation. EMAG separates identifying the past events relevant to the current decision from re-grounding their targets in the current scene, using captioned keyframes as event memory and affordance maps as an explicit spatial interface to low-level VLA control. EMAG achieves 73.4% average success across all 16 RoboMME tasks and 77.2% across three real-robot tasks, demonstrating the effectiveness of this design across diverse forms of history-dependent manipulation.

EMAG currently relies on task-specific supervision and is best suited to event-structured history dependence, where current decisions can be resolved by recalling several salient past events and their associated visual context. Its sparse captioned-keyframe memory is less suited to tasks that require continuous object identity tracking. Extending event-based memory with representations that preserve continuous temporal correspondence is an important direction for future work.

AI USE STATEMENT

Generative AI tools were used to assist with software implementation, support the drafting and editing of the manuscript, summarize and identify relevant literature, assist with figure design, and improve the organization and readability of the paper. In particular, GPT-5.6 Sol was used as an authoring assistant to translate task definitions and temporal dependencies into deterministic annotation rules, but was not invoked on individual trajectories or training samples. AI-assisted code was reviewed, tested, and verified by the authors. All research decisions, experimental designs, analyses, and reported results were carried out and verified by the authors. The authors take full responsibility for the final content of the paper, including all claims, code, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

The real-robot evaluation includes human teleoperation solely as a performance reference. The human operator was a member of the research team and participated only in this teleoperation evaluation. We report only aggregate task-performance results and collect no identifying or sensitive personal information. All robot experiments were conducted under standard laboratory safety procedures.

REPRODUCIBILITY STATEMENT

We provide detailed architecture, memory-management, data-construction, training, and evaluation procedures in the appendix. This includes the implementation of the Where2Act decoder and low-level affordance modulation, the construction of event-memory and target-affordance supervision, training hyperparameters, and detailed simulation and real-robot evaluation protocols. These details are intended to support reproduction of the reported results.

REFERENCES

Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from human videos as a versatile representation for robotics, 2023. URL https://arxiv.org/abs/2304.08488.

Jingjing Chen, Hongjie Fang, Chenxi Wang, Shiquan Wang, and Cewu Lu. History-aware visuomotor policy learning via point tracking, 2026a. URL https://arxiv.org/abs/2509.17141.

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, 2026b. URL https://arxiv.org/abs/2603.01229.

Xiwen Chen, Zelin Li, Zhiruo Zhou, Huiming Chen, Chenwei Wang, and Xiaojun Zhu. Pointing-vla: Typed spatial grounding interfaces for vision-language-action manipulation, 2026c. URL https://arxiv.org/abs/2608.23138.

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

Suhwan Choi, Jaeyoon Jung, Sungkyung Kim, Yunsung Lee, and Youngjae Yu. Ponderpounce: A pretrained mllm as an episode context engine for robot control, 2026. URL https://arxiv.org/abs/2608.24115.

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, 2026. URL https://arxiv.org/abs/2603.04639.

Haoquan Fang, Markus Grotz, Wilbert Pumacay, Yi Ru Wang, Dieter Fox, Ranjay Krishna, and Jiafei Duan. Sam2act: Integrating visual foundation model with a memory architecture for robotic manipulation, 2025. URL https://arxiv.org/abs/2501.18564.

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

Xinying Guo, Chenxi Jiang, Hyun Bin Kim, Yuhang Han, Ying Sun, Yang Xiao, and Jianfei Yang. Chameleon: Control-indexed prospective memory for visuomotor manipulation, 2026. URL https://arxiv.org/abs/2603.24576.

Haifeng Huang, Xinyi Chen, Yilun Chen, Hao Li, Xiaoshen Han, Zehan Wang, Tai Wang, Jiangmiao Pang, and Zhou Zhao. Roboground: Robotic manipulation with grounded vision-language priors, 2025. URL https://arxiv.org/abs/2504.21530.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models, 2023. URL https://arxiv.org/abs/2307.05973.

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, 2025. URL https://arxiv.org/abs/2504.16054.

Huiwon Jang, Sihyun Yu, Heeseung Kwon, Hojin Jeon, Younggyo Seo, and Jinwoo Shin. Contextvla: Vision-language-action model with amortized multi-frame context, 2025. URL https://arxiv.org/abs/2510.04246.

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, 2026. URL https://arxiv.org/abs/2607.03449.

Yuanchen Ju, Kaizhe Hu, Guowei Zhang, Gu Zhang, Mingrun Jiang, and Huazhe Xu. Robo-abc: Affordance generalization beyond categories via semantic correspondence for robot manipulation, 2024. URL https://arxiv.org/abs/2401.07487.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X.

Weijie Kong, Zhian Su, Wei Yu, and Huixu Dong. Affordvla: Injecting affordance representations into vision-language-action models via implicit feature alignment, 2026. URL https://arxiv.org/abs/2605.17517.

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, 2026. URL https://arxiv.org/abs/2510.00695.

Yuxuan Kuang, Junjie Ye, Haoran Geng, Jiageng Mao, Congyue Deng, Leonidas Guibas, He Wang, and Yue Wang. Ram: Retrieval-based affordance transfer for generalizable zero-shot robotic manipulation, 2024. URL https://arxiv.org/abs/2407.04689.

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

Jinming Li, Yichen Zhu, Zhibin Tang, Junjie Wen, Minjie Zhu, Xiaoyu Liu, Chengmeng Li, Ran Cheng, Yaxin Peng, Yan Peng, and Feifei Feng. Coa-vla: Improving vision-language-action models via visual-textual chain-of-affordance, 2025. URL https://arxiv.org/abs/2412.20451.

Fangchen Liu, Kuan Fang, Pieter Abbeel, and Sergey Levine. Moka: Open-world robotic manipulation through mark-based visual prompting, 2024. URL https://arxiv.org/abs/2403.03174.

Kaichun Mo, Leonidas Guibas, Mustafa Mukadam, Abhinav Gupta, and Shubham Tulsiani. Where2act: From pixels to actions for articulated 3d objects, 2021. URL https://arxiv.org/abs/2101.02692.

Soroush Nasiriany, Sean Kirmani, Tianli Ding, Laura Smith, Yuke Zhu, Danny Driess, Dorsa Sadigh, and Ted Xiao. Rt-affordance: Affordances are versatile intermediate representations for robot manipulation, 2024. URL https://arxiv.org/abs/2411.02704.

Jihoon Oh, Kento Kawaharazuka, and Kei Okada. Vlaff: Vision-language-affordance model for unified actionable affordances, 2026. URL https://arxiv.org/abs/2608.05215.

OpenAI. GPT-5.6: Frontier intelligence that scales with your ambition, July 2026. URL https://openai.com/index/gpt-5-6/.

Maxim A. Patratskiy, Alexey K. Kovalev, and Aleksandr I. Panov. Spatial traces: Enhancing vla models with spatial-temporal understanding, 2025. URL https://arxiv.org/abs/2508.09032.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos, 2024. URL https://arxiv.org/abs/2408.00714.

Rutav Shah, Yisu Li, Femi Bello, Yuke Zhu, and Roberto Martín-Martín. Memory retrieval in visuomotor policies for long-horizon robot control, 2026. URL https://arxiv.org/abs/2606.25136.

