LoopVLA: cross-subtask event memory for looped task execution

Anonymous authors
Paper under double-blind review

Abstract

Repetitive tasks are common in real-world manipulation, yet current vision-language-action (VLA) policies remain unreliable for repeated actions. We present LoopVLA, a cross-subtask event-memory interface for looped task execution. Its central idea is to compress a just-completed subtask into memory for the next subtask's action expert, making completed execution available for subsequent repetition and stopping decisions. Learnable event latents query the closed segment's tokens, while action features query history and joint history–event context; the two action-side readouts are residually fused and injected directly into the action head. On a memory benchmark, LoopVLA achieves 80.22% success on the repetition task group with online subgoal prediction, bringing evaluation closer to realistic deployment while improving overall success over strong memory baselines.

1 Introduction

Vision-Language-Action (VLA) policies such as RT-2 (Brohan et al., 2023), OpenVLA (Kim et al., 2024), $\pi_0/\pi_{0.5}$ (Black et al., 2024; Physical Intelligence et al., 2025), and GR00T N1.x (NVIDIA et al., 2025; NVIDIA GEAR Team, 2025) generate manipulation actions from visual observations, robot state, and task instructions. Active perception (Liu et al., 2026) and visual grounding with spatial reasoning (Liu et al., 2025c;d) improve scene understanding, yet the next action also depends on completed execution: visually similar states may call for another repetition or for stopping.

This ambiguity is central to *history-dependent manipulation*. A packaging station must place the requested number of items, an assembly cell must repeat an operation across parts, and an inspection fixture must stop an object after a prescribed number of movements. In such looped tasks, successful execution requires both performing each subtask and deciding when its repetitions are complete. The resulting *cross-subtask dependency* calls for memory that transfers completed execution to the next action decision while retaining access to earlier progress.

Recent memory-augmented VLAs address temporal context through multi-observation summaries and visual traces (Koo et al., 2025; Zheng et al., 2024; Fang et al., 2025; Liu et al., 2025a), semantic or episodic memory and stage retrieval (Torne et al., 2026; Sridhar et al., 2025; Li et al., 2025), and recurrent or query-based summaries (Shi et al., 2025; Li et al., 2026). These approaches differ in what they store, when they update or access it, and where it conditions control. Subgoal control (Yang et al., 2025; Chen et al., 2026a; Han et al., 2025) and keyframe-based history (Mark et al., 2026; Chen et al., 2026b) further motivate event-level temporal abstraction. We study a specific cross-subtask interface: at a subgoal transition, learnable event latents compress the most recently closed execution segment into memory that the next subtask's action expert reads alongside historical tokens.

We identify a key design principle: **subtask transitions are natural moments to organise recently completed execution for the next stage.** Sampled perceptual history retains earlier context, while an event summary represents the most recently closed segment. The next subtask's action expert reads these complementary memories to inform continuing, repeating, or stopping.

Based on this principle, we present **LoopVLA**, a cross-subtask event-memory interface for looped task execution. In the compression stage, *learnable event latents* Q^e serve as queries (Li et al., 2023), and the closed segment's event tokens provide keys and values. Attention aggregation produces content-dependent event memory m_{k-1}. In the action-reading stage, action hidden states separately query historical tokens and the joint history–event context containing m_{k-1}. The two readouts un-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

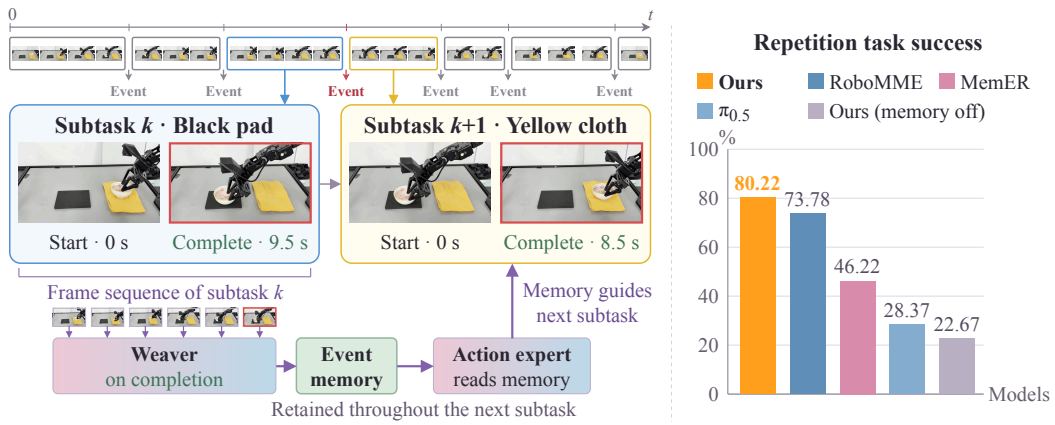

Figure 1: **Memory connects completed subtasks to subsequent actions.** (a) A timeline of a robotic task. Blue segments are compressed into event memory to guide actions in the yellow segment; events trigger memory compression. (b) Results on repeated tasks.

dergo residual fusion before the fused context is injected through MemoryRMS into the $\pi_{0.5}$ action expert. The task instruction ℓ remains a policy input; subsequent-subtask action supervision and auxiliary subgoal alignment train the memory to preserve information useful for execution.

Figure 1 illustrates this memory hand-off and compares success on repeated tasks (in panel (b), Ours' results use online Qwen-VL subgoals to approximate a real-world setting). RoboMME (Frame-Samp+Modul) (Dai et al., 2026) is the baseline shown in panel (b). Table 1 compares the best-performance configuration with strong memory baselines. Across all 16 tasks, LoopVLA achieves 47.17% success with Qwen-VL subgoals and 47.71% with Oracle subgoals, compared with 44.51% for RoboMME (best), the strongest published baseline in this comparison. On the repetition task group—PickXtimes, SwingXtimes, and StopCube—the corresponding rates are 80.22%, 79.78%, and 73.78%; on StopCube, they are 49.33%, 48.67%, and 42.00%. Reference-configuration analyses further examine window organisation, history coverage, repetition-count and difficulty strata, and paired episode outcomes. Both subgoal protocols are formal execution evaluations: Oracle uses simulator annotations, while Qwen-VL predicts subgoals online to bring evaluation closer to realistic deployment.

