Retrieve-then-Steer: Online Success Memory for Test-Time Adaptation of Generative VLAs

Jianchao Zhao[1,2] **Huoren Yang**[1,2] **Yusong Hu**[2,†] **Yuyang Gao**[2] **Qiguan Ou**[2]
Cong Wan[4] **SongLin Dong**[3] **Zhiheng Ma**[3,*] **Yihong Gong**[1,3]

[1]National Key Laboratory of Human-Machine Hybrid Augmented Intelligence,
National Engineering Research Center for Visual Information and Applications,
Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University
[2]One Robotics
[3]Faculty of Computility Microelectronics, Shenzhen University of Advanced Technology
[4]School of Software Engineering, Xi'an Jiaotong University

{zhaojianchao,yanghuoren,wancong}@stu.xjtu.edu.cn,
{dongsl,mazhiheng}@suat-sz.edu.cn, ygong@mail.xjtu.edu.cn,
ys@onerobot.com, hilbertgao@gmail.com, ou1531037815@gmail.com

Abstract

Vision-Language-Action (VLA) models show strong potential for general-purpose robotic manipulation, yet their closed-loop reliability often degrades under local deployment conditions. Existing evaluations typically treat test episodes as independent zero-shot trials. However, real robots often operate repeatedly in the same or slowly changing environments, where successful executions provide environment-verified evidence of reliable behavior patterns. We study this persistent-deployment setting, asking whether a partially competent frozen VLA can improve its reliability by reusing its successful test-time experience. We propose an online success-memory guided test-time adaptation framework for generative VLAs. During deployment, the robot stores progress-calibrated successful observation-action segments in a long-term memory. At inference, it retrieves state-relevant action chunks, filters inconsistent candidates via trajectory-level consistency, and aggregates them into an elite action prior. To incorporate this prior into action generation, we introduce confidence-adaptive prior guidance, which injects the elite prior into an intermediate state of the flow-matching action sampler and adjusts the guidance strength based on retrieval confidence. This design allows the frozen VLA to exploit environment-specific successful experience while preserving observation-conditioned generative refinement. This retrieve-then-steer mechanism enables lightweight, non-parametric test-time adaptation without requiring parameter updates. Simulation and real-world experiments show improved task success and closed-loop stability, especially in long-horizon and multi-stage tasks. Code is available at https://github.com/ZJC25127/Retrieve-then-Steer.

1 Introduction

Vision-Language-Action (VLA) models [3, 36, 13, 2], particularly generative VLAs [5, 10, 15] that incorporate diffusion or flow-matching mechanisms, show immense potential for general-purpose robotic manipulation by generating expressive and temporally coherent action chunks. However, a significant disconnect exists between current evaluation paradigms and real-world deployment.

*Corresponding author. †Project leader.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Most benchmarks treat testing as independent zero-shot trials, overlooking that real robots typically perform repetitive tasks in static or slowly changing environments. In such settings, physical layouts, camera viewpoints, calibration errors, and task patterns exhibit strong inter-episode correlation. Consequently, deployment should not be viewed as a series of isolated test episodes, but rather as a persistent online process operating under correlated local conditions.

Embracing this "persistent online" perspective is vital. While many existing VLAs possess strong foundational capabilities, their closed-loop execution often remains unstable during real-world deployment. Although a robot might occasionally complete a task, it is highly prone to failure in nearly identical states due to perception noise, viewpoint shifts, or accumulated errors [8, 22]. This fragility underscores the value of successful experiences. A successful grasp or placement, for instance, implicitly captures the visual geometry, actuation biases, and execution timing specific to that environment. Consequently, these trajectories should not be treated as isolated samples discarded after evaluation, but rather as environment-verified evidence that dictates reliable behavior patterns under the current physical and visual settings. This motivates our central question: *can a frozen base VLA improve its reliability by reusing its own successful test-time interactions?*

A review of existing research, however, reveals that current paradigms have yet to provide a satisfactory answer to this question. First, pre-training and downstream fine-tuning [20, 13, 10] enhance policies before deployment but fail to enable continuous learning from test-time successes. To this end, reinforcement learning and human-in-the-loop methods [12, 23] utilize deployment experience but typically require extra feedback, safe exploration, and heavy parameter updates. In contrast, recent test-time steering [31, 14, 11, 6] focuses on inference by sampling and filtering action candidates. However, these training-free approaches generally follow a myopic "generate-then-select" paradigm, discarding candidates after local evaluation. Consequently, they struggle to exploit successful experiences accumulated across repeated episodes.

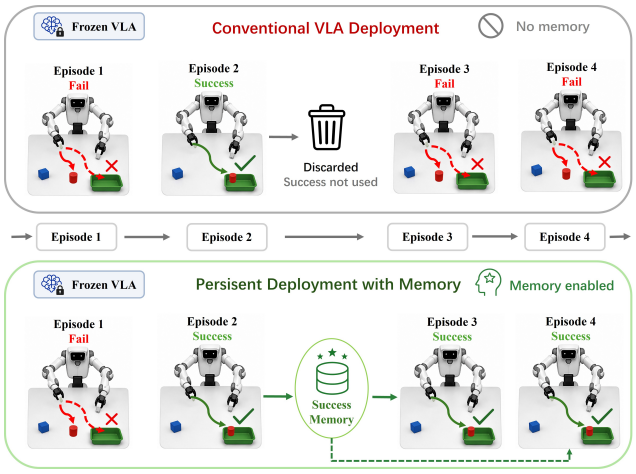


Figure 1: Motivation for persistent VLA deployment. Our method stores successful trials in online memory and reuses them to stabilize later executions without updating the policy.

Therefore, we propose directly formulating successful test-time experience as a soft behavioral prior for generative VLAs. Rather than assuming independent test episodes, we organize successful executions as reusable local evidence for future action generation. This yields a novel "retrieve-then-steer" paradigm: an online success memory compels the frozen VLA toward behavior patterns proven effective in the target environment, while preserving its ability to condition on real-time observations.

To operationalize this paradigm, we introduce an online success-memory guided test-time adaptation framework for generative VLAs. Specifically, during continuous deployment, the robot stores progress-calibrated successful observation-action prefixes while excluding failed or redundant motions. At inference, it retrieves observation-relevant action chunks, eliminates conflicting trajectories via consistency filtering, and aggregates high-quality candidates into an elite action prior. To integrate this prior into action generation, we propose confidence-adaptive prior guidance. By injecting the elite prior directly into the intermediate state of the generative sampler, the system dynamically adjusts guidance strength based on retrieval confidence. This ensures high-confidence retrievals dictate successful behavior patterns, while uncertain retrievals safely revert to the original VLA sampler.

We evaluate the proposed framework in both **simulated and real-world** robotic manipulation. Our method improves generative VLA policies on long-horizon language-conditioned manipulation benchmarks, including LIBERO-10 and SimplerEnv, and further shows consistent gains on real-world bimanual manipulation tasks. Across these settings, online success-memory guidance improves task

success and closed-loop stability, especially on long-horizon and multi-stage tasks. Our contributions are threefold:

- We redefine VLA deployment as a persistent online adaptation process rather than isolated trials, highlighting successful test-time interactions as a critical source of environment-specific evidence for enhancing policy reliability.
- We propose a non-parametric "retrieve-then-steer" mechanism that enables lightweight test-time adaptation (TTA) for frozen VLAs. This mechanism utilizes a progress-calibrated success memory to extract reusable segments and injects consistency-filtered elite priors into the generative sampling process, guiding the model without requiring parameter updates.
- We systematically validate the framework across long-horizon benchmarks and real-world bimanual manipulation. Results demonstrate our method significantly improves success rates and strengthens closed-loop stability in complex, multi-stage tasks.

2 Related Work

Vision-Language-Action Models. Vision-Language-Action models (VLAs) [3, 36, 13, 20, 9] have become a promising paradigm for general-purpose robotic policies by unifying visual perception, language understanding, and action generation. Early systems such as RT-1 [3] and OpenVLA [13] learn end-to-end policies from large-scale robotic data, while recent generative policies, including Diffusion Policy [5] and $\pi_0/\pi_{0.5}$ [2, 10], model continuous action chunks with diffusion or flow-matching heads. Despite these advances, VLAs still suffer from sampling noise, distribution shifts, and accumulated closed-loop errors during deployment, limiting their stability and local adaptability.

Test-Time Policy Steering. Recent work has explored test-time policy steering or scaling to improve VLA deployment stability [11, 6, 14, 31]. These methods enhance current action decisions through additional sampling, external evaluators, or internal confidence signals. For example, RoboMonkey [14] selects among perturbed action candidates with a VLM-based verifier, MG-Select [11] uses condition-masking confidence for verifier-free selection, and TACO [31] constrains generation toward stable successful modes via pseudo-count estimation. While effective, these candidate-selection methods follow a generate-then-select paradigm, which incurs extra inference overhead and makes limited use of reusable cross-episode experience. DSRL [29] also adapts frozen generative policies across episodes, but learns a latent-noise policy through reward-driven reinforcement learning. Our method instead retrieves progress-verified successful action segments to guide generation, without test-time parameter updates or environmental reward feedback.