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

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation, 2021. URL https://arxiv.org/abs/2109.12098.

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

Austin Stone, Ted Xiao, Yao Lu, Keerthana Gopalakrishnan, Kuang-Huei Lee, Quan Vuong, Paul Wohlhart, Sean Kirmani, Brianna Zitkovich, Fei Xia, Chelsea Finn, and Karol Hausman. Open-world object manipulation using pre-trained vision-language models, 2023. URL https://arxiv.org/abs/2303.00905.

Priya Sundaresan, Suneel Belkhale, Dorsa Sadigh, and Jeannette Bohg. Kite: Keypoint-conditioned policies for semantic manipulation, 2023. URL https://arxiv.org/abs/2306.16605.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

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, 2026. URL https://arxiv.org/abs/2603.03596.

Shiang-Feng Tsai, Jin-Cheng Jhang, Yen-Ling Tai, Jia-Hong Lai, Shih-Yun Wong, KangTung-Hsu, and Yi-Ting Chen. Direct action-head injection of a grounded 3d point unlocks spatial and task generalization, 2026. URL https://arxiv.org/abs/2606.27663.

Runze Wang, Yuqian Fu, Yu Li, Tao Lin, Tianwen Qian, Mohamed Elhoseiny, Bo Zhao, Yanwei Fu, Yu-Gang Jiang, and Xiangyang Xue. Afford-vla: Action-aligned visual planning via internalized affordance, 2026a. URL https://arxiv.org/abs/2605.24203.

Zixuan Wang, Yuxin Chen, Yuqi Liu, Jinhui Ye, Pengguang Chen, Changsheng Lu, Shu Liu, Bei Yu, and Jiaya Jia. Vp-vla: Visual prompting as an interface for vision-language-action models, 2026b. URL https://arxiv.org/abs/2603.22003.

Shijie Wu, Yihang Zhu, Yunao Huang, Kaizhen Zhu, Jiayuan Gu, Jingyi Yu, Ye Shi, and Jingya Wang. Afforddp: Generalizable diffusion policy with transferable affordance, 2025. URL https://arxiv.org/abs/2412.03142.

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

Jonathan Yang, Chuyuan Kelly Fu, Dhruv Shah, Dorsa Sadigh, Fei Xia, and Tingnan Zhang. Bridging perception and action: Spatially-grounded mid-level representations for robot generalization, 2025. URL https://arxiv.org/abs/2506.06196.

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, 2026b. URL https://arxiv.org/abs/2606.20562.

Cheng Yin, Wang Xu, Junpeng Yang, Sikyuen Tam, Hanyu Liu, Yuan Yao, Xiangrui Zeng, Junbo Cui, Yequan Wang, Zhouping Yin, and Yankai Lin. Simplememvla: A simple but effective native-video memory for vision-language-action models, 2026. URL https://arxiv.org/abs/2609.05533.

Hang Yu, Juntu Zhao, Yufeng Liu, Kaiyu Li, Cheng Ma, Di Zhang, Yingdong Hu, Guang Chen, Junyuan Xie, Junliang Guo, Junqiao Zhao, and Yang Gao. Point what you mean: Visually grounded instruction policy, 2026a. URL https://arxiv.org/abs/2512.18933.

Hanyang Yu, Haitao Lin, Jingbo Zhang, Wenyao Zhang, Chenghao Gu, Heng Li, and Ping Tan. Maskwam: Unifying mask prompting and prediction for world-action models, 2026b. URL https://arxiv.org/abs/2606.13515.

Qize Yu, Jiadi You, Yuran Wang, Jiaqi Liang, Bowen Ping, Yang Tian, Yue Chen, Minghong Cai, Zeying Gong, Ruihai Wu, Yinchuan Li, Junwei Liang, and Yingcong Chen. Affordancevla: A vision-language-action model empowering action generation through affordance-aware understanding, 2026c. URL https://arxiv.org/abs/2606.06155.

Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics, 2024. URL https://arxiv.org/abs/2406.10721.

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

Jesse Zhang, Marius Memmel, Kevin Kim, Dieter Fox, Jesse Thomason, Fabio Ramos, Erdem Bıyık, Abhishek Gupta, and Anqi Li. Peek: Guiding and minimal image representations for zero-shot generalization of robot manipulation policies, 2025. URL https://arxiv.org/abs/2509.18282.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies, 2025. URL https://arxiv.org/abs/2412.10345.

A Architecture and Implementation Details

This section details the high-level scheduling and memory-management rules, the Where2Act decoder, and the affordance-conditioned low-level policy. Round 1 and Round 2 share a Qwen3.5-4B high-level backbone in both RoboMME and the real-robot experiments. In both settings, the high-level policy receives front-view observations, while the low-level controller uses $\pi_{0.5}$ as its base policy and receives the task goal, language subgoal, front- and wrist-view observations, and the predicted binary target-affordance map.

A.1 High-Level Scheduling and Memory Management

Let t_h denote the environment timestep of high-level call h. During demonstration observation, only Round 1 is invoked: at the first frame, at 16-frame intervals thereafter, and at the final demonstration frame. During execution, each high-level call runs Round 1 followed by Round 2, after which the low-level policy predicts a 20-action chunk. In RoboMME, the first 16 actions are enqueued and executed, and the remaining four are discarded. On the real robot, all 20 actions are enqueued and executed.

Round 1 follows Equations 1 and 2. Its recent-frame context is

$$\mathcal{C}_h = \left[I^{\mathrm{F}}_{\max(0,t_h-15)}, I^{\mathrm{F}}_{\max(0,t_h-10)}, I^{\mathrm{F}}_{\max(0,t_h-5)}, I^{\mathrm{F}}_{t_h}\right], \tag{10}$$

where I^{F}_t denotes the front-view image at timestep t. For early calls, negative indices in Equation 10 are clamped to zero, so the episode's first frame is repeated. During demonstration observation, Round 1 predicts the updated summary S_h and an optional memory entry m^{new}_h, but omits the language subgoal ℓ_h and retrieval set $\mathcal{J}_h$.

When a new memory entry is generated, Round 1 selects one of the four recent images in Equation 10 and predicts its event caption. All stored entries are exposed to subsequent Round 1 calls through their identifiers and captions; their RGB keyframes are not included in the Round 1 input. During execution, only the keyframes identified by $\mathcal{J}_h$ are loaded and passed to Round 2 together with their captions.

Phase and Temporal Signals. Each high-level call receives a phase signal indicating whether the current observation belongs to demonstration or execution, together with an event signal from $\{\texttt{DEMO_START}, \texttt{DEMO_END}, \texttt{EXECUTION_START}, \texttt{NONE}\}$. The first and final demonstration calls use `DEMO_START` and `DEMO_END`, respectively; the first execution call uses `EXECUTION_START`; all other calls use `NONE`. Each input frame and stored event caption is also tagged with its absolute episode timestep. The absolute timestep attached to each caption provides a temporal reference for determining whether a visually persistent event was already recorded in a previous call, helping suppress duplicate memory entries.

A.2 Where2Act Decoder

We introduce $N_q = 16$ learnable Where2Act token embeddings as additional inputs to Round 2, corresponding to the special tokens $\{\langle \mathrm{WHERE}_i \rangle\}_{i=0}^{N_q-1}$. The shared high-level VLM processes these token embeddings together with the language subgoal ℓ_h, the retrieved keyframe–caption pairs $\mathcal{R}_h$, and the current front-view observation $I_h \equiv I^{\mathrm{F}}_{t_h}$.