Our contributions are:

1. We identify subtask completion as a natural, task-aligned point for memory compression, highlighting the *when* dimension of cross-subtask hand-off. Each self-contained summary directly conditions the next subtask's action expert, specifying *where* memory is consumed.
2. We propose LoopVLA, whose compact Weaver summarises completed subtasks into event memory, defining *what* to pass forward. Action-driven reading and residual fusion combine this memory with history for direct injection into the action expert.
3. On a memory benchmark, LoopVLA uses online subgoal prediction to approximate a real-world setting and achieves 80.22% aggregate success on the repetition task group, versus 22.67% for the memory-off configuration. This demonstrates effective repeated execution with predicted event triggers.

2 Related Work

2.1 Memory-augmented VLA policies

Memory-augmented VLA policies differ along three axes: *what* is stored, *where* it is consumed, and *when* it is updated or accessed. Prior work stores visual histories or overlays (Koo et al., 2025; Zheng et al., 2024; Fang et al., 2025; Liu et al., 2025a), semantic traces or retrieved episodes (Torne et al.,

2026; Sridhar et al., 2025; Li et al., 2025), and recurrent/query summaries (Shi et al., 2025; Li et al., 2026); these memories are consumed through prompt/input augmentation (Li et al., 2025; Jang et al., 2025; Fan et al., 2025), planning or hierarchical control (Sridhar et al., 2025; Zhang et al., 2025b). LoopVLA targets the 'what, where and when' by compressing the most recently closed execution segment into event memory supplied directly to the next sub-task's action expert.

Along the *when* axis, observation-level updates (Koo et al., 2025), retrieval of demonstration-time stages (Li et al., 2025), and rollout-time sub-goal events organise memory access differently. Completion-aware execution (Yang et al., 2025; Chen et al., 2026a), keyframe-based history (Mark et al., 2026; Chen et al., 2026b), hierarchical sub-goal control (Han et al., 2025), and skill segmentation in imitation learning (Kipf et al., 2019; Zhu et al., 2022; Wan et al., 2024) provide related temporal abstractions. In LoopVLA, an observed sub-goal transition closes the current execution segment; its event memory is then passed to the next sub-task's action expert.

2.2 Query-bottleneck memory modules

Learnable query bottlenecks compress variable-length context into fixed-size representations; BLIP-2's Q-Former (Li et al., 2023) uses learnable queries to bridge frozen vision and language modules. Related representation work includes hierarchical robot control (Zhang et al., 2025b), cross-segment recurrent memory (Bulatov et al., 2022), and latent sub-segments in imitation learning (Kipf et al., 2019). Our trigger–weaver factorisation is inspired by MemGen (Zhang et al., 2025a), whose trigger decides when to invoke a latent-memory weaver during LLM reasoning. LoopVLA adapts this division of roles to robot control: sub-goal transitions close execution segments, whose event tokens provide keys and values, while learnable event latents serve as queries. Attention pooling produces content-dependent event memory for the next sub-task's action expert; Section 3.4 gives the formulation.

2.3 Task-structure priors and repetition counting

Task-structure methods track progress through reward-machine states (Toro Icarte et al., 2018) or LTL formula progression in instruction-conditioned policies (Vaezipoor et al., 2021). LoopVLA instead represents the closed segment with learned event memory shaped by subsequent-action prediction and sub-goal alignment. Perception-side work studies repetition counting in video (Dwibedi et al., 2020; Hu et al., 2022; Sinha et al., 2024) and object counting in vision-language models (Paiss et al., 2023; Guo et al., 2025). LoopVLA addresses the complementary control question of how past execution should condition the next decision to repeat, proceed, or stop. Its learnable event latents produce a representation for subsequent control, without assuming that this representation encodes an exact, decodable repetition count.

3 LoopVLA architecture

3.1 Overview

LoopVLA targets ordered sub-tasks $\mathcal{E} = (\tau_0, \ldots, \tau_{K-1})$ whose actions depend on earlier execution. The VLA predicts an action chunk (Zhao et al., 2023; Liu et al., 2025b) from observation o_t, state s_t, and instruction ℓ, augmented with perceptual history $F_t = \text{History}(o_{0:t})$ and the preceding segment's summary. Learnable event latents Q^e query segment event tokens Z_{k-1} to produce the compressed output m_{k-1}. The joint memory is $M_t = [F_t \| m_{k-1}]$. At each policy query,

$$\begin{aligned} m_{k-1} &\leftarrow \begin{cases} \text{Weaver}(Z_{k-1}; Q^e, \ell), & \text{on an observed event transition,} \\ m_{k-1}, & \text{otherwise,} \end{cases} \\ \hat{a}_t &\leftarrow \text{ActionExpert}(o_t, s_t, \ell; \underbrace{M_t}_{\text{History}(o_{0:t})\|m_{k-1}}). \end{aligned} \tag{1}$$

Action hidden states separately read F_t and joint memory M_t; residual fusion supplies the context for MemoryRMS (Section 3.5 and Figure 2). The visual branches in Figure 2 use SigLIP (Zhai et al., 2023).

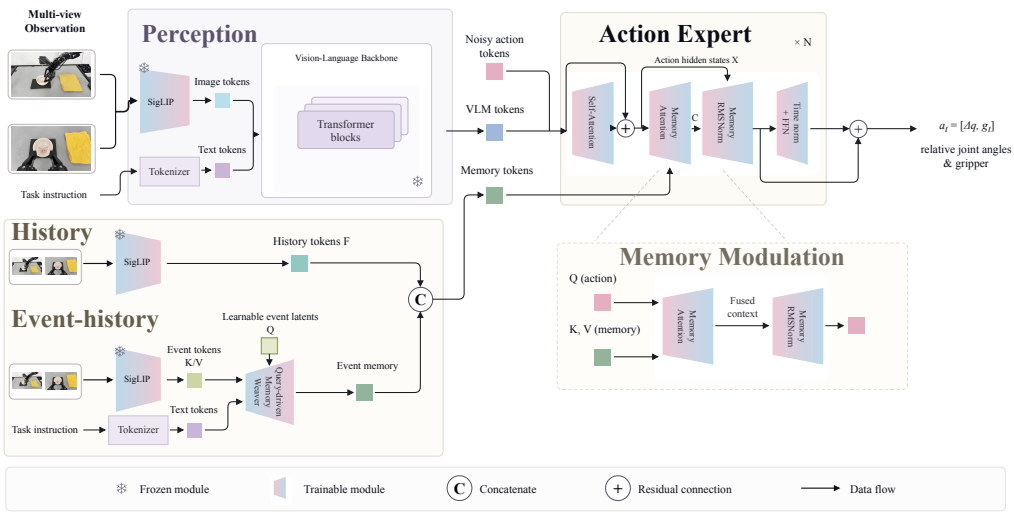

Figure 2: **History and event memory are consumed by the action expert.** Learnable event latents query segment event tokens (K/V) to form event memory. Action hidden states read history and joint history--event context; residual fusion and output projection supply the context injected through MemoryRMS. Each square denotes a token set; snowflakes denote frozen modules.