Retrieval-Augmented and Memory-Based Robot Learning. Retrieval-augmented and memory-based mechanisms [25, 18, 19] have long been used in robot learning to improve the utilization of historical experience, demonstrations, and task context. Existing methods typically retrieve relevant trajectories from offline demonstration datasets for few-shot imitation, skill retrieval, or local policy adaptation [18, 35, 27], while others treat memory as a replay buffer for continual learning or reinforcement learning [30, 4]. Although effective, these methods often rely on offline data, use retrieval mainly as context, or require policy updates, limiting their suitability for lightweight test-time adaptation of frozen VLAs. Unlike these approaches, our method constructs memory directly during deployment from verified successful executions and uses it as a lightweight non-parametric prior for frozen VLAs, without offline demonstration banks or parameter updates.

3 Preliminaries

3.1 Problem Formulation

We consider language-conditioned robotic manipulation in downstream deployment. Let π_{vla} be a generative Vision-Language-Action policy fine-tuned on downstream demonstrations. At decision step t, given observation $o_t = (I_t^{1:N_c}, q_t)$ and instruction l, the policy samples an action chunk $a_t \sim \pi_{\mathrm{vla}}(\cdot \mid o_t, l)$ with horizon H, where $I_t^{1:N_c}$ denotes multi-view RGB images and q_t denotes the proprioceptive state. During each test episode, the robot executes action chunks in closed loop, producing a trajectory $\tau^{(i)} = \{(o_t^{(i)}, a_t^{(i)})\}_{t=0}^{T_i-1}$. Unlike standard zero-shot evaluation that treats

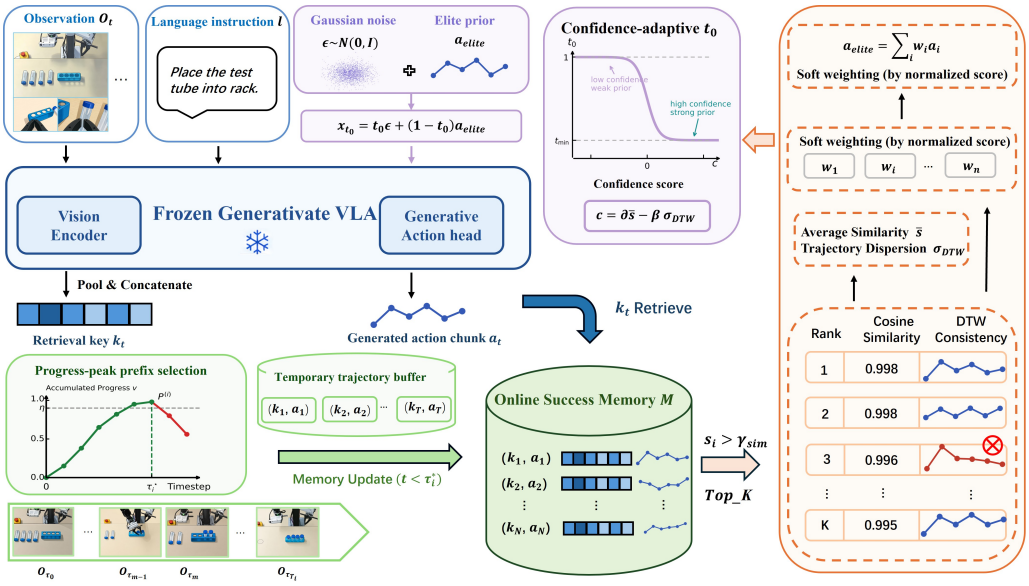


Figure 2: Overview of our retrieve-then-steer test-time adaptation framework. The frozen VLA accumulates progress-verified successful observation–action segments in an online success memory. For each new observation, relevant action chunks are retrieved, filtered, and aggregated into an elite prior, which initializes the flow-matching sampler with confidence-adaptive guidance.

episodes independently, we study continuous deployment, where successful cross-episode experience can be accumulated and reused for test-time adaptation.

4 Methodology

4.1 Online Progress-Calibrated Memory

Since the frozen VLA cannot update its parameters during deployment, we construct an Online Progress-Calibrated Memory $\mathcal{M}$ to store successful observation–action segments.

Trajectory buffering and memory representation. For the i-th test episode, we maintain a temporary buffer $\mathcal{B}^{(i)} = \{(k_t^{(i)}, a_t^{(i)})\}_{t=0}^{T_i-1}$ to record candidate memory entries generated during execution. Here, $k_t^{(i)}$ denotes the retrieval key at time step t. Instead of introducing a separate retrieval encoder, we reuse the VLA visual encoder to extract image features. For each view, spatial patch tokens are reshaped into a feature grid, downsampled by 2×2 average pooling, and flattened. For multi-view observations, features from all views are concatenated and normalized to form $k_t^{(i)}$.

Interval-based progress calibration. To identify reusable successful experience, we instantiate the progress estimator Φ_ψ with a pretrained VLAC critic [32], which automatically evaluates trajectories for memory construction without human success labels. For each task, one successful demonstration video from the training set is used as the reference process R, serving as an in-context example of the task execution procedure. Details are provided in Appendix A.

Let Δ denote the evaluation interval. For trajectory i, we evaluate progress at timesteps $\mathcal{T}_i^\Delta = \{0, \Delta, 2\Delta, \ldots, \lfloor T_i/\Delta \rfloor \Delta\} \cup \{T_i\}$. For adjacent timesteps $\tau_{m-1}, \tau_m \in \mathcal{T}_i^\Delta$, the pretrained critic predicts the signed progress change conditioned on the instruction and reference process:

$$c_{\tau_m}^{(i)} = \Phi_\psi \left(o_{\tau_{m-1}}^{(i)}, o_{\tau_m}^{(i)}, l^{(i)}; R \right), \tag{1}$$

the value of $c_{\tau_m}^{(i)}$ indicates whether the task progresses or regresses within the interval. We then accumulate interval-level progress into a trajectory-level progress score $v_{\tau_m}^{(i)}$, initialized as $v_{\tau_0}^{(i)} = 0$:

$$v_{\tau_m}^{(i)} = v_{\tau_{m-1}}^{(i)} + \left(100 - v_{\tau_{m-1}}^{(i)}\right) \frac{c_{\tau_m}^{(i)}}{100}. \tag{2}$$

Progress-peak prefix selection and memory update. After the episode finishes, we define the completion score as the maximum accumulated progress, $P^{(i)} = \max_{\tau_m \in \mathcal{T}_i^{\Delta}} v_{\tau_m}^{(i)}$, and denote the progress-peak timestep as $\tau_i^{\star} = \arg\max_{\tau_m \in \mathcal{T}_i^{\Delta}} v_{\tau_m}^{(i)}$. We use the maximum accumulated progress instead of terminal progress to handle possible regressions after near-success, such as overshooting, collisions, or unnecessary motions. This allows the progress peak to preserve the best achieved task state and retain reusable successful experience. Given a success threshold η, the episode-level success indicator is defined as $y^{(i)} = \mathbb{I}[P^{(i)} \geq \eta]$. If $y^{(i)} = 0$, the temporary buffer is discarded; otherwise, we retain only candidate entries before the progress peak: $\mathcal{B}_+^{(i)} = \{(k_t^{(i)}, a_t^{(i)}) \in \mathcal{B}^{(i)} \mid t \leq \tau_i^{\star}\}$. Then the online success memory is updated as $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{B}_+^{(i)}$.

4.2 Retrieval-based Action Prior

When the online success memory $\mathcal{M}$ is non-empty, we retrieve historical successful actions that are similar to the current state during test time, which are used to assist subsequent action generation.

Successful action retrieval with similarity gating. Given the current retrieval key k_t, for each memory entry $(k_i, a_i) \in \mathcal{M}$, we compute its relevance to the current state using cosine similarity, $s_i = \frac{\langle k_t, k_i \rangle}{\|k_t\|_2 \|k_i\|_2}$. We then select the top-K candidates with the highest similarity scores and remove weakly related results using a threshold γ_{sim}, yielding the initial candidate set $\mathcal{I}_{\text{sim}}$.

DTW-based trajectory consistency filtering. State-level similarity alone may still introduce action-level mismatches, where retrieved states are close to the current state but their action chunks follow inconsistent trajectory patterns. To remove such outliers, we compute pairwise multivariate Dynamic Time Warping (DTW) distances [24, 26] among the candidates in $\mathcal{I}_{\text{sim}}$:

$$d_{ij} = \text{DTW}(a_i, a_j), \quad i, j \in \mathcal{I}_{\text{sim}}. \tag{3}$$

For each candidate action chunk a_i, we define its trajectory inconsistency score as the median distance to the remaining candidates, $r_i = \text{median}_{j \in \mathcal{I}_{\text{sim}}, j \neq i} d_{ij}$. A larger r_i indicates that the candidate deviates from the dominant successful trajectory pattern. We therefore remove candidates with excessively large inconsistency scores and obtain the final candidate set $\mathcal{I}$.