As illustrated in Figure 4, after the Round 2 forward pass, we extract the final-layer hidden states at the 16 Where2Act token positions. These hidden states aggregate information from the language subgoal, retrieved historical event evidence, and the current front-view observation, forming a history-conditioned target representation for spatial grounding:

$$\mathbf{H}^{\mathrm{where}}_h \in \mathbb{R}^{16\times d_{\mathrm{HL}}}, \tag{11}$$

where d_{HL} denotes the hidden dimension of the high-level VLM. We project these hidden states into the decoder dimension to obtain the cross-attention memory:

$$\mathbf{K} = \mathbf{V} = \Phi_{\mathrm{where}}\left(\mathbf{H}^{\mathrm{where}}_h\right) = \mathrm{Proj}_{\mathrm{where}}\left(\mathbf{H}^{\mathrm{where}}_h\right) \in \mathbb{R}^{16\times 512}. \tag{12}$$

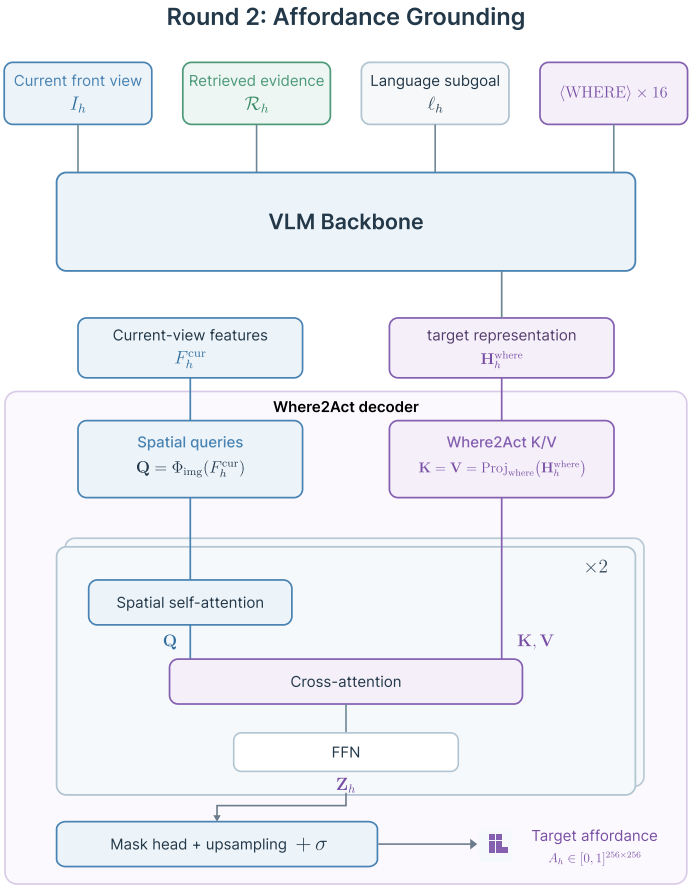

Figure 4: Round 2 affordance grounding and Where2Act decoder. Sixteen history-conditioned Where2Act representations serve as decoder memory, while current-view visual features are transformed into spatial queries for target-affordance prediction.

The spatial queries are constructed from the vision post-merge features of the current front-view image. Let F_h^{cur} denote these features arranged on their native two-dimensional token grid. We resize them to a 32×32 spatial grid, project them to the decoder dimension, apply LayerNorm, add learnable two-dimensional positional embeddings, and flatten the spatial dimensions:

$$\mathbf{Q} = \Phi_{\text{img}}(F_h^{\text{cur}}) = \text{Flatten}\big(\text{LN}\big(\text{Proj}_{\text{img}}(\text{Resize}_{32\times32}(F_h^{\text{cur}}))\big) + P_{\text{2D}}\big) \in \mathbb{R}^{1024\times512}, \tag{13}$$

where $P_{\text{2D}} \in \mathbb{R}^{32\times32\times512}$ is implemented as the sum of learnable row and column positional embeddings. The 1024 queries correspond to the 32×32 spatial locations in the current front view.

We use a two-layer Transformer decoder, denoted by D_{tr}. Each layer applies self-attention over the current-scene spatial queries, followed by cross-attention to the 16 Where2Act memory representations and a feed-forward block. This allows spatial locations in the current observation to exchange contextual information and attend to the history-conditioned target representation. The decoded spatial features are

$$\mathbf{Z}_h = D_{\text{tr}}\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right), \qquad \mathbf{Z}_h \in \mathbb{R}^{1024\times512}. \tag{14}$$

The decoded features are reshaped into a 32×32 feature map and passed through a convolutional upsampling head:

$$L_h = D_{\text{mask}}\left(\text{Reshape}_{32\times32}\left(\mathbf{Z}_h\right)\right), \qquad L_h \in \mathbb{R}^{256\times256}. \tag{15}$$

Together, D_{tr}, the spatial reshape, and D_{mask} constitute the complete Where2Act decoder D_{where} used in the main text.

The corresponding target-affordance map is $A_h = \sigma(L_h) \in [0,1]^{256\times256}$. The spatial-grounding loss and the conversion of A_h into low-level spatial conditioning are detailed in Appendices C.1 and A.3, respectively.

A.3 Affordance-Conditioned Low-Level Modulation

We use $\pi_{0.5}$ as the low-level VLA and apply affordance-conditioned modulation to its front-view visual tokens. At inference time, the Round 2 affordance map is thresholded at $\delta = 0.5$ to obtain a binary target mask:

$$\bar{A}_h(u) = \mathbf{1}\left[A_h(u) \geq \delta\right], \qquad \delta = 0.5, \tag{16}$$

where $\bar{A}_h \in \{0,1\}^{256\times256}$.

The front-view image is processed by the resize-with-padding transform used by $\pi_{0.5}$ to 224×224, producing a 16×16 spatial visual-token grid. We resize the 256×256 binary target mask to 224×224. The resized mask is then partitioned into non-overlapping 14×14 regions aligned with the 256 front-view visual tokens. Let $\mathcal{P}_j$ denote the region corresponding to front-view token j. We compute the average target occupancy within each region and convert it into a soft modulation gate:

$$\begin{aligned} o_j &= \frac{1}{|\mathcal{P}_j|} \sum_{u\in\mathcal{P}_j} \bar{A}_h^{\mathrm{LL}}(u), \\ g_j &= 1 - \exp\left(-\frac{o_j}{\tau}\right), \qquad \tau = 0.05, \end{aligned} \tag{17}$$

where $\bar{A}_h^{\mathrm{LL}} \in \{0,1\}^{224\times224}$ denotes the resized low-level target mask. The resulting gates $g_j \in [0,1]$ are spatially aligned with the front-view visual tokens. The nonlinear mapping produces soft gate values for visual patches that partially overlap the target region, while patches with no target overlap have $g_j = 0$ and are therefore not directly modulated.