The event memory is initially empty; closing a segment updates m_{k-1}, which remains available throughout the next segment τ_k. History F_t advances with execution, and current action hidden states initiate each read. The available summary can therefore yield different readouts at successive actions.

3.2 Perceptual History

We uniformly sample up to H observations across the executed history through the current timestep; encoding and projection produce F_t. The main memory setting is 32 history observations + 32 event latents ($H = M = 32$). Appendix A describes event inputs, encoding, and padding.

3.3 Sub-goal-event trigger

A change in the sub-goal label supplied to the policy closes the preceding segment and updates event memory. Let g_q denote this label at query q. Queries occur every 16 executed actions, so an event is an observed change in g_q. This defines the segment for memory hand-off, complementing observation-level summaries (Koo et al., 2025) and demonstration-stage retrieval (Li et al., 2025). Appendix A summarises event inputs and updates.

To approximate realistic deployment, we use *Qwen-VL* to predict sub-goals online and supply labels to this event interface. We also evaluate an *Oracle* protocol using simulator-annotated sub-goals as a controlled benchmark setting. Section 4 reports task outcomes under both protocols. Figure 1 illustrates how a completed subtask supplies event memory to the next subtask's action expert.

Event boundaries provide a temporal abstraction related to option termination (Sutton et al., 1999; Bacon et al., 2017). Each summary describes the most recently closed segment and remains available until the next observed transition; earlier execution remains accessible through F_t.

3.4 Query-driven memory weaver

For the preceding segment τ_{k-1}, the Weaver encodes up to $n = 130$ observations into event tokens $Z_{k-1} \in \mathbb{R}^{L \times d_h}$ using frozen visual, temporal-position, and proprioceptive features, projection, and layer normalisation (Ba et al., 2016). Learnable event latents $Q^e \in \mathbb{R}^{M \times d_h}$ provide queries; Z_{k-1} supplies keys and values. Attention pooling (Vaswani et al., 2017) normalises over segment tokens

to compress variable-length input into M fixed slots. With episode-instruction conditioning implicit in Z_{k-1} and Q^e,

$$K^e = V^e = Z_{k-1},$$
$$A^e = \text{softmax}\left(\frac{Q^e(K^e)^\top}{\sqrt{d_h}} + B^e\right), \tag{2}$$
$$m_{k-1} = P_{\text{out}}(A^e V^e + Q^e).$$

Here B^e masks segment padding, and P_{out} maps to memory width d with output normalisation. The trainable slots Q^e have an internal query residual $+Q^e$; only their content-dependent output $m_{k-1} \in \mathbb{R}^{M\times d}$ enters the action-side K/V context alongside F_t. This output describes the latest closed segment; F_t supplies earlier execution. Appendix A details inputs, conditioning, and empty-segment handling.

3.5 Action-side memory consumption

We instantiate the interface in the Gemma action expert (Gemma Team, 2024a;b) of $\pi_{0.5}$ (Physical Intelligence et al., 2025). Let X_j be the action features after block j's self-attention residual. Define $\text{Read}_j(X, S)$ as the attention readout before its output projection, with queries from X and keys and values from S. The action features read history and joint history–event context:

$$R_j^{\text{hist}} = \text{Read}_j(X_j, F_t), \tag{3a}$$
$$R_j^{\text{joint}} = \text{Read}_j(X_j, [F_t \| m_{k-1}]). \tag{3b}$$

R_j^{hist} and R_j^{joint} are action-dependent reads, each normalised within its valid context. The reads use RMSNorm (Zhang & Sennrich, 2019) and RoPE (Su et al., 2024).

Residual fusion combines the two readouts before the output map produces C_j:

$$C_j = P_j^O\left(\text{ResFuse}_j(R_j^{\text{hist}}, R_j^{\text{joint}})\right). \tag{3c}$$

MemoryRMS takes action features X_j and context C_j, which generates their scale and shift:

$$[\gamma_j, \beta_j] = C_j W_j^\Delta + b_j^\Delta, \tag{4a}$$
$$X_j' = \frac{X_j}{\sqrt{\text{mean}(X_j^2) + \varepsilon}} \odot (1 + \gamma_j) + \beta_j. \tag{4b}$$

The Weaver's query residual and the action-side residual fusion are separate operations. MemoryRMS precedes the existing timestep-conditioned pre-FFN norm and FFN, preserving the original time-conditioning pathway.

3.6 Training strategy and objective

The vision encoder and non-adapter language parameters of PaliGemma (Beyer et al., 2024) remain frozen. Language LoRA adapters (Hu et al., 2021), the action expert, history projection, Weaver, memory attention/modulation, and alignment projection are trained. Training pairs each preceding segment's summary with subsequent-segment actions, restricting history to observations available at each action.

We retain the base policy's flow-matching action objective (Lipman et al., 2022), conditioned on the joint memory M_t:

$$\mathcal{L}_{\text{action}} = \mathbb{E}_{\rho,a,\epsilon}\left\|v_\theta(a^{(\rho)}, \rho \mid o_t, s_t, \ell, M_t) - (\epsilon - a)\right\|^2_{2,\text{valid}}. \tag{5}$$

The norm averages squared error only over valid action-horizon positions, excluding padding beyond the episode end.