Elite action prior aggregation. Given the filtered candidate set $\mathcal{I}$, we aggregate multiple successful action chunks with similarity-based soft weights, rather than directly selecting a single nearest-neighbor action. The weight of each candidate is defined as

$$w_i = \frac{\exp\left((s_i - \max_{j \in \mathcal{I}} s_j)/\tau\right)}{\sum_{j \in \mathcal{I}} \exp\left((s_j - \max_{m \in \mathcal{I}} s_m)/\tau\right)}, \quad i \in \mathcal{I}, \tag{4}$$

where $\tau > 0$ is a temperature parameter that controls the sharpness of the weight distribution. The resulting elite action prior is given by

$$a_{\text{elite}} = \sum_{i \in \mathcal{I}} w_i a_i. \tag{5}$$

This formulation provides a unified representation for action-prior aggregation. For action components in Euclidean spaces, such as positions, joint angles, we adopt linear weighted aggregation; for orientations, we compute the geodesic mean on $SO(3)$. Further details are provided in Appendix C.

4.3 Confidence-Adaptive Prior Guidance

The retrieved elite action prior a_{elite} provides a local successful behavior reference from the online success memory. However, unreliable retrievals caused by representation bias, nearest-neighbor

mismatch, or trajectory inconsistency may introduce incorrect constraints. We therefore propose *confidence-adaptive prior guidance*, which injects the retrieved prior into the flow-matching sampler as a soft generative constraint and adapts its strength according to retrieval confidence.

For a VLA with a flow-matching action head, the original sampler starts from Gaussian noise $x_1 = \epsilon, \epsilon \sim \mathcal{N}(0, I)$ and integrates the conditional velocity field $v_\theta(x_t, t, z_t)$ from $t = 1$ to $t = 0$, where z_t is the conditioning feature from the current observation and instruction. Instead of modifying the model or velocity field, we initialize the sampling process from an intermediate state:

$$x_{t_0} = (1 - t_0)a_{\text{elite}} + t_0\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{6}$$

Here, $t_0 \in [0, 1]$ controls the guidance strength. A smaller t_0 places the initial state closer to a_{elite}, yielding stronger prior guidance, while a larger t_0 preserves more randomness and recovers the original sampler.

To adapt the guidance strength to retrieval reliability, we estimate a confidence score from both state-level similarity and action-level consistency. Given the filtered candidate set $\mathcal{I}$, we compute the average retrieval similarity $\bar{s}_{\text{top-}K} = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} s_i$. Since deployment similarities often lie in a narrow high-score range, we normalize it as

$$\tilde{s} = \text{clip}\left(\frac{\bar{s}_{\text{top-}K} - s_{\text{ref}}}{s_{\text{scale}}}, -c_{\max}, c_{\max}\right). \tag{7}$$

We further measure action-level dispersion using the DTW inconsistency scores, $\sigma_{\text{DTW}} = \text{Std}(\{r_i\}_{i\in\mathcal{I}})$, and define the retrieval confidence as

$$c = \alpha\tilde{s} - \beta\sigma_{\text{DTW}}, \tag{8}$$

where larger c indicates a more reliable prior. Finally, we map the retrieval confidence to the sampling starting time:

$$t_0 = t_{\min} + (1 - t_{\min})\sigma(-\gamma c), \tag{9}$$

where $t_{\min}$ is the strongest-guidance starting time and γ controls the mapping sharpness. Higher confidence moves t_0 toward $t_{\min}$, while lower confidence moves it toward 1, recovering the original sampler.

After determining t_0, the sampler starts from x_{t_0} and integrates the same conditional velocity field to $t = 0$. With Euler discretization, the update is

$$x_{t-\Delta t} = x_t - \Delta t \cdot v_\theta(x_t, t, z_t), \qquad t : t_0 \to 0. \tag{10}$$

The final state x_0 serves as the generated action chunk $\hat{a}_{t:t+H-1}$. If the memory is empty or retrieval fails the similarity and trajectory-consistency filters, no prior is injected and the method falls back to the original sampling process. We also provide the diffusion counterpart in Appendix B.

5 Experiments

5.1 Simulation Experiments

5.1.1 Setup and Baselines

Benchmarks. We evaluate our method on two simulation benchmarks, LIBERO [17] and SimplerEnv [16]. LIBERO is a benchmark for lifelong learning in decision making, consisting of multiple task suites. As easier suites are near-saturated, we focus on the more challenging LIBERO-10 suite to examine whether our method can mitigate state drift, accumulated action errors, and unstable closed-loop execution in long-horizon, multi-stage manipulation tasks. SimplerEnv is a real-to-sim manipulation benchmark built upon the SAPIEN simulator and the ManiSkill2 benchmark, providing simulated task environments for both the WidowX and Google Robot platforms. In this work, we primarily use the tasks designed for the Google Robot platform to evaluate robustness under realistic deployment conditions, including object layout variations, visual perturbations, and fine-grained manipulation.

Table 1: Success rates (%) on LIBERO-10. Each task is evaluated over 50 trials. * denotes reproduced results. For reproduced results, the average row reports mean ± std over three random seeds.

Task	OpenVLA [13]	π_0-FAST [21]	π_0* [2]	π_0 + TACO* [31]	π_0 + Ours	$\pi_{0.5}$* [10]	$\pi_{0.5}$ + Ours
Soup and Sauce in Basket	60.0	74.0	78.0	82.0	84.0	90.0	**100.0**
Cheese and Butter in Basket	76.0	72.0	98.0	94.0	92.0	**100.0**	**100.0**
Turn on Stove and Place Moka	58.0	62.0	84.0	92.0	96.0	96.0	**98.0**
Black Bowl in Drawer	36.0	52.0	90.0	92.0	96.0	94.0	**100.0**
Mugs on Plates	32.0	54.0	84.0	82.0	82.0	**96.0**	**96.0**
Book in Caddy	82.0	82.0	96.0	94.0	96.0	**100.0**	92.0
Mug and Pudding on Plate	60.0	58.0	82.0	82.0	80.0	**94.0**	**94.0**
Soup and Cheese in Basket	70.0	72.0	98.0	96.0	94.0	96.0	**100.0**
Moka Pots on Stove	20.0	26.0	30.0	36.0	38.0	64.0	**70.0**
Mug in Microwave	46.0	50.0	76.0	88.0	86.0	**94.0**	**94.0**
Average	54.0 –	60.2 –	81.6 ± 0.8	83.8 (↑ 2.2) ± 0.2	84.4 (↑ 2.8) ± 0.4	92.4 ± 0.2	**94.4** (↑ 2.0) ± 0.3

Table 2: Success rates (%) on the SIMPLER benchmark. We compare our method on top of CogACT with prior VLA policies. For CogACT and CogACT + Ours, we report mean ± std over three random seeds.

Method	**Pick Coke Can**	**Move Near**	**Open/Close Drawer**	**Open Top Drawer and Place Apple**	***Average***
RT-1 [3]	85.7	44.2	73.0	6.5	52.4
RT-1-X [20]	56.7	31.7	59.7	21.3	42.4
RT-2-X [20]	78.7	77.9	25.0	3.7	46.3
Octo-Base [28]	17.0	4.2	22.7	0.0	11.0
OpenVLA [13]	18.0	56.3	63.0	0.0	34.3
CogACT [15]	91.3 ± 0.3	83.3 ± 0.6	71.8 ± 0.2	56.8 ± 0.1	75.8 ± 0.3
CogACT + Ours	**94.6** ± 0.2 (↑ 3.3)	**85.8** ± 0.2 (↑ 2.5)	**75.4** ± 0.3 (↑ 3.6)	**62.3** ± 0.2 (↑ 5.5)	**79.5** ± 0.2 (↑ 3.7)

Baselines. We mainly evaluate our framework on VLA policies with flow-matching or diffusion-based action heads. Specifically, we select π_0 [2] and $\pi_{0.5}$ [10] as the primary baseline models for LIBERO, and adopt CogACT [15] for experiments on SimplerEnv. We also compare with TACO [31], a test-time scaling method, to evaluate our method against existing test-time steering approaches. For a more comprehensive comparison, we further report the success rates of representative VLA policies on selected benchmarks, including OpenVLA [13], π_0-FAST [21], RT-1 [3], RT-1-X [20], RT-2-X [20], and Octo [28].

5.1.2 Results

The simulation results are reported in Tables 1 and 2. On LIBERO-10, our method improves both base policies. For π_0, the average success rate increases from 81.6% to 84.4%, outperforming the test-time scaling method TACO, which achieves 83.8%. For the stronger $\pi_{0.5}$ policy, our method further improves the average success rate from 92.4% to 94.4%. Task-level gains are especially clear on long-horizon and multi-stage tasks such as Turn on Stove and Place Moka, Black Bowl in Drawer, Moka Pots on Stove, and Soup and Sauce in Basket. These results show that online success memory provides reusable environment-specific action priors that help stabilize closed-loop execution.

On SimplerEnv, our method also improves CogACT from 75.8% to 79.5% on average, with consistent gains across all four tasks. The improvements are 3.3, 2.5, 3.6, and 5.5 points on Pick Coke Can, Move Near, Open/Close Drawer, and Open Top Drawer and Place Apple, respectively. The largest gain appears on the most challenging long-horizon task, suggesting that our retrieval-guided prior is effective under layout variations and visual perturbations.

5.2 Real-World Experiments

5.2.1 Setup

We evaluate our method on two real-world bimanual platforms: an OpenArm-based dual-arm system [7] and an ALOHA-PiPER system [34, 1]. We collect 100 training trajectories per task and evaluate four tasks: bowl stacking, cube handoff, and sequential test-tube placement on OpenArm,

Table 3: Performance comparison on the real-robot test tube placement task. The task is to pick up the test tubes one by one from left to right and place them onto the test tube rack.