Using zero-based layer indexing, we inject these gates at selected PaliGemma VLM layers:

$$\Omega_{\mathrm{mod}} = \{3, 5, 7, 9, 11, 13, 15, 17\}. \tag{18}$$

Let d_{LL} denote the hidden dimension of the low-level VLM, and let $\mathbf{x}_j^{(k)} \in \mathbb{R}^{d_{\mathrm{LL}}}$ denote the front-view token representation at spatial position j in layer k. EMAG modulates this representation as

$$\widetilde{\mathbf{x}}_j^{(k)} = \mathbf{x}_j^{(k)} + g_j \boldsymbol{\beta}^{(k)}, \qquad k \in \Omega_{\mathrm{mod}}, \tag{19}$$

where $\boldsymbol{\beta}^{(k)} \in \mathbb{R}^{d_{\mathrm{LL}}}$ is a learnable layer-specific additive vector. Within each layer, the same $\boldsymbol{\beta}^{(k)}$ is shared across all front-view spatial positions, while the token-specific gate g_j determines the strength of the injected target conditioning. Tokens with greater overlap with the target region therefore receive stronger modulation, whereas tokens with no target overlap remain unchanged.

The resulting VLM representation then conditions the $\pi_{0.5}$ action expert to predict the next 20-action chunk. Execution and replanning schedules follow Appendix A.1.

This feature-level modulation provides an explicit spatial interface between high-level target-affordance prediction and low-level action generation. If the binary target mask $\bar{A}_h$ is empty, then $g_j = 0$ for every front-view token, so Equation 19 reduces to an identity operation. The $\pi_{0.5}$ action expert therefore receives the same VLM context as under the unmodulated pathway for the same visual and language inputs, without introducing a modulation-induced shift in the VLM context.

B Data Construction and Annotation

B.1 High-Level Memory and Retrieval Annotations

We construct the high-level annotation targets from replayed RoboMME demonstrations, following the task definitions and subgoal annotations provided by RoboMME. For each annotated semantic event, we select the first available sampled frame at or after the corresponding event timestep. This frame is assigned a sequential episode-level memory identifier K_i and paired with a concise event caption E_i, forming a memory entry

$$m_i = (K_i, I_{K_i}, E_i)\,. \tag{20}$$

Table 3: An example Round 1 annotation for VIDEOPLACEBUTTON. The retrieved keyframe I_{K_2} records the final placement event preceding the button press.

Task instruction. Place the cube on the target used in the final placement before the button was pressed.

Existing event memory.

K_0: Step0_Demo: episode started.	K_1: Step96_Demo: cube picked up.
K_2: Step192_Demo: cube placed on target A.	K_3: Step262_Demo: button pressed.
K_4: Step384_Demo: cube picked up.	K_5: Step459_Demo: cube placed on target B.

Current annotated state. The robot is in the execution phase and has just picked up the cube.

Progress summary S_h. The demo placed the cube on two targets and pressed the button. Execution: the robot picked up the cube.

Language subgoal ℓ_h.
Place the cube on the correct target.

New memory entry m_h^{new}.
(K_6, I_{K_6}, E_6), where E_6 = "Step523_Exec: cube picked up."

Retrieval target. $\mathcal{J}_h = [K_2]$.

Most such events correspond to subgoal completions, such as picking up an object, placing it on a target, pressing a button, or completing a route segment.

We use GPT-5.6 Sol (OpenAI, 2026) only as an authoring assistant to translate the RoboMME task definitions, subgoal sequences, and temporal dependencies into task-specific annotation rules. These rules are fixed before dataset generation and subsequently executed deterministically offline to produce progress summaries, keyframe captions, language subgoals, and retrieval targets. The language model is not invoked on individual trajectories or training samples. At each high-level decision step, the progress summary describes the task-relevant progress completed so far together with the current phase and execution state. The keyframe caption concisely describes the semantic event associated with the selected frame. During execution, the language subgoal specifies the operation to be performed under the current task state.

The retrieval rule considers the task instruction, current progress summary, and ordered sequence of stored keyframe captions, and returns the identifiers of the memory entries whose historical visual evidence is required by the current decision. The resulting retrieval target is represented as an ordered list of memory identifiers,

$$\mathcal{J}_h = [K_{j_1}, \ldots, K_{j_r}], \qquad K_{j_k} \in \{K_0, \ldots, K_i\}, \qquad k = 1, \ldots, r, \tag{21}$$

where K_i is the identifier of the latest memory entry available at that step and r depends on the current decision. We use $\mathcal{J}_h = []$ when the current decision does not require historical visual evidence. Table 3 illustrates the annotations for one VIDEOPLACEBUTTON decision.

Real-robot data. For the real-robot data, we similarly design deterministic annotation rules using the task definitions, event keyframes, and episode metadata, and apply them offline to generate progress summaries, event captions, language subgoals, and retrieval targets in the same structured format as RoboMME.

B.2 TARGET-AFFORDANCE ANNOTATIONS

RoboMME. We construct binary 256×256 target-affordance masks aligned with the front-view image. For object-centric subgoals, we obtain the target identity from the task-specific subgoal annotation and use simulator segmentation to identify its visible image region. Depending on the subgoal, the target may correspond to an object, receptacle, or interaction region. For the route-target tasks `PatternLock` and `RouteStick`, the annotation represents the remaining route associated with the current subgoal. For `InsertPeg`, when the subgoal is to pick up the peg, the target region corresponds to the designated end of the peg to be grasped rather than the entire peg. If there is no visible spatial target, the annotation is an all-zero mask. Simulator target identities, segmentation labels, and route geometry are used only to construct offline supervision and are not provided to the policy at inference time.

Real-robot data. Each timestep contains front- and wrist-view images, while spatial annotations are generated only for the front view. We use task descriptions, event keyframes, and episode metadata together with task-specific deterministic rules to construct frame-level subgoal intervals and identify the current target at each timestep. For each subgoal interval, we select an initialization frame in which the target is clearly visible and prompt SAM 2.1 Hiera Base+ (Ravi et al., 2024) with a bounding box and one or more positive points. When propagation drifts, we provide additional positive or negative correction points on selected frames. Initialization frames and spatial prompts are obtained from fixed workspace calibration for buttons and receptacles, image-based detection rules for colored blocks, or episode-specific target configurations for visually similar covers. The resulting mask is propagated only within the corresponding subgoal interval, and a new initialization is performed when the subgoal target changes. Task-specific spatial and appearance constraints are applied after propagation to reduce drift toward the robot, human hands, neighboring objects, or background regions. Each mask represents only the visible region of the current target in the front view; occluded regions are not completed, and a fully invisible or inactive target receives an empty mask. At most one target instance is active at each timestep. For BUTTON REPLAY, repeated occurrences of the same button in the demonstrated sequence are treated as separate temporal subgoals. Button color names are used only as offline annotation metadata and are never included in task goals, language subgoals, event captions, or other model inputs. For COUNT PLACE, when multiple blocks share the target color, the annotation corresponds to the block manipulated by the recorded action. The resulting masks are temporally aligned with the RGB observations, language subgoals, event keyframes, progress summaries, and action labels to construct the high- and low-level training datasets. We inspect overlay videos and run automatic consistency checks, revising initialization or correction prompts only when propagation fails.