An auxiliary cosine-alignment objective $\mathcal{L}_{\text{align}}$ aligns the event summary with the upcoming sub-goal. The total objective is

$$\mathcal{L} = \mathcal{L}_{\text{action}} + \lambda_{\text{align}}\mathcal{L}_{\text{align}}. \tag{6}$$

Subsequent-stage action supervision trains the learnable event latents by backpropagating through action reads, residual fusion, and MemoryRMS. Semantic alignment supplies a training target for the event representation; predicted sub-goals supply labels for online event triggering. Appendix A lists the key training settings.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Tasks and reporting. We evaluate all 16 RoboMME tasks (Dai et al., 2026), with 50 benchmark episodes per task and evaluation seed. We report equal-weight task means for all 16 tasks and the *repetition task group*: PICKXTIMES, SWINGXTIMES, and STOPCUBE. Requested repetition count N and Easy/Medium/Hard difficulty are separate strata. Means and sample SD summarize evaluation seeds 7, 21, and 42 of one fixed checkpoint. Each protocol contains 2,400 episodes; group SD uses seed-level group means.

Observation and event protocols. Both protocols use the same checkpoint of our method (Ours) and memory configuration, with nominal subgoal queries every 16 executed actions. For deployment-oriented evaluation, Qwen3-VL-4B-Instruct (Bai et al., 2025) supplies online subgoals using the simple-subgoal adapter; post-hold label changes trigger memory updates. Oracle supplies simulator-annotated subgoals on the same query grid as a second formal evaluation protocol. Both use seeds 7, 21, and 42. History and window configuration comparisons use Oracle and common seed 7. Timing analysis uses all 2,400 Qwen-VL trajectories, with simulator labels on visited states as temporal references. Configuration and training settings appear in Appendices B and A.

4.2 PERFORMANCE AGAINST PRIOR WORK

We compare with $\pi_{0.5}$, DP-Unet (Chi et al., 2023), SAM2Act (Fang et al., 2025), MemER (Sridhar et al., 2025), and RoboMME (FrameSamp+Modul), the strongest non-SG RoboMME variant on repetition tasks. We label this variant *RoboMME (best)* in the tables. Table 1 shows that LoopVLA with Qwen-VL reaches 80.22% on repetition tasks, compared with 73.78% for RoboMME (best). It also improves overall success and performs best on each of the three repetition tasks, supporting the use of completed-subtask memory for repeated execution. Table 5 in the appendix compares all seven methods on every task. Public baselines retain their published aggregation, so these comparisons are descriptive.

Table 1: Task success (%). The repetition task group averages PickXtimes, SwingXtimes, and StopCube. Public baselines follow RoboMME aggregation (Dai et al., 2026); Ours reports mean ± sample SD over evaluation seeds 7, 21, and 42. Column-best means, including ties, are red.

Method	All 16	Repetition	PickX	SwingX	StopCube
$\pi_{0.5}$	17.93	28.37	42.89	35.56	6.67
DP-Unet	7.02	4.00	5.33	6.67	0.00
SAM2Act	21.37	33.78	76.00	25.33	0.00
MemER	42.38	46.22	79.33	59.33	0.00
RoboMME (best)	44.51	73.78	87.33	92.00	42.00
Ours (Qwen-VL)	47.17 ± 1.42	80.22 ± 2.04	94.00 ± 2.00	97.33 ± 2.31	49.33 ± 6.43
Ours (Oracle)	47.71 ± 1.37	79.78 ± 1.39	93.33 ± 1.15	97.33 ± 1.15	48.67 ± 3.06

4.3 HISTORY COVERAGE FOR REPEATED EXECUTION

Table 2 compares $H \in \{10, 20, 32, 40\}$ with $M = 32$, Oracle, and seed 7. Success increases from 39.88% at $H = 10$ to 49.25% at $H = 32$, while success on repetition tasks increases from 60.00% to 81.33%. Increasing to $H = 40$ gives 46.75% and 74.00%, respectively. Thus $H = 32$ has the highest success among the compared configurations; additional coverage does not continue to improve execution. Among the latent counts we tried, 32 event latents yielded the best performance; additional latents did not improve execution. Appendix B retains all four points in a separate history figure and records their metadata.

Figure 3 resolves these aggregate rates by required repetition count. All four configurations complete every single-repeat episode in both tasks. They separate as the demand grows: at four PickXtimes repetitions, success counts are 1/6, 3/6, 6/6, and 4/6 for increasing H. At five repetitions, $H = 40$

Table 2: History-coverage configurations under Oracle, seed 7. All rows use 32 event latents; column-best success rates are red.

History observations	Event latents	All 16 (%)	Repetition (%)
10	32	39.88	60.00
20	32	41.75	66.67
32	32	49.25	81.33
40	32	46.75	74.00

reaches 4/6 versus 3/6 at $H = 32$. The aggregate preference for $H = 32$ therefore coexists with a remaining high-count limitation; the full grid retains both patterns.

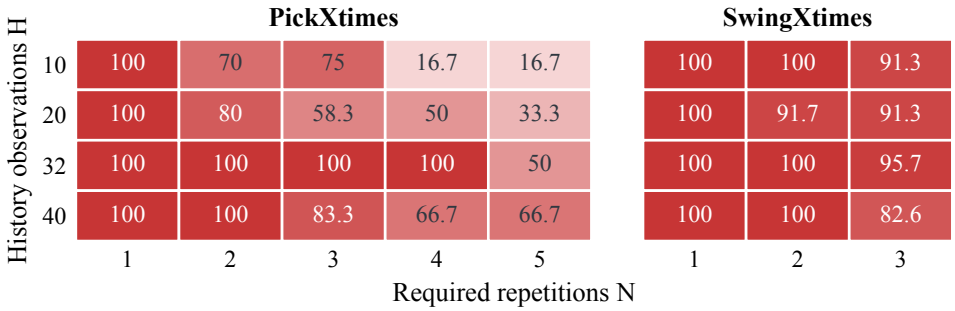

Figure 3: **History coverage across repetition demands.** Success (%) under Oracle, seed 7, $M = 32$. Cells share a white-to-red 0--100% scale and display their rates. These are configuration comparisons of history observations.

4.4 Event-aligned windows improve task completion

We compare the most recently closed event segment with a fixed recent window. Both use 32 history observations + 32 event latents and an event-input cap of 130 observations. Under Oracle and seed 7, event-aligned windows improve success from 46.38% to 49.25%, including an increase from 26.04% to 31.25% on Hard tasks. Figure 4 shows both rates and changes in episode outcomes.

Across 800 matched benchmark keys, the event configuration gains 46 successes and loses 23, with 10 net gains on Hard tasks. This supports using a complete event segment for subsequent actions, while the configuration differences prevent an isolated causal claim. Appendix B describes the pairing.