Method	**Success Rate (%)**				**Avg. Len**
	1/4	**2/4**	**3/4**	**4/4**	
π_0 [2]	64.0	26.0	14.0	8.0	1.12
$\pi_{0.5}$ [10]	80.0	32.0	24.0	18.0	1.54
$\pi_{0.5}$ + Ours	90.0	48.0	32.0	24.0	1.94

Table 4: Component ablation on LIBERO-10. We report the average success rate (%) over all tasks. The ablation separately studies how the retrieved prior is constructed and how it is used for action generation. All intermediate-initialization variants use dynamic t_0. Gated Chunk Execution denotes retrieval-gated chunk execution with VLA fallback, rather than open-loop replay of a complete trajectory.

Variant	**Retrieved Prior**	**Prior Usage**	**Avg. Success**
Base $\pi_{0.5}$	–	Original sampler	92.4
Prior-construction ablation			
Top-1 Retrieval	Nearest success chunk	Intermediate init.	93.6
Top-K Soft Aggregation	Top-K weighted prior	Intermediate init.	94.0
Prior-usage ablation			
Gated Chunk Execution	Top-K + DTW prior	Execution / VLA fallback	57.8
Output Interpolation	Top-K + DTW prior	Post-hoc interpolation	93.0
Full Ours	Top-K + DTW prior	Intermediate init.	94.4

and bimanual T-shirt folding on ALOHA-PiPER. The tasks cover long-horizon manipulation, bimanual coordination, fine-grained placement, deformable-object manipulation, and appearance shifts. Hardware details, task definitions, and training/testing protocols are provided in Appendix D.

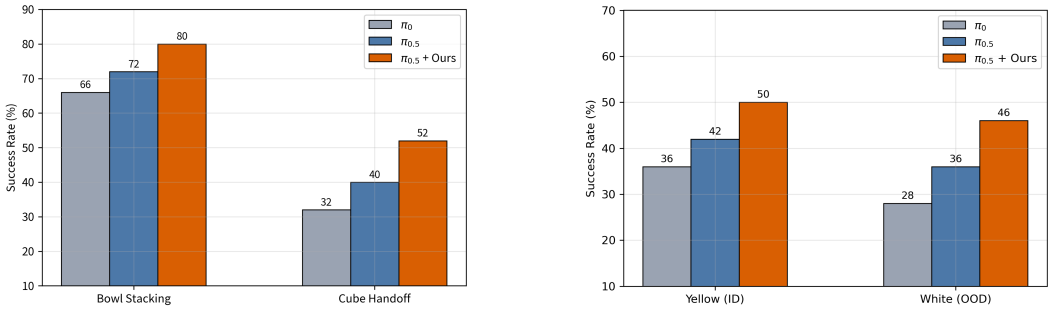
5.2.2 Results

OpenArm results. The OpenArm results are reported in Figure 3(a) and Table 3. As shown in Figure 3(a), $\pi_{0.5}$ + Ours improves the success rate from 72.0% to 80.0% on Bowl Stacking and from 40.0% to 52.0% on Cube Handoff. For Sequential Test-Tube Placement, Table 3 shows that our method improves all completion stages, increasing the full 4/4 success rate from 18.0% to 24.0% and the average completed length from 1.54 to 1.94. These results indicate that online success memory improves execution stability in long-horizon and bimanual manipulation tasks.

ALOHA-PiPER results. The ALOHA-PiPER T-shirt folding results are shown in Figure 3(b). On in-domain yellow T-shirts, $\pi_{0.5}$ + Ours improves the success rate from 42.0% to 50.0%. Under out-of-domain white T-shirts, our method improves the success rate from 36.0% to 46.0%. Averaged across both settings, it increases performance from 39.0% to 48.0%, indicating improved robustness to appearance shifts in deformable-object manipulation.

6 Ablation Studies and Analyses

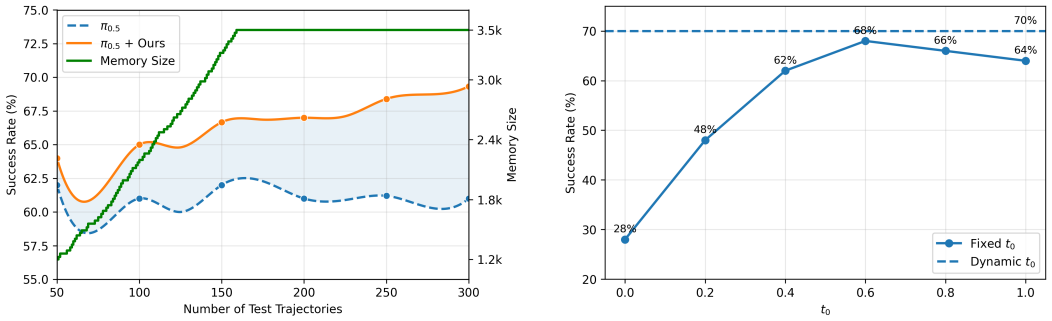
Continuous deployment analysis. We evaluate continuous deployment on the *Moka Pots on Stove* task from LIBERO-10. Starting from an empty memory, the policy is tested for 300 trajectories across different random seeds. During this process, verified successful observation–action segments are progressively written into the online memory and retrieved in later episodes to guide the frozen VLA. We use a bounded memory of 3.5k entries with FIFO replacement once the capacity is reached. As shown in Figure 4(a), the cumulative success rate of the original $\pi_{0.5}$ remains stable, while $\pi_{0.5}$ + Ours gradually improves as the memory accumulates reusable experience and then stabilizes at a higher level. Notably, the gain is maintained after the memory saturates, showing that our method can exploit recent and relevant successful experience under a finite memory budget. A detailed memory-capacity ablation is provided in Appendix E.



(a) OpenArm tasks.

(b) ALOHA-PiPER cloth folding.

Figure 3: Success rates on real-world robot tasks. (a) OpenArm results on Bowl Stacking and Cube Handoff. (b) ALOHA-PiPER results on bimanual T-shirt Folding.



(a) Cumulative success curve.

(b) Effect of t_0.

Figure 4: Analysis of continuous deployment and confidence-adaptive prior guidance.

Component ablation. We conduct component ablations on LIBERO-10 using $\pi_{0.5}$ as the frozen base policy. As shown in Table 4, Top-1 Retrieval improves the average success rate from 92.4% to 93.6%, showing that a retrieved successful chunk already provides useful test-time guidance. Top-K Soft Aggregation further increases the success rate to 94.0%, and the full method reaches 94.4%, indicating that multi-candidate aggregation and DTW-based filtering improve the retrieved prior. We also compare different prior-usage strategies with the same Top-K + DTW prior. Retrieval-Gated Chunk Execution with VLA Fallback (Gated Chunk Execution in the table) achieves only 57.8%, compared with 92.4% for the base policy. At each decision step, this baseline retrieves short historical action chunks from the current observation, directly executes the retrieved prior only when the cosine-similarity gate and DTW filter accept it, and otherwise falls back to the original VLA sampler. It then observes the new state and retrieves again; this is closed-loop chunk reuse rather than open-loop replay of a complete trajectory. At the default similarity threshold of 0.9992, historical chunks are executed at 40.7% of visited decision steps, with VLA fallback at the remaining steps. Despite this conservative gating and fallback, directly executing the retrieved prior substantially degrades performance. Output Interpolation improves over the base policy to 93.0%, but remains below intermediate initialization. These results show that injecting the prior into the generative sampler is more effective than post-hoc action reuse, as it allows the frozen VLA to refine actions under the current observation.

Effect of t_0. We analyze the effect of t_0 on prior-guidance strength using the Moka Pots on Stove task. Fixed t_0 shows clear sensitivity: a large t_0 underuses the retrieved prior, while a small t_0 over-constrains generation and weakens observation-conditioned refinement. The best fixed setting achieves a success rate of 68% at $t_0 = 0.6$, whereas dynamic t_0 further improves it to 70%, showing the advantage of confidence-adaptive guidance over manual tuning.

Effect of success-memory construction. Table 5 shows that memory quality is critical. Storing all trajectories without verification reduces the average success rate from 92.4% to 87.6%, indicating that

Table 5: Ablation of success-memory construction on LIBERO-10 and evaluation of the success discriminator.

Success-Memory Construction			
Memory Type	**Label Source**	**Progress-Peak Truncation**	**Avg. Success**
No Memory	–	–	92.4
Unverified Memory	None	×	87.6
Oracle Successful Full Trajectory	Environment Label	✓	**94.8**
Predicted Successful Full Trajectory	Discriminator	×	91.8
Predicted Successful Prefix Memory	Discriminator	✓	94.4
Success Discriminator Evaluation			
Metric	**Value**	**Metric**	**Value**
Accuracy	0.676	Precision	0.970
Recall	0.678	F1-score	0.798

noisy test-time experience can introduce harmful priors. Using predicted successful full trajectories also underperforms the base policy, suggesting that untrimmed trajectories may contain redundant or regressive segments. In contrast, our predicted successful prefix memory achieves 94.4%, close to the oracle memory result of 94.8%. These results confirm the importance of both reliable success verification and progress-peak truncation. Although the discriminator has moderate recall, its high precision of 0.970 is more desirable for memory construction, since a smaller but cleaner memory is preferable to one contaminated by false successful segments.