B.3 DATASET COMPOSITION

We use all 16 tasks from the official RoboMME training split. The original split contains 1,600 episodes; we exclude two episodes that failed during replay, resulting in 1,598 training episodes. Round 1 training data cover both the demonstration and execution phases, allowing the policy to update its progress summary and episodic memory throughout observation and task execution. Round 2 and low-level training data are constructed only from execution timesteps. Round 2 supervises target-affordance prediction in the current view, whereas the low-level data supervise action generation conditioned on the corresponding language subgoal and affordance map.

For the real-robot experiments, we initially collect 100 training episodes for each of COUNT PLACE, BLOCK UNMASK, and BUTTON REPLAY. We exclude three low-quality episodes, resulting in 297 episodes in total. All learned policies use this filtered dataset. We combine the episodes across the three tasks into a multi-task dataset and train the high- and low-level policies separately.

Event-memory and retrieval annotations are described in Appendix B.1, and target-affordance annotations are described in Appendix B.2. The real-robot platform, task definitions, scene randomization, and evaluation protocol are detailed in Appendix D.

C TRAINING, EVALUATION, AND BASELINE IMPLEMENTATION

C.1 EMAG TRAINING

We train the high- and low-level policies separately using the data described in Appendix B.3. Table 4 summarizes the principal training configurations for RoboMME and the real-robot experiments.

High-level policy. For RoboMME, we initialize the high-level policy from Qwen3.5-4B and jointly train Round 1 and Round 2 across all 16 tasks for 80,000 steps. For the real-robot experiments, we initialize from Qwen3.5-4B and jointly optimize Round 1 and Round 2 on the combined three-task dataset for 40,000 steps. In both settings, the high-level policy receives only front-view images. All high-level images are processed using the image processor provided with the Qwen3.5 checkpoint, with its default dynamic-resolution settings. The same preprocessing is used during training and evaluation.

We freeze the vision encoder and fine-tune the remaining backbone parameters together with the Where2Act components. Round 1 and Round 2 training examples are randomly interleaved. Round 1 is optimized with teacher forcing over its autoregressive outputs, while Round 2 directly supervises the Where2Act mask prediction.

Training uses AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon_{\text{AdamW}} = 10^{-8}$, together with BF16 precision, task-balanced sampling, and a global batch size of 16 on 8 $\times$ H100 GPUs. The high-level backbone uses a learning rate of 10^{-5}, while the newly initialized Where2Act components use 10^{-4}. The latter parameter group includes the 16 learnable Where2Act token embeddings, the visual and Where2Act projections, learned two-dimensional positional embeddings, the spatial Transformer decoder, and the dense prediction head. We use a constant learning rate without warmup, zero weight decay, a maximum gradient norm of 1.0, and a maximum input sequence length of 4,096 tokens. These optimization settings are shared by the RoboMME and real-robot high-level policies. All reported evaluations use the final training checkpoint.

During inference, Round 1 uses deterministic greedy decoding with at most 384 newly generated tokens. Round 2 performs no autoregressive decoding and directly predicts the target-affordance map.

We optimize the high-level policy using the objective defined in Section 3.4. The retrieval term is implemented as an auxiliary token-level cross-entropy loss on the retrieved-memory-ID positions in the Round 1 autoregressive output, without introducing a separate retrieval head. The Where2Act architecture is detailed in Appendix A.2, and its spatial supervision is described in Appendix B.2.

Where2Act Loss. Let $z_b(u)$ denote the predicted affordance logit at pixel u for training sample b, let $p_b(u) = \sigma(z_b(u))$, and let $y_b(u) \in \{0, 1\}$ denote the ground-truth target mask. We compute foreground and background binary cross-entropy terms separately and normalize each by the corresponding number of pixels:

$$\mathcal{L}_{\text{pos}}^{(b)} = \frac{\sum_u y_b(u)\,\text{softplus}(-z_b(u))}{\max(\sum_u y_b(u), \epsilon)}, \tag{22}$$

$$\mathcal{L}_{\text{neg}}^{(b)} = \frac{\sum_u (1 - y_b(u))\,\text{softplus}(z_b(u))}{\max(\sum_u (1 - y_b(u)), \epsilon)}, \tag{23}$$

where $\epsilon = 10^{-6}$. The class-balanced BCE assigns equal weight to the foreground and background terms:

$$\mathcal{L}_{\text{bal-BCE}}^{(b)} = \frac{1}{2}\mathcal{L}_{\text{pos}}^{(b)} + \frac{1}{2}\mathcal{L}_{\text{neg}}^{(b)}. \tag{24}$$

The soft Dice loss is

$$\mathcal{L}_{\text{Dice}}^{(b)} = 1 - \frac{2\sum_u p_b(u) y_b(u) + \epsilon}{\sum_u p_b(u) + \sum_u y_b(u) + \epsilon}. \tag{25}$$

Only Round 2 samples with valid spatial supervision contribute to the spatial-grounding objective. Let $\mathcal{V}$ denote the set of such samples in a batch. The loss is averaged over $\mathcal{V}$:

$$\mathcal{L}_{\text{where}} = \frac{1}{|\mathcal{V}|} \sum_{b \in \mathcal{V}} \left(\mathcal{L}_{\text{bal-BCE}}^{(b)} + \mathcal{L}_{\text{Dice}}^{(b)} \right). \tag{26}$$

If $\mathcal{V}$ is empty, we set $\mathcal{L}_{\text{where}} = 0$. The foreground and background BCE weights are both 0.5, and the Dice-loss coefficient is 1.0. In Equation 9, we set $\lambda_1 = 2$ for retrieval and $\lambda_2 = 1$ for spatial grounding.

Low-level policy. We initialize the low-level policy from the public $\pi_{0.5}$ base checkpoint. During training, the policy is conditioned on front- and wrist-view images, the task goal, language subgoal, and a ground-truth target-affordance map. Expert action chunks provide supervision for the flow-matching action objective. We freeze the vision encoder and optimize the remaining policy parameters with AdamW and BF16 precision on 8 $\times$ H100 GPUs. For the real-robot experiments, the policy additionally receives the current 7-dimensional proprioceptive state, corresponding to the six arm joints and the gripper.

For RoboMME, the low-level policy is trained for 80,000 steps with a global batch size of 64. We linearly warm up the learning rate to 5×10^{-5} over the first 10,000 steps and keep it constant

Table 4: EMAG training configurations. All policies use AdamW, BF16 precision, and 8 × H100 GPUs. High-level learning rates list the backbone and Where2Act parameter groups, respectively. Maximum token length refers to the maximum input sequence length for the high-level policy and the maximum language-token length for the low-level policy.

Setting	RoboMME HL	RoboMME LL	Real-robot HL	Real-robot LL
Initialization	Qwen3.5-4B	$\pi_{0.5}$ base	Qwen3.5-4B	$\pi_{0.5}$ base
Camera inputs	Front	Front + wrist	Front	Front + wrist
Frozen module	Vision encoder	Vision encoder	Vision encoder	Vision encoder
Training budget	80k steps	80k steps	40k steps	40k steps
Global batch size	16	64	16	64
Learning rate	$10^{-5}/10^{-4}$	5×10^{-5}	$10^{-5}/10^{-4}$	5×10^{-5}
LR warmup	0	10k steps	0	10k steps
Maximum token length	4,096	128	4,096	128
Action horizon	—	20	—	20

thereafter. We use gradient clipping at 1.0 and an EMA decay of 0.999. The maximum language-token length is 128, and the action horizon is 20. Action chunks are constructed so that they do not cross subgoal boundaries. For the real-robot experiments, we use the same optimization settings and train on the combined three-task dataset for 40,000 steps.