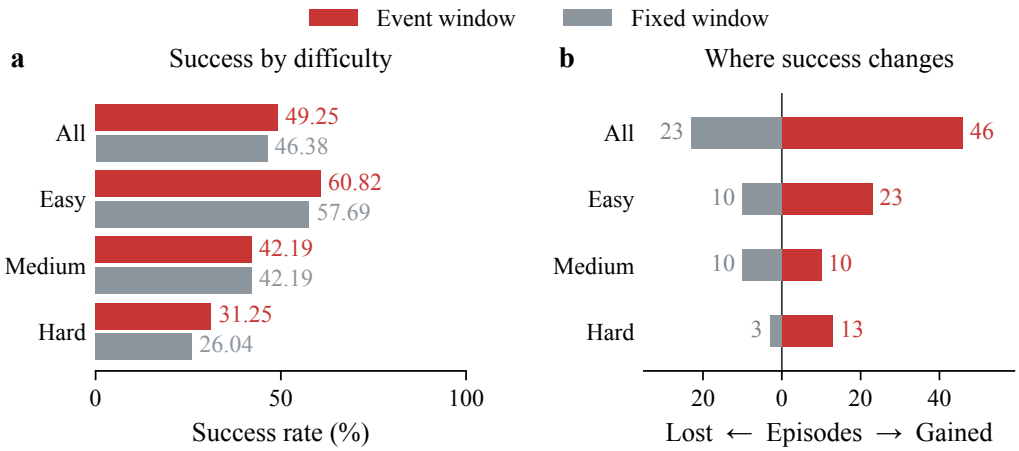

Figure 4: **Event-aligned windows expand successful execution.** (a) Absolute success rates under Oracle, seed 7; $H = M = 32$. (b) Newly successful (red) and no-longer-successful (gray) benchmark keys relative to the fixed window.

4.5 Online subgoals support repeated execution

Both protocols achieve 100% on PickXtimes through $N = 4$; at $N = 5$, Qwen-VL and Oracle reach $50.00 \pm 16.67\%$ and $44.44 \pm 9.62\%$. At $N = 3$ on SwingXtimes, they reach $97.10 \pm 2.51\%$ and $94.20 \pm 2.51\%$.

4.6 Strong task performance with online subgoals

Qwen-VL predicts subgoals online to approximate realistic deployment. Figure 5 compares its execution performance with Oracle. Across 2,400 matched rollouts, Qwen-minus-Oracle is -0.54 percentage points overall (95% CI $[-1.67, 0.50]$) and $+0.44$ on repetition tasks ($[-1.78, 2.44]$). Each of the three group tasks differs by at most 0.67 points; Appendix B reports all tasks. These intervals describe protocol differences for this checkpoint and evaluation set.

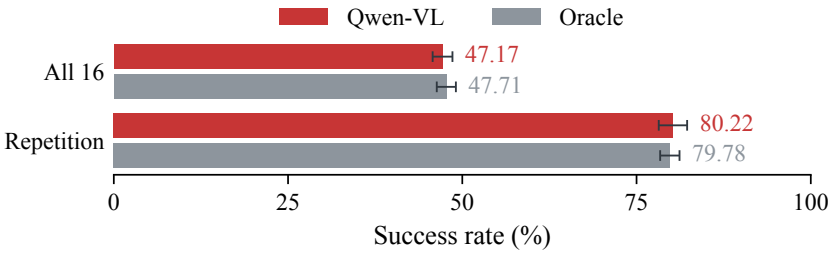

Figure 5: **Strong task performance with online subgoals.** Mean ± sample SD over evaluation seeds 7, 21, and 42; 2,400 episodes per protocol, including 450 episodes from repetition tasks.

Continuous execution between memory updates. Across all 2,400 Qwen-VL trajectories, ordered event F1 is 88.86%; matched boundaries have median offset $+10$ actions. Perceptual history continues updating between boundaries, while the last closed-segment memory remains available for action-conditioned reads until replacement. Our model is robust under these conditions and executes actions in a real-world setting. Matching has no timing cutoff; Appendix C explains event F1 and the estimation procedure.

4.7 Real-world repeated execution

We train LoopVLA with 200 real-robot demonstrations per task and deploy it on two NVIDIA RTX 5090 GPUs, using Qwen-VL subgoals for online memory updates. We evaluate LoopVLA and $\pi_{0.5}$ on 20 trials per task: *Drawer Push–Pull* (three pull–push cycles), *Bowl Transfer* (four round trips), and *Battery Transfer* (three round trips followed by placement in a cup). Each task requires tracking completed cycles to stop or switch actions at the requested count.

LoopVLA succeeds in 39 of 60 real-robot trials, compared with 21 for $\pi_{0.5}$ (Table 3). These results show that cross-subtask event memory also supports repeated execution in real-world settings.

Table 3: Real-robot task success (successes/trials), with 20 trials per task and method.

Method	Drawer Push–Pull	Bowl Transfer	Battery Transfer	Total
$\pi_{0.5}$	5/20	7/20	9/20	21/60
Ours	**11/20**	**13/20**	**15/20**	**39/60**

5 Conclusion

We presented **LoopVLA**, a cross-subtask event-memory interface that organises completed execution at subgoal transitions for the next subtask's action expert. Learnable event latents summarise the most recently closed segment; action-dependent reads of perceptual history and joint history–event context are fused and injected through MemoryRMS to inform when to continue, repeat, or stop.

With online Qwen-VL subgoals, LoopVLA achieves 47.17% overall success across 16 simulation tasks and 80.22% on repetition tasks, versus 22.67% with memory off. Real-robot results show that this memory mechanism also supports repeated execution in real-world settings, with 39 successes in 60 trials for LoopVLA and 21 for $\pi_{0.5}$.

AI USE STATEMENT

We used generative AI tools to revise and polish manuscript text and to search for and identify relevant literature.

REPRODUCIBILITY STATEMENT

The appendix documents memory and training configurations, evaluation aggregation, and deployment analyses. We provide demonstration videos in the supplementary material via an anonymous public cloud link.

REFERENCES

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *AAAI*, 2017.

S. Bai, Y. Cai, R. Chen, K. Chen, X. Chen, et al. Qwen3-VL Technical Report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, et al. PaliGemma: A versatile 3B VLM for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, et al. π_0: A vision-language-action flow model for general robot control, 2024.