7 Conclusion

We propose an online success-memory guided test-time adaptation method for generative VLAs. During continuous deployment, successful observation–action segments are stored and retrieved as action priors to initialize the generative sampler. This enables a frozen VLA to reuse environment-specific experience without parameter updates. Progress-calibrated prefix selection and trajectory-consistency filtering organize deployment experience into reusable local evidence, while confidence-adaptive guidance balances historical priors with observation-conditioned generation. Experiments in simulation and real-world bimanual manipulation show improved stability and success rates, especially on long-horizon tasks. These findings highlight successful cross-episode experience as a practical resource for persistent, non-parametric adaptation of frozen generative policies.

8 Limitations

Our framework is designed for repeated deployment where the base policy can occasionally produce useful experience and task conditions remain relatively stable. Its benefits depend on progress-estimation quality; persistent estimation errors may affect memory reliability, while consistency filtering may favor frequently observed successful behaviors. Improving memory verification and management, preserving diverse successful strategies, and extending adaptation to more dynamic environments are promising directions for future work.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants Nos. 62576224 and 62606333; the Guangdong Provincial Key Laboratory of Computility Microelectronics (2024B1212010007); the Guangdong Basic and Applied Basic Research Foundation Project (2026A1515011733); the Characteristic Innovation Project of Ordinary Universities in Guangdong Province (2024KTSCX026); the Research and Development Fund Project of China Huaneng Group Co., Ltd. (HNKJ23-HF64).

References

[1] ALOHA 2 Team, Jorge Aldaco, Travis Armstrong, Robert Baruch, Jeff Bingham, Sanky Chan, Kenneth Draper, Debidatta Dwibedi, Chelsea Finn, Pete Florence, Spencer Goodrich, Wayne Gramlich, Torr

Hage, Alexander Herzog, Jonathan Hoech, Thinh Nguyen, Ian Storz, Baruch Tabanpour, Leila Takayama, Jonathan Tompson, Ayzaan Wahid, Ted Wahrburg, Sichun Xu, Sergey Yaroshenko, Kevin Zakka, and Tony Z. Zhao. ALOHA 2: An enhanced low-cost hardware for bimanual teleoperation. *arXiv preprint arXiv:2405.02292*, 2024. doi: 10.48550/arXiv.2405.02292. URL `https://arxiv.org/abs/2405.02292`.

[2] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[3] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[4] Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. Conrft: A reinforced fine-tuning method for vla models via consistency policy. *arXiv preprint arXiv:2502.05450*, 2025.

[5] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[6] Mingtong Dai, Lingbo Liu, Yongjie Bai, Yang Liu, Zhouxia Wang, Rui Su, Chunjie Chen, Liang Lin, and Xinyu Wu. Rover: Robot reward model as test-time verifier for vision-language-action model. *arXiv preprint arXiv:2510.10975*, 2025.

[7] Enactic, Inc. Openarm: A fully open-source humanoid robot arm for physical ai research. `https://openarm.dev/`, 2025. Accessed: 2026-05-05.

[8] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

[9] Zhi Hou, Tianyi Zhang, Yuwen Xiong, Haonan Duan, Hengjun Pu, Ronglei Tong, Chengyang Zhao, Xizhou Zhu, Yu Qiao, Jifeng Dai, et al. Dita: Scaling diffusion transformer for generalist vision-language-action policy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7686–7697, 2025.

[10] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[11] Suhyeok Jang, Dongyoung Kim, Changyeon Kim, Youngsuk Kim, and Jinwoo Shin. Verifier-free test-time sampling for vision language action models. *arXiv preprint arXiv:2510.05681*, 2025.

[12] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083. IEEE, 2019.

[13] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[14] Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. Robomonkey: Scaling test-time sampling and verification for vision-language-action models. *arXiv preprint arXiv:2506.17811*, 2025.

[15] Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

[16] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[17] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[18] Marius Memmel, Jacob Berg, Bingqing Chen, Abhishek Gupta, and Jonathan Francis. Strap: Robot sub-trajectory retrieval for augmented policy learning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations*, volume 2025, pages 63813–63832, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a06e129e01e0d2ef853e9ff67b911360-Paper-Conference.pdf.

[19] Soroush Nasiriany, Tian Gao, Ajay Mandlekar, and Yuke Zhu. Learning and retrieval from prior data for skill-based imitation learning. *arXiv preprint arXiv:2210.11435*, 2022.

[20] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[21] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

[22] Ralf Römer, Adrian Kobras, Luca Worbis, and Angela P Schoellig. Failure prediction at runtime for generative robot policies. *arXiv preprint arXiv:2510.09459*, 2025.

[23] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[24] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, February 1978. ISSN 0096-3518. doi: 10.1109/tassp.1978.1163055. URL http://dx.doi.org/10.1109/TASSP.1978.1163055.

[25] Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025.

[26] Mohammad Shokoohi-Yekta, Bing Hu, Hongxia Jin, Jun Wang, and Eamonn Keogh. Generalizing dtw to the multi-dimensional case requires an adaptive approach. *Data Mining and Knowledge Discovery*, 31 (1):1–31, Jan 2017. ISSN 1573-756X. doi: 10.1007/s10618-016-0455-0. URL https://doi.org/10.1007/s10618-016-0455-0.

[27] Shahram Najam Syed, Yatharth Ahuja, Arthur Jakobsson, and Jeff Ichnowski. Expres-vla: Specializing vision-language-action models through experience replay and retrieval. *arXiv preprint arXiv:2511.06202*, 2025.

[28] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[29] Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning, 2025. URL https://arxiv.org/abs/2506.15799.

[30] Yuxuan Wu, Guangming Wang, Zhiheng Yang, Maoqing Yao, Brian Sheil, and Hesheng Wang. Continually evolving skill knowledge in vision language action model. *arXiv preprint arXiv:2511.18085*, 2025.

[31] Siyuan Yang, Yang Zhang, Haoran He, Ling Pan, Xiu Li, Chenjia Bai, and Xuelong Li. Steering vision-language-action models as anti-exploration: A test-time scaling approach. *arXiv preprint arXiv:2512.02834*, 2025.

[32] Shaopeng Zhai, Qi Zhang, Tianyi Zhang, Fuxian Huang, Haoran Zhang, Ming Zhou, Shengzhe Zhang, Litao Liu, Sixu Lin, and Jiangmiao Pang. A vision-language-action-critic model for robotic real-world reinforcement learning. *arXiv preprint arXiv:2509.15937*, 2025.

[33] Jianchao Zhao, Chenhao Ding, SongLin Dong, Jiangyang Li, Qiang Wang, Yuhang He, and Yihong Gong. Shared & domain self-adaptive experts with frequency-aware discrimination for continual test-time adaptation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(34):28697–28705, March 2026. ISSN 2159-5399. doi: 10.1609/aaai.v40i34.40102. URL http://dx.doi.org/10.1609/aaai.v40i34.40102.

[34] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[35] Yichen Zhu, Zhicai Ou, Xiaofeng Mou, and Jian Tang. Retrieval-augmented embodied agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17985–17995, 2024.

[36] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

A Details of the Progress Estimator

Model overview. We use a pretrained VLAC critic [32] as the progress estimator for automatic trajectory evaluation and online success-memory construction. VLAC is a vision-language-action-critic model built upon InternVL, where action generation and process evaluation are unified in a multimodal autoregressive framework. In this work, we only use its critic capability. Given two visual observations and a language instruction, the critic predicts a signed progress change indicating whether the second observation is closer to the task goal than the first one. A positive value indicates forward task progress, a negative value indicates regression or deviation, and a value close to zero indicates little task-relevant change. Compared with single-state success classification, this pair-wise progress formulation provides a more fine-grained signal for long-horizon manipulation.

VLAC critic is trained from temporal supervision in successful task videos. Given a trajectory $O = (o_1, \ldots, o_T)$ with task instruction l_{task}, two frames o_i and $o_{i+\Delta t}$ are sampled, and the progress label is constructed from their temporal offset:

$$c_{i,i+\Delta t} = \frac{\Delta t}{T - i}. \tag{11}$$

Forward pairs correspond to positive progress, while reversed pairs provide negative progress samples. The training further includes static-frame filtering, forward/backward joint sampling, task-completion prediction, and semantically mismatched samples, which improve robustness to stagnation, regression, and task-irrelevant visual changes. Since this learning process mainly depends on visual states and language goals rather than a unified action space, VLAC can be trained with heterogeneous human and robot trajectory data and generalizes across different embodiments and scenes.

Role of the reference video. VLAC supports in-context progress understanding through a reference process. For unseen tasks, scenes, or embodiments, a language instruction alone may not fully specify the task stages and completion condition. A reference process provides an execution example that reveals the expected temporal structure, key visual transitions, and task logic. Formally, VLAC estimates progress with an additional reference sequence as

$$c_{i,i+\Delta t} = \text{VLAC}\left(o_i, o_{i+\Delta t}; l_{\text{task}}, O_{\text{ref}}, o_0\right), \tag{12}$$

where O_{ref} can be a robot or human demonstration, and o_0 denotes the initial observation of the current trajectory. The reference process helps align the current execution with a successful example and enables one-shot transfer to new task instances.