The target-affordance conditioning mechanism is detailed in Appendix A.3. During training, ground-truth affordance masks are used as the spatial conditioning signal, with the augmentation described below. At inference time, they are replaced by the target-affordance maps predicted by the high-level policy.

Mask augmentation. To improve robustness to imperfect high-level predictions, we augment the ground-truth target-affordance masks during low-level training. Each training mask is independently assigned one of four variants: unchanged, translated, dilated, or eroded, with probabilities 0.65, 0.18, 0.12, and 0.05, respectively. Translation offsets are independently sampled along both image axes from $\mathcal{N}(0, 2)$, rounded to the nearest integer, and clipped to $[-4, 4]$ pixels. Dilation uses a disk-shaped structuring element with radius 1, 2, or 3, sampled with probabilities 0.50, 0.35, and 0.15, respectively. Erosion uses a disk-shaped structuring element with radius 1.

C.2 RoboMME Evaluation Protocol

We evaluate EMAG on all 16 tasks in RoboMME (Dai et al., 2026). We evaluate three independently trained EMAG policies with different random seeds, using 50 episodes per task and seed, resulting in 150 rollouts per task. Except for PonderPounce, the baseline results reported in the main paper are taken from the official RoboMME report; the PonderPounce results are taken from its original paper. Each episode is limited to 1,300 environment steps. We follow the official RoboMME task-success criteria, early-termination rules, and evaluation-scene distributions. Evaluation uses synchronous inference. At the transition from demonstration to execution, we use the simulator-provided phase-transition signal only to indicate that the demonstration has ended and robot execution should begin. This signal does not reveal intermediate subgoal completion, target identity, or other task-state information. High-level invocation, demonstration observation, action execution, and replanning schedules are detailed in Appendix A.1.

C.3 Real-Robot Baseline Implementation

For each baseline, we train a separate multi-task policy across Count Place, Block Unmask, and Button Replay using the same 297 real-robot episodes as EMAG. At each timestep, both policies receive the task goal, current front- and wrist-view images, and the current 7-dimensional proprioceptive state.

Fine-tuned $\pi_{0.5}$. We initialize the policy from the public $\pi_{0.5}$ base checkpoint (Intelligence et al., 2025) and fine-tune it for 40,000 steps using the same optimization settings as the EMAG low-level policy. The policy uses only the task goal and current observation and does not receive a language subgoal, episodic memory, or target-affordance map.

FrameSamp+Modul. We initialize the policy from the same public $\pi_{0.5}$ base checkpoint and train it for 40,000 steps following the official FrameSamp+Modul configuration (Dai et al., 2026). Unlike the fine-tuned $\pi_{0.5}$ baseline, FrameSamp+Modul additionally conditions the policy on temporal visual memory constructed from uniformly sampled front-view frames from the observed episode history. We use the official 512-token memory budget; the sampled historical features are integrated into the policy through the official modulation mechanism. The policy does not receive a language subgoal or target-affordance map, and the vision encoder is frozen following the official implementation.

C.4 Ablation Implementation

All three configurations use three independent seeds and are evaluated on the same 50 episodes per task and seed. Full EMAG-4 trains both policies. Without retrieval retrains the high-level policy and reuses the seed-matched Full EMAG-4 low-level checkpoint, whereas affordance as context reuses the seed-matched Full EMAG-4 high-level checkpoint and retrains the low-level policy. For high-level training, we apply $4\times$ subsampling and retain every fourth training sample.

Without retrieval. This variant modifies only the high-level policy while keeping the low-level policy unchanged by reusing the seed-matched Full EMAG-4 checkpoint. Round 1 continues to update the event memory and predict the language subgoal, but no longer predicts retrieved memory identifiers. Accordingly, we remove the retrieval-token loss. At each high-level decision, Round 2 receives all previously stored image–caption pairs in chronological order instead of a retrieved subset. All other high-level components and training settings remain the same as Full EMAG-4.

Affordance as context. This variant reuses the seed-matched Full EMAG-4 high-level checkpoint and modifies only the low-level affordance-conditioning mechanism. The single-channel affordance map is replicated across three channels and supplied to the low-level policy as a third visual input alongside the front and wrist views. The affordance input is processed by the same vision encoder as the camera observations. We remove token-level feature modulation and retrain the low-level policy using the same training budget as Full EMAG-4. The training-time ground-truth masks and inference-time predicted masks are otherwise identical to those used by Full EMAG-4.

D Real-Robot Experimental Details

D.1 Robot Platform and Sensing Setup

Figure 5 shows the real-robot platform and the three physical tasks. We use an AgileX PiPER single-arm robot equipped with its integrated gripper. A fixed Intel RealSense L515 camera provides the front-view observation, while an Intel RealSense D405 camera mounted near the robot end effector provides the wrist-view observation. Both cameras capture 640×480 images at 20 Hz.

The high-level policy receives only the front-view images, whereas the low-level policy receives both front- and wrist-view images. The robot is controlled at 20 Hz using absolute joint-angle commands together with a gripper command. Action execution and high-level replanning follow the schedules specified in Appendix A.1.

All real-robot inference runs on a single NVIDIA RTX 5090 GPU with 32 GB of memory. The blocks, receptacle, covers, and buttons are placed within a fixed workspace and arranged or randomized before each episode according to the corresponding task definition.

D.2 Real-World Task Definitions and Randomization

We design three real-robot tasks that respectively test accumulated progress tracking, history-dependent target memory under occlusion, and ordered event memory with current-scene visual



Figure 5: Real-robot platform and tasks. (a) The AgileX PiPER single-arm robot with its integrated gripper, a wrist-mounted Intel RealSense D405 camera, and a fixed Intel RealSense L515 front-view camera. (b–d) The Count Place, Button Replay, and Block Unmask tasks, respectively.

grounding. For each task, we construct 20 evaluation configurations and use the same configurations for all learned policies and the human evaluation.

Count Place. Each episode contains five or six green and yellow blocks randomly placed on the table. The task instruction specifies a target color and a target count sampled from one to three. Blocks of the other color serve as visual and manipulation distractors. Block positions are randomized across episodes, while the receptacle and completion button remain within fixed regions of the workspace. The robot must track the number of completed placements and press the completion button after placing the instructed number of target-colored blocks into the receptacle.

Block Unmask. Each episode contains three or four visually similar covers, with the cover arrangement, occluded blocks, and workspace positions randomized across episodes. An episode may contain one or more target blocks. From the demonstration, the robot must remember the association between each target block and its corresponding cover, and then lift all corresponding covers during execution.

Button Replay. The scene contains four buttons at fixed spatial locations. The robot first observes a human demonstration consisting of three to six button presses and must subsequently reproduce the complete sequence in the same order. A button may appear multiple times within a sequence. Consecutive repetitions of the same button are excluded. The language conditioning provided during training and evaluation does not contain button colors or explicit button-identity labels. The policy must therefore infer the target location from the visual events observed during the demonstration.