A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control, 2023.

A. Bulatov, Y. Kuratov, and M. S. Burtsev. Recurrent memory transformer, 2022.

T. Chen, H. Wu, J. Wang, X. Li, and L. Fang. StreamVLA: Breaking the reason-act cycle via completion-state gating. *arXiv preprint arXiv:2602.01100v2*, 2026a.

Y. Chen, W. Tan, L. Zhu, F. Li, J. Li, G. Yang, and H. T. Shen. Non-Markovian long-horizon robot manipulation via keyframe chaining. *arXiv preprint arXiv:2603.01465*, 2026b.

C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

Y. Dai, H. Fu, J. Lee, Y. Liu, H. Zhang, J. Yang, C. Finn, N. Fazeli, and J. Chai. RoboMME: Benchmarking and understanding memory for robotic generalist policies. *arXiv preprint arXiv:2603.04639*, 2026.

D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman. Counting out time: Class-agnostic video repetition counting in the wild, 2020.

Y. Fan, P. Ding, S. Bai, X. Tong, Y. Zhu, et al. Long-VLA: Unleashing long-horizon capability of vision-language-action model for robot manipulation. In *CoRL*, 2025.

H. Fang, M. Grotz, W. Pumacay, Y. R. Wang, D. Fox, R. Krishna, and J. Duan. SAM2Act: Integrating visual foundation model with a memory architecture for robotic manipulation, 2025.

Gemma Team. Gemma: Open models based on Gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024a.

Gemma Team. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024b.

X. Guo, Z. Huang, Z. Shi, Z. Song, and J. Zhang. Your vision-language model can't even count to 20: Exposing the failures of VLMs in compositional counting. *arXiv preprint arXiv:2510.04401*, 2025.

S. Han, B. Qiu, Y. Liao, S. Huang, C. Gao, S. Yan, and S. Liu. RoboCerebra: A large-scale benchmark for long-horizon robotic manipulation evaluation, 2025.

E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, et al. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

H. Hu, S. Dong, Y. Zhao, D. Lian, Z. Li, and S. Gao. TransRAC: Encoding multi-scale temporal correlation with transformers for repetitive action counting, 2022.

H. Jang, S. Yu, H. Kwon, H. Jeon, Y. Seo, and J. Shin. ContextVLA: Vision-language-action model with amortized multi-frame context, 2025.

M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, et al. OpenVLA: An open-source vision-language-action model, 2024.

T. Kipf, Y. Li, H. Dai, V. Zambaldi, A. Sanchez-Gonzalez, E. Grefenstette, P. Kohli, and P. Battaglia. CompILE: Compositional imitation learning and execution. In *ICML*, 2019.

M. Koo, D. Choi, T. Kim, K. Lee, C. Kim, Y. Seo, and J. Shin. HAMLET: Switch your vision-language-action model into a history-aware policy. *arXiv preprint arXiv:2510.00695*, 2025.

H. Li, F. Shen, D. Chen, L. Yang, X. Wang, J. Shi, Z. Bing, Z. Liu, and A. Knoll. ReMem-VLA: Empowering vision-language-action model with memory via dual-level recurrent queries. *arXiv preprint arXiv:2603.12942*, 2026.

J. Li, D. Li, S. Savarese, and S. Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023.

R. Li, W. Guo, Z. Wu, C. Wang, H. Deng, Z. Weng, Y.-P. Tan, and Z. Wang. MAP-VLA: Memory-augmented prompting for vision-language-action model in robotic manipulation. *arXiv preprint arXiv:2511.09516*, 2025.

Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling, 2022.

Z. Liu, Y. Gu, S. Zheng, Y. Fu, X. Xue, and Y.-G. Jiang. TriVLA: A triple-system-based unified vision-language-action model with episodic world modeling for general robot control. *arXiv preprint arXiv:2507.01424v3*, 2025a.

Z. Liu, Y. Wang, K. Wang, L. Liang, X. Xue, and Y. Fu. Spatial-temporal aware visuomotor diffusion policy learning. In *ICCV*, 2025b.

Z. Liu, Y. Wang, S. Zheng, T. Pan, L. Liang, Y. Fu, and X. Xue. ReasonGrounder: LVLM-guided hierarchical feature splatting for open-vocabulary 3D visual grounding and reasoning. In *CVPR*, 2025c.

Z. Liu, S. Zheng, S. Chen, C. Zhao, L. Liang, X. Xue, and Y. Fu. A neural representation framework with LLM-driven spatial reasoning for open-vocabulary 3D visual grounding. In *ACM MM*, 2025d.

Z. Liu, Y. Gu, Y. Wang, X. Xue, and Y. Fu. ActiveVLA: Injecting active perception into vision-language-action models for precise 3D robotic manipulation. *arXiv preprint arXiv:2601.08325*, 2026.

M. Sobol Mark, J. Liang, M. Attarian, C. Fu, D. Dwibedi, D. Shah, and A. Kumar. BPP: Long-context robot imitation learning by focusing on key history frames. *arXiv preprint arXiv:2602.15010*, 2026.

NVIDIA, J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, et al. GR00T N1: An open foundation model for generalist humanoid robots, 2025.

NVIDIA GEAR Team. GR00T N1.6: An improved open foundation model for generalist humanoid robots, 2025. URL https://research.nvidia.com/labs/gear/gr00t-n1_6.

R. Paiss, A. Ephrat, O. Tov, S. Zada, I. Mosseri, M. Irani, and T. Dekel. Teaching CLIP to count to ten, 2023.

Physical Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization, 2025.

H. Shi, B. Xie, Y. Liu, L. Sun, F. Liu, et al. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation, 2025.

S. Sinha, A. Stergiou, and D. Damen. Every shot counts: Using exemplars for repetition counting in videos. In *ACCV*, 2024.

A. Sridhar, J. Pan, S. Sharma, and C. Finn. MemER: Scaling up memory for robot control via experience retrieval. *arXiv preprint arXiv:2510.20328*, 2025.

J. Su, A. Murtadha, Y. Lu, S. Pan, Wen. Bo, and Y. Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999.

M. Torne, K. Pertsch, H. Walke, K. Vedder, S. Nair, B. Ichter, A. Z. Ren, et al. MEM: Multi-scale embodied memory for vision language action models, 2026.