In our framework, we provide one successful demonstration video for each task as the reference process R. The reference video is not used as action supervision and does not update the policy. It only serves as an in-context example for the VLAC critic. For example, in pick-and-place tasks, it implicitly specifies the stage order such as approaching, grasping, moving, and placing. This helps the critic distinguish forward progress from stagnation, regression, and failure more reliably than using the language instruction alone.

Usage in our framework. For the i-th test trajectory, we evaluate progress at a fixed interval Δ. Let τ_m and τ_{m-1} denote two adjacent evaluation timesteps. Conditioned on the instruction $l^{(i)}$ and the reference process R, the interval-level progress is computed as

$$c^{(i)}_{\tau_m} = \Phi_\psi\left(o^{(i)}_{\tau_{m-1}}, o^{(i)}_{\tau_m}, l^{(i)}; R\right). \tag{13}$$

These interval-level scores are accumulated into a trajectory-level progress score. Instead of using the terminal progress, we take the maximum accumulated progress as the completion score, which is more robust to overshooting, collisions, or redundant motions after near completion. If the maximum progress exceeds a threshold, the trajectory is considered to contain reusable successful experience. We then store only the observation-action prefix before the progress peak into the online memory, filtering out failure segments, regressions, and post-success redundancy.

Why VLAC critic. VLAC critic is well suited to our setting because our goal is not episode-level success labeling, but conservative extraction of reusable successful segments. The quality of online memory directly affects the reliability of the retrieved action prior. Thus, the estimator should identify task-relevant progress while avoiding the inclusion of failed or regressive segments. VLAC provides

such a task-conditioned pair-wise progress signal and can use a reference video as an in-context task prior without training a task-specific classifier.

This design is also compatible with frozen-VLA test-time adaptation. The progress estimator is pretrained and external to the base policy, requiring no policy parameter update and no assumption about the action representation of the underlying generative VLA. Our discriminator analysis further supports this choice. Although the overall accuracy and recall are moderate, the precision reaches 0.970. This means that the estimator may miss some successful trajectories, but trajectories predicted as successful are highly reliable. For success-memory construction, high precision is more important than high recall: missing a successful segment only slows memory growth, whereas storing a failed segment can contaminate memory and induce misleading action priors. Empirically, storing all trajectories without verification reduces the average success rate from 92.4% to 87.6%, while the predicted successful prefix memory achieves 94.4%, close to the oracle memory result of 94.8%. This indicates that a smaller but cleaner memory is preferable to a larger noisy memory for retrieve-then-steer test-time adaptation.

B Prior Guidance for Diffusion Action Heads

The main paper presents confidence-adaptive prior guidance using a flow-matching action head. For VLA policies equipped with diffusion-based action heads, the retrieved elite action prior a_{elite} can also be used to guide action generation. The key idea is to first perturb a_{elite} to an intermediate noise level following the diffusion forward process, and then let the original diffusion action generator perform the remaining conditional denoising steps.

B.1 Diffusion Action Generation

For diffusion-based action heads, an action chunk is modeled as a random variable generated by gradually denoising Gaussian noise. Given the conditioning feature z_t corresponding to the current observation and task instruction, standard diffusion sampling starts from

$$x_N \sim \mathcal{N}(0, I), \tag{14}$$

and iteratively applies the reverse denoising process:

$$x_{n-1} \sim p_\theta(x_{n-1} \mid x_n, z_t), \qquad n = N, N-1, \ldots, 1. \tag{15}$$

The final action chunk is obtained as

$$\hat{a}_{t:t+H-1} = x_0. \tag{16}$$

Here, N denotes the number of diffusion sampling steps, H denotes the action prediction horizon, and z_t is the conditional feature extracted by the VLA from the current observation o_t and language instruction l. Since standard diffusion sampling starts entirely from random noise, the generated action may be sensitive to noise initialization and local state deviations. To exploit environment-specific experience stored in the online success memory, we use the retrieved elite prior a_{elite} to guide the diffusion denoising process.

B.2 Prior Perturbation and Conditional Denoising

For DDPM/DDIM-style diffusion action heads, the forward noising process can be written as

$$q(x_n \mid x_0) = \mathcal{N}\left(x_n; \sqrt{\bar{\alpha}_n} x_0, (1 - \bar{\alpha}_n) I\right), \tag{17}$$

or equivalently,

$$x_n = \sqrt{\bar{\alpha}_n} x_0 + \sqrt{1 - \bar{\alpha}_n}\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{18}$$

Here, $\bar{\alpha}_n$ denotes the cumulative signal coefficient in the diffusion noise schedule. As n increases, $\bar{\alpha}_n$ decreases and the sample moves from an action-like state toward a high-noise state.

When the retrieval module produces an elite action prior a_{elite}, we regard it as a local successful action endpoint and perturb it to an intermediate diffusion step n_0 using the original forward noising process:

$$x_{n_0} = \sqrt{\bar{\alpha}_{n_0}} a_{\text{elite}} + \sqrt{1 - \bar{\alpha}_{n_0}}\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{19}$$

The intermediate step n_0 is determined by the confidence-adaptive guidance schedule introduced in the main text. Intuitively, a more reliable retrieved prior corresponds to a lower noise level and thus stronger prior guidance, whereas a less reliable prior corresponds to a higher noise level and leaves more freedom for the diffusion model.

The sampler then starts from x_{n_0} instead of x_N and performs the remaining reverse denoising steps:

$$x_{n-1} \sim p_\theta(x_{n-1} \mid x_n, z_t), \qquad n = n_0, n_0 - 1, \ldots, 1. \tag{20}$$

The final generated action chunk is

$$\hat{a}_{t:t+H-1} = x_0. \tag{21}$$

This procedure provides a diffusion-compatible form of prior guidance: a_{elite} is first mapped to an intermediate noisy state on the diffusion trajectory, and the original diffusion policy then completes conditional denoising under the current observation feature z_t. Therefore, the model does not simply replay the retrieved action; instead, it generates an action chunk near the neighborhood of historical successful behavior while remaining conditioned on the current state. If the online success memory is empty or the retrieved candidates fail the similarity and trajectory-consistency filters, the system does not construct a_{elite} and falls back to standard diffusion sampling from $x_N \sim \mathcal{N}(0, I)$.

C Component-Aware Aggregation of Action Priors

Action component decomposition. Since the action spaces of different robotic platforms may contain different types of control variables, directly applying element-wise linear averaging to the entire action vector is not always appropriate. We therefore adopt a component-aware aggregation strategy for constructing the elite action prior. For the h-th prediction step of the i-th candidate action chunk, we write

$$a_{i,h} = (\Delta p_{i,h}, \Delta r_{i,h}, g_{i,h}), \tag{22}$$

where $\Delta p_{i,h}$ denotes action components in Euclidean spaces, such as end-effector position increments or joint-angle increments; $\Delta r_{i,h}$ denotes orientation or orientation-increment components; and $g_{i,h}$ denotes the gripper command.

Aggregation of Euclidean action components. For action components that lie in Euclidean spaces, including end-effector positions, joint angles, and their corresponding increments, we directly apply similarity-weighted averaging:

$$\Delta p_{\text{elite},h} = \sum_{i \in \mathcal{I}} w_i \Delta p_{i,h}. \tag{23}$$

This operation is suitable for action dimensions with a linear structure and forms a smooth local action prior from multiple similar successful behaviors.

Aggregation of orientation components. For orientations and orientation increments, direct linear averaging may violate the geometry of the rotation space. Therefore, we first map each axis-angle orientation increment to the rotation group $SO(3)$:

$$R_{i,h} = \mathrm{Exp}(\Delta r_{i,h}), \tag{24}$$

where $\mathrm{Exp}(\cdot)$ denotes the exponential map from the Lie algebra $\mathfrak{so}(3)$ to $SO(3)$. We then compute the weighted geodesic mean on $SO(3)$:

$$R_{\text{elite},h} = \arg\min_{R \in SO(3)} \sum_{i \in \mathcal{I}} w_i \left\| \mathrm{Log}(R_{i,h}^\top R) \right\|_2^2, \tag{25}$$

where $\mathrm{Log}(\cdot)$ denotes the logarithm map from $SO(3)$ to $\mathfrak{so}(3)$. Finally, the averaged rotation is mapped back to the axis-angle representation:

$$\Delta r_{\text{elite},h} = \mathrm{Log}(R_{\text{elite},h}). \tag{26}$$

This procedure ensures that orientation aggregation respects the geometry of the rotation manifold and avoids the inconsistency that may arise from naive linear averaging.

Aggregation of gripper actions. Since different robotic platforms parameterize gripper commands differently, we use an adaptive aggregation rule for gripper actions. For continuous gripper commands, such as gripper width, finger-joint position, or normalized continuous control values, we apply weighted averaging:

$$g_{\mathrm{elite},h} = \sum_{i \in \mathcal{I}} w_i g_{i,h}, \tag{27}$$

and clip the result to the valid action range. For discrete gripper commands, such as open-or-close commands, we use weighted voting:

$$P_h(c) = \sum_{i \in \mathcal{I}} w_i \mathbb{I}[g_{i,h} = c], \tag{28}$$

$$g_{\mathrm{elite},h} = \arg\max_c P_h(c). \tag{29}$$

When the retrieved gripper commands exhibit strong conflicts between opening and closing directions, we adopt a conservative fallback strategy: the gripper command from the most similar nearest-neighbor candidate is used. This avoids averaging contradictory gripper actions into an ambiguous gripper prior.