D.3 Trials, Success Criteria, and Human Teleoperation Protocol

For each learned policy, we evaluate three independently trained policy seeds. Each seed is evaluated once on the same set of 20 task configurations, resulting in 60 trials per method and task. All learned

policies use identical task configurations and initial-state assignments to reduce variation caused by differences in scene setup.

Each trial is limited to 1,200 robot-control steps. A trial is marked as a failure if the task is not completed within this horizon. After an ordinary grasp failure, the policy may retry within the remaining control horizon. However, grasping an incorrect block, lifting an incorrect cover, or pressing an incorrect button immediately causes the trial to fail. No human intervention or manual retry is allowed during policy execution. Trials invalidated by hardware, communication, or control-system failures are excluded and rerun using the same task configuration. Success and failure are manually verified from the recorded robot executions.

Count Place succeeds only when the robot places the correct number of blocks of the instructed color into the receptacle and subsequently presses the completion button. Placing too many or too few blocks, placing a block of the wrong color, or pressing the button before task completion constitutes a failure. Block Unmask requires the robot to lift all and only the covers associated with the target blocks; omitting any target cover or lifting a distractor cover constitutes a failure. Button Replay requires the robot to reproduce the complete demonstrated sequence in the correct order, and any incorrect button press constitutes a failure.

Human Performance uses the same robot platform, 20 task configurations, and success criteria. For tasks with a demonstration phase, the human operator first observes the same demonstration provided to the learned policies and then controls the robot through a master–slave teleoperation system: the operator moves the master arm, and the PiPER slave arm executes the corresponding motion. During execution, the human operator directly observes the robot workspace while controlling the slave arm. A second operator arranges the scene and, when applicable, constructs the demonstration for each episode. The same human operator completes three evaluation rounds over the 20 configurations for each task, resulting in 60 human trials per task.

For learned policies, we report the average success rate and sample standard deviation across the three policy seeds. The three human evaluation rounds share the same operator and task configurations and are therefore not described as independent seeds. Human error bars denote the standard deviation across the three repeated evaluation rounds.

E Additional Diagnostic Analyses

We separately evaluate the retrieval and spatial-grounding capabilities of the high-level policy on held-out RoboMME episodes. Round 1 and Round 2 are evaluated independently using the corresponding annotations, isolating the performance of each component from errors accumulated during closed-loop execution.

E.1 Event-Retrieval Evaluation

For each sampled Round 1 call, we compare the predicted and annotated memory-ID lists. Precision, recall, and F1 are micro-averaged over memory IDs, while exact match requires the predicted list to exactly match the annotated list. For tasks in which none of the sampled states requires memory retrieval, precision, recall, and F1 are reported as dashes; these states are still included in exact-match accuracy. As shown in Table 5, event retrieval is nearly saturated, achieving 99.90% precision, 99.42% recall, 99.66% F1, and 99.69% exact match overall. This indicates that retrieval itself is highly reliable when evaluated independently of errors accumulated during closed-loop execution.

E.2 Spatial-Grounding Evaluation

For each sampled Round 2 call, we provide the annotated language subgoal and retrieved memory and evaluate the predicted target-affordance map. IoU and Dice are computed on samples with non-empty annotated target masks.

As shown in Table 5, EMAG achieves sample-weighted mean IoU and Dice scores of 65.94% and 76.19%, respectively. Grounding is strongest for tasks whose targets correspond directly to visible object regions, such as ButtonUnmask, StopCube, and VideoUnmask. Some lower IoU values, however, do not directly correspond to incorrect task-level target selection. In BinFill,

Table 5: Single-step diagnostic results on held-out RoboMME episodes. Retrieval metrics compare predicted and annotated memory-ID lists, while grounding metrics evaluate predicted target-affordance maps on samples with non-empty annotated target masks. All values are reported in percent. Dashes indicate tasks in which none of the sampled states requires memory retrieval.

	Event Retrieval				Spatial Grounding	
Task	Precision	Recall	F1	Exact	IoU	Dice
BINFILL	—	—	—	100.00	49.95	58.41
BUTTONUNMASK	100.00	100.00	100.00	100.00	81.95	89.71
BUTTONUNMASKSWAP	100.00	100.00	100.00	100.00	73.23	81.75
INSERTPEG	100.00	100.00	100.00	100.00	54.48	63.88
MOVECUBE	100.00	100.00	100.00	100.00	64.84	76.67
PATTERNLOCK	100.00	98.55	99.27	98.75	56.24	69.68
PICKHIGHLIGHT	100.00	96.23	98.08	98.75	73.67	82.58
PICKXTIMES	—	—	—	100.00	72.05	82.62
ROUTESTICK	98.57	98.57	98.57	98.75	44.64	61.30
STOPCUBE	—	—	—	100.00	82.23	90.17
SWINGXTIMES	—	—	—	100.00	76.09	84.94
VIDEOPLACEBUTTON	100.00	100.00	100.00	100.00	66.60	77.94
VIDEOPLACEORDER	100.00	100.00	100.00	100.00	65.18	76.59
VIDEOREPICK	100.00	97.73	98.85	98.75	54.17	64.76
VIDEOUNMASK	100.00	100.00	100.00	100.00	76.09	85.31
VIDEOUNMASKSWAP	100.00	100.00	100.00	100.00	64.34	72.85
Overall	99.90	99.42	99.66	99.69	65.94	76.19

multiple same-colored cubes can be semantically valid targets, while the annotation is tied to the specific instance selected during simulator replay; predicting another valid same-colored cube is therefore penalized by pixel-level overlap metrics.

PATTERNLOCK and ROUTESTICK pose a different challenge: their target affordances represent routes that are not directly visible as target regions in the current RGB observation and must instead be inferred from task state and history in the current image coordinates. These cases make exact pixel-level overlap substantially more demanding. Overall, the diagnostic results show that event retrieval is highly reliable under independent evaluation, while spatial grounding exhibits substantially more residual error.

F REAL-ROBOT FAILURE ANALYSIS

We qualitatively inspect representative failure cases from the real-robot rollouts of the three evaluated policies. These examples are selected to illustrate distinct failure mechanisms rather than their empirical frequencies. We organize the analysis by policy; within each policy card, groups (a), (b), and (c) correspond to COUNT PLACE, BLOCK UNMASK, and BUTTON REPLAY, respectively.

Fine-tuned $\pi_{0.5}$. The fine-tuned $\pi_{0.5}$ baseline receives current observations and the task goal, but no episodic memory or language subgoal. It is therefore not equipped to reliably resolve the history-dependent decisions required by these tasks. In COUNT PLACE, in addition to imprecise grasps, the policy may stall after successfully picking up a target block (Figure 6(a.1)) or press the completion button before placing the instructed number of blocks (Figure 6(a.2)). In BLOCK UNMASK, grasp slippage can prevent a selected cover from being lifted (Figure 6(b.1)). More fundamentally, without memory of the demonstration, the policy cannot reliably infer which cover conceals the target block and may select an incorrect cover (Figure 6(b.2)). In BUTTON REPLAY, it presses buttons without preserving the demonstrated sequence (Figure 6(c.1)), reflecting the absence of ordered event memory rather than a single local control error.