R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *ICML*, 2018.

P. Vaezipoor, A. C. Li, R. Toro Icarte, and S. A. McIlraith. LTL2Action: Generalizing LTL instructions for multi-task RL. In *ICML*, 2021.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

W. Wan, Y. Zhu, R. Shah, and Y. Zhu. LOTUS: Continual imitation learning for robot manipulation through unsupervised skill discovery. In *ICRA*, 2024.

R. Yang, Z. An, L. Zhou, and Y. Feng. SeqVLA: Sequential task execution for long-horizon manipulation with completion-aware vision-language-action model. *arXiv preprint arXiv:2509.14138*, 2025.

X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid Loss for Language Image Pre-Training. In *ICCV*, 2023. URL https://arxiv.org/abs/2303.15343.

B. Zhang and R. Sennrich. Root mean square layer normalization. In *NeurIPS*, 2019.

G. Zhang, M. Fu, and S. Yan. MemGen: Weaving generative latent memory for self-evolving agents. *arXiv preprint arXiv:2509.24704*, 2025a.

J. Zhang, Y. Guo, X. Chen, Y.-J. Wang, Y. Hu, C. Shi, and J. Chen. HiRT: Enhancing robotic control with hierarchical robot transformers. In *CoRL*, volume 270 of *PMLR*, pp. 933–946, 2025b.

T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.

R. Zheng, Y. Liang, S. Huang, J. Gao, H. Daumé, III, A. Kolobov, F. Huang, and J. Yang. TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies, 2024.

Y. Zhu, P. Stone, and Y. Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters*, 2022.

A Implementation Details

Table 4 lists the settings needed to interpret the experiments.

Table 4: Key settings for LoopVLA.

Component	Setting
Policy backbone	$\pi_{0.5}$ with a trainable memory interface
Memory setting	32 history observations + 32 event latents
Event input	Up to 130 observations from the latest closed segment
Event query	Every 16 executed actions
Action prediction	Chunks of 20 actions; episode-end padding is masked
Training	3 epochs; AdamW with warm-up and cosine decay
Training objectives	Flow matching and next-subgoal cosine alignment

Event inputs and updates. Visual, temporal-position, and proprioceptive features are projected and normalised. The instruction conditions the event features and learnable queries. Padding is masked; an empty segment produces invalid, zero-valued memory. Offline subgoal labels are carried between query points, and each transition pairs the closed segment with subsequent actions. History uses only observations available at each action. Online, Qwen-VL predicts the subgoal labels used to trigger event-memory updates.

B Complete Numerical Results

Aggregation and coverage. Each task has 50 benchmark episodes per policy seed. All-task success is the equal-weight mean over 16 tasks; the repetition task group averages PickXtimes, SwingXtimes, and StopCube. Both protocols use seeds 7, 21, and 42 of the same fixed checkpoint and report their mean and sample SD (ddof $= 1$). Group SD is calculated from each seed's group mean.

Table 5 compares the seven main-table methods on every task. All LoopVLA results use the same fixed best-performance checkpoint.

Table 5: Task success (%). Ours reports means over evaluation seeds 7, 21, and 42; public baselines follow RoboMME aggregation (Dai et al., 2026). The three repetition tasks are listed first. Row-best means, including ties, are red.

Task	$\pi_{0.5}$	DP-Unet	SAM2Act	MemER	RoboMME (best)	**Ours (Qwen-VL)**	**Ours (Oracle)**
PickXtimes	42.89	5.33	76.00	79.33	87.33	94.00	93.33
SwingXtimes	35.56	6.67	25.33	59.33	92.00	97.33	97.33
StopCube	6.67	0.00	0.00	0.00	42.00	49.33	48.67
BinFill	30.00	30.67	40.00	56.67	39.56	40.67	44.67
VideoUnmask	20.44	14.00	27.33	81.33	32.67	43.33	42.67
ButtonUnmask	22.22	4.00	32.00	72.00	25.11	26.67	22.00
VideoUnmaskSwap	18.67	4.00	18.00	38.00	24.44	26.67	29.33
ButtonUnmaskSwap	6.67	0.00	26.67	21.33	18.22	11.33	19.33
PickHighlight	11.33	0.00	17.33	70.67	22.89	24.00	23.33
VideoRepick	0.44	0.00	5.33	25.33	30.44	30.00	30.67
VideoPlaceButton	31.11	22.00	24.67	30.00	60.00	55.33	55.33
VideoPlaceOrder	25.78	4.33	20.00	26.00	32.00	39.33	38.67
MoveCube	26.00	8.00	29.33	82.67	77.78	82.67	84.00
InsertPeg	1.56	0.00	0.00	6.67	7.56	6.67	6.67
PatternLock	2.89	3.33	0.00	16.67	53.56	61.33	62.67
RouteStick	4.67	10.00	0.00	12.00	66.67	66.00	64.67
All 16	17.93	7.02	21.37	42.38	44.51	47.17	47.71
Repetition	28.37	4.00	33.78	46.22	73.78	80.22	79.78

Memory-off results. The memory-off configuration succeeds in 19/50 PickXtimes, 14/50 SwingXtimes, and 1/50 StopCube episodes, giving 22.67% across this group. The main Qwen-VL protocol succeeds in 361/450 episodes (80.22%).

Configuration metadata and comparability. Table 6 records the memory settings used in the history and window comparisons. All use the same event-input cap, Oracle subgoals, and evaluation seed 7.

Table 6: Memory settings for the history and window comparisons.