Final action prior. After component-aware aggregation, the elite action prior at the h-th prediction step is written as

$$a_{\mathrm{elite},h} = (\Delta p_{\mathrm{elite},h}, \Delta r_{\mathrm{elite},h}, g_{\mathrm{elite},h}). \tag{30}$$

By concatenating the priors over all prediction steps, we obtain the complete elite action prior:

$$a_{\mathrm{elite}} = \{a_{\mathrm{elite},h}\}_{h=1}^{H}. \tag{31}$$

This design allows our method to support end-effector pose control, joint-space control, and different gripper parameterizations, thereby avoiding overfitting the method formulation to a specific dataset or robotic platform.

D Real-World Robot Experiment Details

D.1 Hardware Platforms and Camera Configuration

We evaluate our method on two real-world bimanual robot platforms: an OpenArm-based dual-arm system and an ALOHA-PiPER system. The OpenArm platform consists of two 7-DoF humanoid robot arms equipped with parallel grippers, while the ALOHA-PiPER platform follows an ALOHA-style bimanual setup and uses two 6-DoF AgileX PiPER arms with two-finger grippers. These two platforms provide complementary testbeds for evaluating our method across different arm kinematics, gripper designs, and workspace layouts.

Both platforms take multi-view RGB images and proprioceptive states as policy inputs. Specifically, each platform is equipped with three camera views: two wrist-mounted cameras attached to the left and right grippers, and one external third-person camera. The wrist cameras provide close-up observations of local manipulation regions, such as grasping, handoff, and placement areas, while the third-person camera provides a global view of the workspace, including object layouts, arm configurations, and overall task progress.

The placement of the third-person camera differs slightly between the two platforms due to hardware constraints. On the ALOHA-PiPER platform, the external camera is mounted above the center region between the two arms on the same side as the robot arms, providing a direct overhead view of the T-shirt manipulation area. On the OpenArm platform, due to the limited mounting space on the robot side, the external camera is placed above the center region between the two arms but on the opposite side of the workspace. Although the camera placements are different, both configurations provide complementary global observations together with the two wrist-mounted views, enabling the policy to perceive both fine-grained local interactions and long-horizon task progress.

D.2 Task Definitions

We design four real-world manipulation tasks to evaluate the proposed method, covering long-horizon object manipulation, bimanual coordination, fine-grained placement, and deformable-object manipulation.

Bowl Stacking. In this task, the two robot arms grasp one bowl from each side of the workspace and stack the two bowls in the center region. The task requires the policy to coordinate both arms to complete grasping, transportation, alignment, and stacking. This task mainly evaluates long-horizon object manipulation, bimanual spatial coordination, and relative pose alignment between objects. The main challenges come from unstable bowl grasps, the need for accurate rim alignment, and the risk of collision or slippage during the stacking stage.

Cube Handoff. In this task, the right arm first picks up a red cube from a bowl on the right side of the workspace, transfers it to the left arm, and the left arm then places it into a bowl on the left side. This task evaluates bimanual handoff ability, including grasping, approaching, pose alignment between two grippers, object transfer, release timing, and final placement. Compared with single-arm pick-and-place tasks, Cube Handoff requires more precise coordination between the two end-effectors. A small error in relative gripper pose or release timing can easily cause the cube to drop.

Sequential Test-Tube Placement. In this task, the right arm sequentially picks up four test tubes arranged on the table from left to right and places them into a test-tube rack. This task evaluates long-horizon sequential manipulation, fine-grained grasping, and precise placement into narrow slots. Since test tubes are thin and elongated objects, they are difficult to grasp stably, and the rack holes impose strict requirements on placement position and orientation. To improve task success and better exploit the bimanual setup, we keep the left arm stationary during execution and orient its wrist camera toward the test-tube rack. This provides a fine-grained local view of the rack region, helping the policy observe the target holes more accurately. The main challenges include accumulated errors over repeated operations, unstable grasping of thin objects, precise insertion, and state drift during long-horizon execution.

Bimanual T-shirt Folding. On the ALOHA-PiPER platform, we evaluate a bimanual T-shirt folding task. The two arms need to collaboratively grasp key regions of the T-shirt and complete the folding process. Unlike rigid-object manipulation, T-shirt folding involves deformable-object dynamics, where the object shape changes continuously during grasping, dragging, and folding. This task therefore requires robust visual understanding and closed-loop action adjustment. Demonstrations are collected using a yellow T-shirt, while testing is conducted under both in-domain and out-of-domain settings. The in-domain setting uses yellow T-shirts, whereas the out-of-domain setting uses T-shirts with unseen colors. This design evaluates the robustness of the policy to visual appearance shifts in deformable-object manipulation. The main challenges include uncertain cloth deformation, self-occlusion, localization of key grasping regions, and visual distribution shifts caused by color changes.

D.3 Training and Testing Protocol

For each real-world task, we collect 100 demonstration trajectories to fine-tune the base VLA policy. We use the same basic training configuration across all tasks, with a batch size of 64 and a learning rate of 2.5×10^{-5}. The number of training steps is adjusted according to the task horizon and manipulation complexity, ranging from 8k to 10k steps. For relatively shorter tasks, such as Bowl Stacking and Cube Handoff, the policy is trained for around 8k steps. For longer-horizon or more fine-grained tasks, such as Sequential Test-Tube Placement and Bimanual T-shirt Folding, the number of training steps is increased toward 10k steps to better cover the full task procedure and key manipulation stages.

During evaluation, the base VLA policy is kept frozen, and no additional gradient updates or online fine-tuning are performed. Each real-world task is evaluated over 50 trials. For fair comparison, the baseline policy and our method are tested under the same set of initial states. We only enable the proposed online success-memory mechanism on top of the frozen policy. During the evaluation of each task, the online memory is initialized and continuously updated according to our success-memory construction mechanism. Specifically, the robot executes action chunks generated by the frozen VLA policy during continuous test-time deployment, and reusable observation-action segments are stored into the online success memory when successful experience is identified. In subsequent trials, the system retrieves relevant successful experience according to the current observation and uses the retrieved action prior to guide the generative action sampling process. This protocol ensures that the performance improvement comes from non-parametric test-time memory retrieval and prior-guided

Table 6: Effect of memory capacity on continuous deployment. We evaluate $\pi_{0.5}$ + Ours on the *Moka Pots on Stove* task for 300 test trajectories. C denotes the maximum number of stored observation-action entries. When the memory exceeds C, FIFO eviction is applied.

Memory Capacity C	Eviction	Final Mem. Size	Final Cum. SR (%)
0	–	0	61.0
1k	FIFO	1.0k	64.0
2k	FIFO	2.0k	66.2
3k	FIFO	3.0k	68.5
4k	FIFO	4.0k	70.2
5k	FIFO	5.0k	70.8
Unlimited	–	All	71.2

Table 7: Relative inference time normalized by the frozen base policy. "Ours w/o Retrieval" measures the prior-guided sampler after the elite prior is available, while "Ours Full" includes retrieval, filtering, prior aggregation, and action generation.

Method	**Retrieval Included**	**Relative Time**
Base VLA	–	$1.00\times$
Ours w/o Retrieval	No	$0.95\times$
Ours Full	Yes	$1.10\times$

generation, rather than from additional policy training, parameter updates, or different initial-state distributions.

E Effect of Memory Capacity

As shown in Table 6, increasing the memory capacity consistently improves the final cumulative success rate, indicating that a larger memory provides more reusable successful experience for retrieval-guided action generation. The improvement is more pronounced when the capacity increases from 0 to 3k entries, while the gain gradually saturates beyond 4k. In particular, the 5k memory budget achieves a final cumulative success rate of 70.8%, which is close to the unlimited-memory setting of 71.2%. This suggests that our method does not rely on an ever-growing memory, but can achieve near-saturated performance with a bounded memory budget and FIFO replacement.

F Inference-Time Analysis

We further analyze the inference-time overhead introduced by the proposed retrieve-then-steer mechanism. All results are reported as relative inference time normalized by the frozen base policy. The runtime includes action generation for the current decision step. For our method, we report two settings: one excluding retrieval overhead, which measures only the prior-guided generative sampling process, and one including the full retrieval, filtering, aggregation, and sampling pipeline.

As shown in Table 7, the proposed method does not introduce heavy inference overhead. When the retrieval cost is excluded, our prior-guided sampler is slightly faster than the base policy, reducing the normalized inference time from $1.00\times$ to $0.95\times$. This is because the retrieved elite prior initializes the generative process from an intermediate state, allowing the sampler to perform fewer effective generation steps than the original noise-starting sampler.

When retrieval, trajectory-consistency filtering, and prior aggregation are included, the full method requires $1.10\times$ the inference time of the base policy. This moderate overhead mainly comes from nearest-neighbor search and candidate filtering in the online success memory. Since these operations are non-parametric and do not require additional forward passes through the VLA or parameter updates, the overall runtime remains lightweight. Combined with the success-rate gains reported in the main experiments, these results suggest that the proposed retrieve-then-steer mechanism offers a favorable trade-off between deployment efficiency and closed-loop reliability.

Table 8: Hyperparameter sensitivity on LIBERO-10. We report the average success rate (%) over all tasks. The default setting is highlighted in gray.