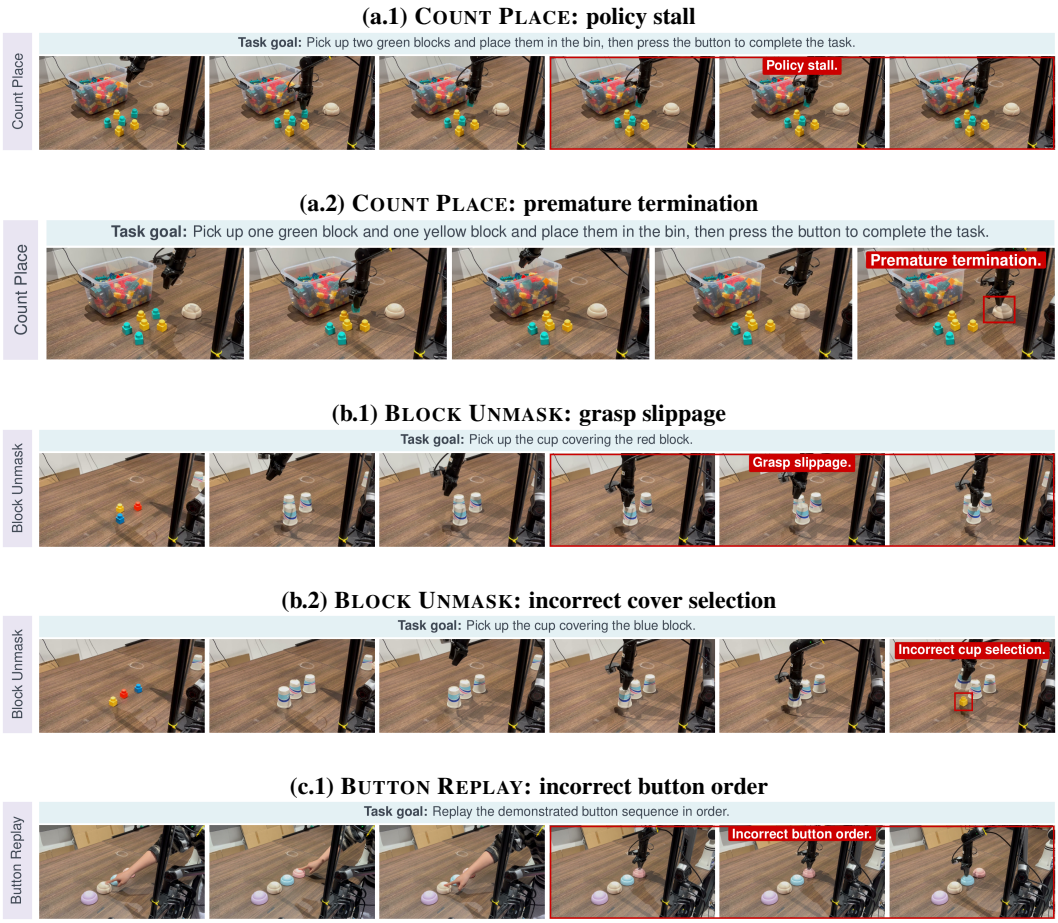

Figure 6: Representative real-robot failures of the fine-tuned $\pi_{0.5}$ policy across the three tasks.

FrameSamp+Modul. FrameSamp+Modul has access to sampled visual history, but the illustrated failures show that retaining history does not always yield stable target selection or execution progress. In COUNT PLACE, the failures are primarily low-level: the policy does not establish a stable grasp during the first or a subsequent pickup (Figure 7(a.1–a.2)). In BLOCK UNMASK, it may lose the selected cover through grasp slippage (Figure 7(b.1)) or select an incorrect nearby cover (Figure 7(b.2)), suggesting unstable spatial disambiguation of the remembered target. In the illustrated BUTTON REPLAY failures, the policy executes the demonstrated button sequence in an incorrect order (Figure 7(c.1–c.2)). These cases are consistent with difficulty aligning sampled history with the current execution step.



Figure 7: Representative real-robot failures of FrameSamp+Modul across the three tasks.

EMAG. In the illustrated EMAG failures, target selection is generally correct, while the observed errors arise primarily during execution or event updating. In COUNT PLACE, EMAG may miss the target-block grasp (Figure 8(a.1)) or release a successfully grasped block before reaching the receptacle (Figure 8(a.2)). In BLOCK UNMASK, EMAG selects the correct target cover but can still fail because the grasp slips during lifting (Figure 8(b.1)). BUTTON REPLAY reveals a different closed-loop failure. Although EMAG initially follows the demonstrated sequence correctly, it can remain near a button after completing the press instead of promptly transitioning to the next one. The same completed press is then recorded repeatedly as new event keyframes, causing the progress state to advance more than once. This shifts the sequence index and causes the subsequent action to skip the next required button (Figure 8(c.1)). The resulting omission is therefore an event-update synchronization failure rather than an error in initial sequence recall.

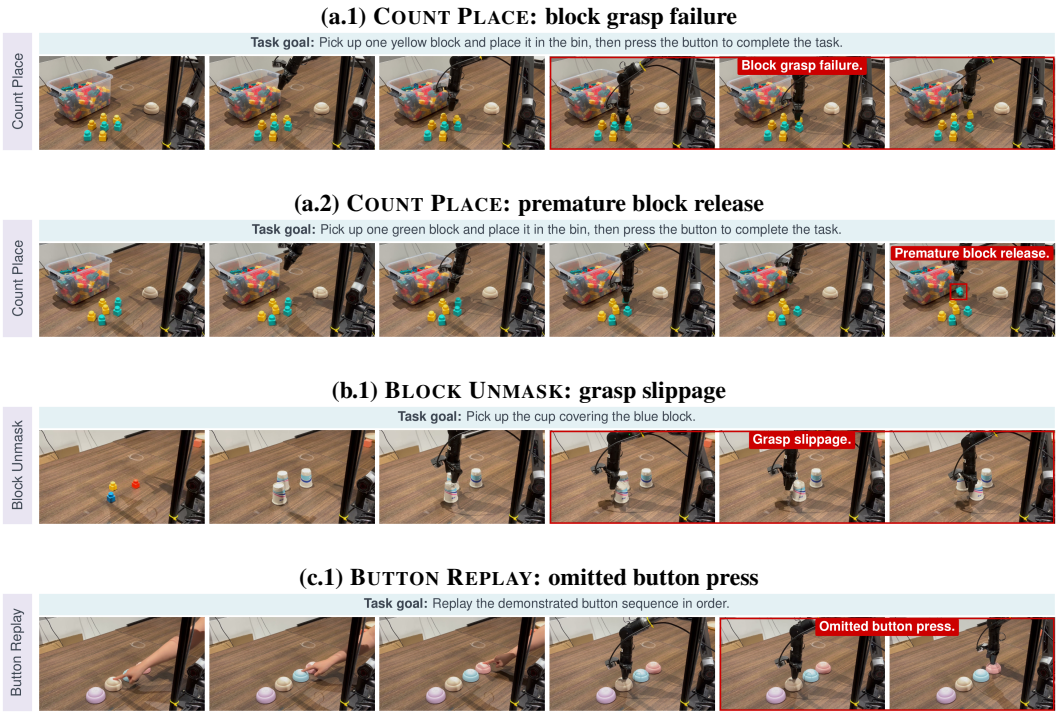

Figure 8: Representative real-robot failures of EMAG across the three tasks.