Configuration	History observations	Event latents
Event history	10	32
Event history	20	32
Main event history	32	32
Event history	40	32
Fixed recent window	32	32

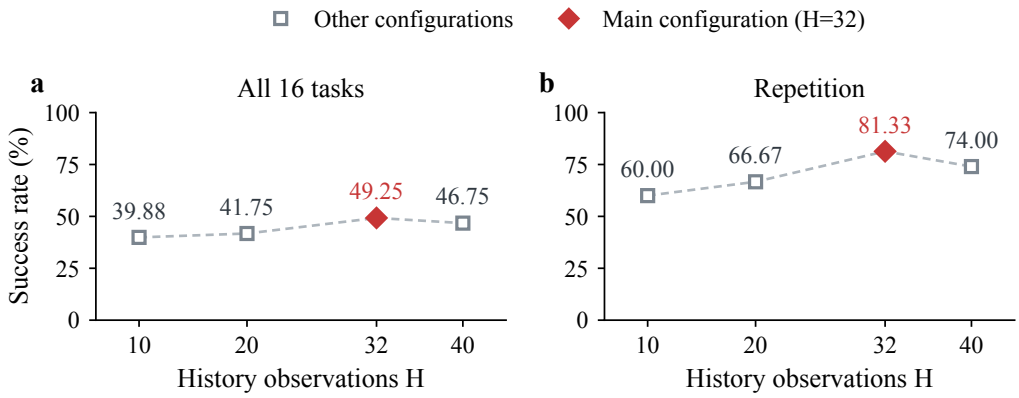

Figure 6: **History coverage across all compared configurations.** Oracle, seed 7, $M = 32$: 800 episodes per point, including 150 episodes from repetition tasks. The $H = 32$ configuration achieves the highest measured success. All four budgets are shown on 0--100% axes; points are single-seed rates. Table 6 documents configuration differences.

Window pairing. The window comparison pairs 800 task/episode keys at seed 7. This assumes the same initial states.

C Execution with Online Subgoals

Execution with predicted subgoals. Table 7 shows that Qwen-VL achieves repetition-task success close to Oracle despite imperfect boundary timing. Perceptual history continues to update, and the previous event memory remains available until replacement. These results support robust execution with online subgoal predictions.

What event F1 measures. Ordered event F1 is the harmonic mean of precision and recall for subgoal transitions matched by stage label and temporal order. Extra predictions lower precision; missed reference transitions lower recall. The score of 88.86% indicates good recovery of the subgoal sequence. Matching has no timing cutoff, so F1 measures transition recognition; boundary offsets separately measure timing accuracy.

Table 7: Execution and event timing with online Qwen-VL subgoals in simulation. Offsets use matched boundaries; positive values indicate later updates.

Scope	Metric	Result
All 16 tasks	Ordered event F1	88.86%
	Median boundary offset	+10 actions
Repetition tasks	Success (Qwen-VL / Oracle)	80.22% / 79.78%

Measurement. Timing analysis uses all 2,400 Qwen-VL trajectories. The protocol intervals in Section 4.6 use 5,000 task-stratified bootstrap resamples of benchmark episode keys, retaining both protocols and all three seeds together. They describe this checkpoint and evaluation set, not variation across training runs.

D Execution Examples and Real-world Setup

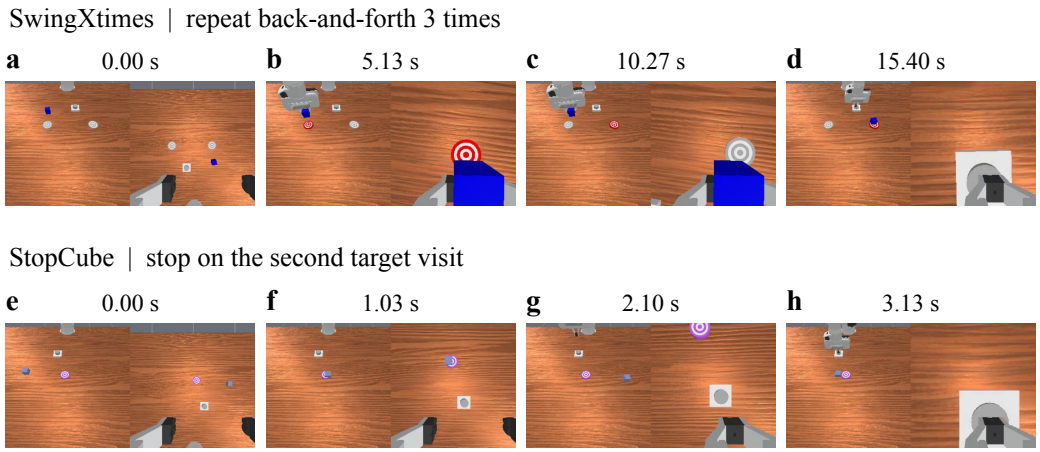

Figure 7: **Archived successful rollouts with predicted sub-goals.** Earlier reference configuration, checkpoint 21284, Qwen-VL, seed 7. Top: SwingXtimes episode 17, three back-and-forth repetitions. Bottom: StopCube episode 3, stopping on the second target visit. Each image retains both camera views. Timestamps are rendered-video seconds; images are uniformly sampled.

Real-world setup. Figure 8 illustrates the dual-arm ALOHA setup. Using online Qwen-VL sub-goals, LoopVLA succeeds in 39 of 60 real-robot trials, compared with 21 for $\pi_{0.5}$ (Section 4.7). These results show that the memory mechanism also works for repeated execution in a physical setting.

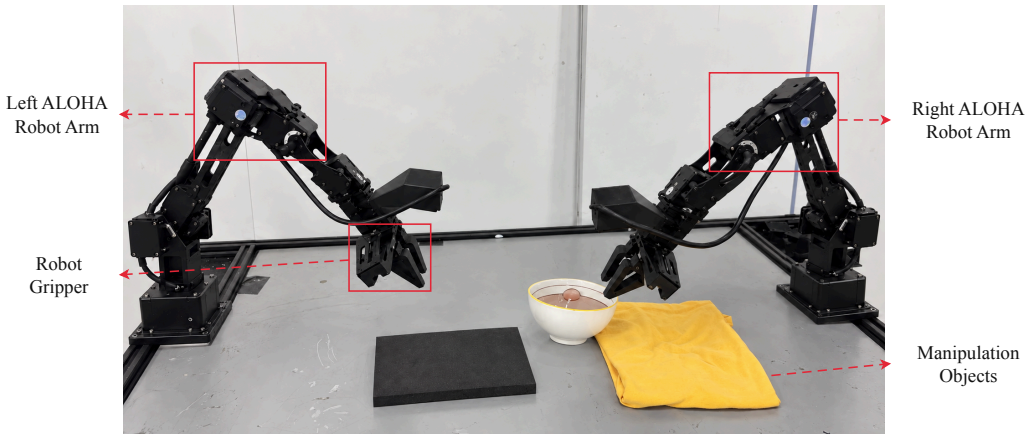

Figure 8: **Real-world experimental setup.** Illustration of the dual-arm ALOHA platform and table-top workspace used for the repeated manipulation tasks.