Hyperparameter	**Value**	**Avg. Success (%)**
Base $\pi_{0.5}$	–	92.4
Top-K retrieval size	$K = 1$	93.6
	$K = 3$	93.8
	$K = 5$	94.1
	$K = 10$	94.4
	$K = 20$	94.2
Similarity threshold γ_{sim}	0.9988	93.7
	0.9990	94.1
	0.9992	94.4
	0.9995	94.0
	0.9998	93.5
Aggregation temperature τ	0.01	93.8
	0.03	94.2
	0.05	94.4
	0.10	94.1
	0.20	93.9

Table 9: Default hyperparameters for adaptive prior guidance. These parameters are kept fixed across experiments and are empirically stable within reasonable ranges.

Hyperparameter	**Value**
Reference similarity s_{ref}	γ_{sim}
Similarity scale s_{scale}	5×10^{-4}
Clipping range $c_{\max}$	50
Similarity coefficient α	1.0
DTW-dispersion coefficient β	0.05
Mapping sharpness γ	2.0

G Hyperparameter Sensitivity

We analyze the sensitivity of the proposed retrieve-then-steer framework to key hyperparameters in the retrieval and prior-construction pipeline. Unless otherwise specified, experiments are conducted on LIBERO-10 using $\pi_{0.5}$ as the frozen base policy. We vary one hyperparameter at a time while keeping the others fixed to the default setting: $K = 10$, $\tau = 0.05$, and $\gamma_{\text{sim}} = 0.9992$. The base policy achieves an average success rate of 92.4%.

As shown in Table 8, the proposed method is relatively robust to a broad range of hyperparameter choices. First, increasing the retrieval size from $K = 1$ to $K = 10$ consistently improves performance, indicating that aggregating multiple successful chunks provides a more reliable action prior than using a single nearest neighbor. However, further increasing K to 20 slightly reduces the success rate, likely because less relevant trajectories are introduced into the candidate set.

Second, the similarity threshold γ_{sim} controls the trade-off between prior coverage and prior quality. A lower threshold accepts more retrieved candidates but may include mismatched action chunks, while an overly strict threshold rejects useful candidates and causes the method to fall back to the base sampler more frequently. The default value $\gamma_{\text{sim}} = 0.9992$ achieves the best balance between filtering unreliable retrievals and preserving sufficient reusable experience.

Third, the aggregation temperature τ affects the sharpness of the similarity-based soft weights. A very small temperature makes the aggregation close to nearest-neighbor selection, whereas a large temperature assigns nearly uniform weights to retrieved candidates. The best performance is obtained at $\tau = 0.05$, suggesting that softly emphasizing the most relevant successful chunks while still aggregating multiple candidates yields a more stable prior.

For adaptive prior guidance, we use the default configuration in Table 9. We set s_{ref} to the same value as the retrieval threshold γ_{sim}, since both measure whether a retrieved candidate is sufficiently close to the current observation. The scale $s_{\mathrm{scale}} = 5 \times 10^{-4}$ is used to magnify small differences in high cosine-similarity regimes. The clipping range $c_{\max} = 50$ prevents extreme confidence values from dominating the mapping. The coefficients $\alpha = 1.0$ and $\beta = 0.05$ balance state-level similarity and action-level dispersion, while $\gamma = 2.0$ controls the sharpness of the confidence-to-guidance mapping. We find that these parameters are not sensitive within reasonable ranges, and the default setting works consistently across tasks.

Overall, these results suggest that the performance gain does not depend on a narrowly tuned hyperparameter choice, and the default configuration provides a stable trade-off between retrieval coverage, memory quality, and action-prior robustness.

H Limitations and Future Work

Although our retrieve-then-steer framework improves the reliability of frozen generative VLAs, it still has several limitations. First, the method is designed for persistent deployment in relatively stable or slowly changing environments. Its effectiveness depends on the existence of reusable cross-episode experience; when object layouts, camera viewpoints, task goals, or robot calibration change rapidly, previously stored action priors may become less informative or even misleading. Second, the method requires the base policy to occasionally produce successful trials so that the online memory can be initialized and expanded. If the frozen VLA is far from competent on a target task, the memory may grow slowly and provide limited benefit.

Third, memory quality depends on the reliability of the progress or success estimator. Although our progress-calibrated prefix selection is designed to avoid storing failed, regressive, or post-success redundant segments, inaccurate progress estimation may still introduce noisy entries or discard useful successful experience. This issue is especially important because a small number of false successful entries can contaminate the retrieved prior and affect later generations. Fourth, retrieval based on visual similarity and trajectory consistency cannot fully resolve state aliasing. Visually similar states may require different actions due to subtle differences in object pose, contact state, occlusion, or task phase, which may lead to mismatched action priors even after similarity gating and consistency filtering.

Finally, our current memory management uses simple bounded storage and replacement strategies. While this is sufficient for the evaluated settings, larger-scale deployment may require more structured memory organization, long-term forgetting, and task-aware indexing. One possible extension is to detect recurring deployment contexts and reuse the corresponding success memories, drawing on domain detection in continual test-time adaptation for visual recognition [33]. Future work will explore more reliable success verification, uncertainty-aware retrieval, scalable memory management, and adaptation to more dynamic environments with changing tasks and layouts.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction clearly state the persistent-deployment setting, the proposed retrieve-then-steer framework, and the claim that online success memory improves test-time reliability for frozen generative VLAs. The scope is also reflected in the experiments on simulation benchmarks and real-world bimanual manipulation tasks.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer:[Yes].

   Justification: The paper includes a dedicated limitation paragraph discussing the assumptions of relatively stable deployment environments, the dependence on the progress/success estimator, and the possibility of mismatched retrieved action priors.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A].

Justification:The paper does not present formal theoretical results, theorems, or proofs. The equations are used to define the proposed memory construction, retrieval, aggregation, and prior-guided sampling procedures.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes].

Justification: The paper describes the benchmarks, base policies, evaluation protocol, memory construction procedure, retrieval and filtering mechanisms, prior aggregation, and sampling guidance used in the main experiments. Additional details on real-world hardware, task definitions, and training/testing protocols are provided in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: We provide anonymized code and reproduction instructions in the supplemental material, including the implementation of online success-memory construction, retrieval-based action prior aggregation, and confidence-adaptive prior guidance. The released materials include instructions for reproducing the main simulation experiments and evaluating the proposed method on the reported benchmarks.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The paper specifies the simulation benchmarks, real-world platforms, task settings, base policies, number of trials, real-world demonstration collection, and main training/testing protocol. The appendix further reports details such as the number of real-world demonstrations, batch size, learning rate, training steps, and frozen-policy evaluation protocol.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes].

Justification: The paper reports appropriate uncertainty information for the main simulation experiments. Specifically, for reproduced simulation results on LIBERO-10 and SimplerEnv, we report mean $\pm$ standard deviation over three random seeds, where the variability is computed across independent evaluation runs under fixed and reproducible benchmark settings. For real-world robot experiments, due to the high cost and time required for large-scale repeated physical trials, we do not report full error bars in the main figures. Instead, each real-world task is evaluated over 50 trials under the same set of initial states for fair comparison. We report the number of trials and success rates, and the corresponding binomial standard error can be estimated as $\sqrt{p(1-p)/50}$ for a reported success rate p. This provides an appropriate measure of uncertainty for real-world success/failure evaluations while accounting for the practical cost of physical robot experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: We provide the compute resources used for the experiments in the appendix, including the GPU type, memory, approximate training and evaluation time, and total compute budget. These details are reported to support reproducibility of both the simulation and real-world experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes].

Justification: The research studies robotic manipulation policies in simulation and controlled laboratory settings, and does not involve sensitive personal data, human-subject experiments, or deceptive data collection. The submission is anonymized and the experiments are described for scientific evaluation.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [No].

Justification: The current paper focuses on technical contributions and experimental validation, and does not include a dedicated broader-impact discussion. A future revision should discuss potential benefits for reliable robot manipulation as well as risks from deploying more autonomous robots in unsafe, unstructured, or insufficiently monitored environments.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A].

Justification: The paper cites the existing benchmarks, models, and robotic platforms used in the experiments, including LIBERO, SimplerEnv, OpenVLA, $\pi_0/\pi_{0.5}$, CogACT, and related robot assets. However, the current manuscript does not explicitly list the licenses and terms of use for these assets, which should be added in the appendix.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: The paper properly credits the existing benchmarks, models, and robotic platforms used in the experiments, including LIBERO, SimplerEnv, OpenVLA, $\pi_0/\pi_{0.5}$, CogACT, and related robot assets. We provide the corresponding licenses and terms of use in the appendix, and all assets are used in accordance with their intended research purposes.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A].

    Justification: The paper introduces a method and reports newly collected real-world demonstrations for evaluation, but it does not currently release a new dataset, benchmark, model checkpoint, or software package as a public asset. If these assets are released later, documentation and license information should be provided.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing experiments or human-subject studies. The real-world data are robot demonstration and evaluation trajectories collected for manipulation tasks.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The paper does not involve human-subject research, crowdsourcing, or participant studies. Therefore, IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A].

    Justification: The core method does not use an LLM as an important, original, or non-standard component. The method operates on frozen VLA policies and uses online success memory, retrieval, filtering, aggregation, and generative prior guidance.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.