VLA-Corrector: Lightweight Detect-and-Correct Inference for Adaptive Action Horizon

Yi Pan[1] **Miao Pan**[1] **Qi Lu**[1] **Jiaming Huang**[1] **Man Zhang**[1]
Siteng Huang[3] **Xin Li**[3] **Jie Zhang**[1]
Yongliang Shen[1] **Xuhong Zhang**[2,1] **Wenqi Zhang**[2,1,†]
[1]School of Software Technology, Zhejiang University, OmniAI Group
[2]Ningbo Global Innovation Center, Zhejiang University
[3]Alibaba DAMO Academy
Yi Pan: panyi0304@gmail.com
Wenqi Zhang: zhangwenqi@zju.edu.cn, [†]Corresponding author.

Abstract

Vision-Language-Action (VLA) foundation models have recently achieved strong progress in embodied intelligence. To reduce policy-call frequency while preserving temporal coherence, most generative policies adopt an **action chunk** mechanism, executing multiple future actions in an open-loop manner under a fixed **action horizon**. However, this "predict-then-blindly-execute" paradigm sacrifices closed-loop reactivity: in contact-rich physical interactions, even small local perturbations can rapidly amplify within the open-loop blind spot, leading to compounding errors and ultimately task failure. To address this limitation, we propose **VLA-Corrector**, a lightweight corrective inference framework for action-chunked VLA policies. Without modifying the backbone policy weights, VLA-Corrector introduces a lightweight **Latent-space Vision Monitor** (LVM) that continuously compares predicted and actual visual feature evolution, enabling online detection of visual dynamics deviations. Once persistent deviation is detected, the system triggers a truncation event, discards the remaining stale actions, and invokes corrective replanning via **Online Gradient Guidance** (OGG). The detect-and-correct mechanism of VLA-Corrector naturally induces an **event-triggered adaptive action horizon**: it preserves long-horizon execution when the current chunk remains reliable, and invokes short-horizon corrective replanning when execution begins to drift. In doing so, VLA-Corrector mitigates the trade-off imposed by static horizons between execution robustness and policy-call frequency. It can be integrated into different VLA models without further retraining the VLA backbone, interrupting compounding errors while preserving much of the efficiency benefit of action chunking and substantially improving robustness in long-horizon, contact-rich robotic manipulation tasks. Our code is available at https://github.com/ZJU-OmniAI/vla-corrector.

1 Introduction

Vision-Language-Action (VLA) foundation models have emerged as a promising path toward general-purpose robot control, unifying perception, language, and action generation within a single framework [59, 33, 22, 2, 25, 1, 16, 31, 44, 4, 19]. Modern VLA policies often rely on generative action models, such as diffusion models and flow matching, to capture the high-dimensional and multi-modal nature of continuous robot actions [2, 7, 18, 45, 27]. However, the per-step latency of generative models precludes closed-loop replanning, creating a fundamental tension between action expressiveness and high-frequency feedback control.

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

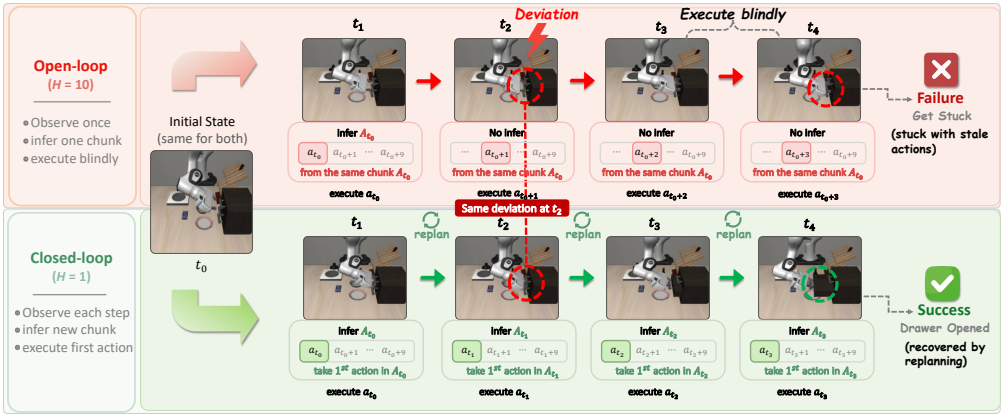


Figure 1: **Open-loop vs. Closed-loop execution.** The top row ($H = 10$) illustrates how blindly executing long action chunks leads to compounding errors, causing the robot to get stuck during a drawer-opening task. In contrast, the bottom row ($H = 1$) demonstrates strict closed-loop execution, which maintains environmental reactivity and yields a smooth, successful manipulation.

A common engineering compromise is to use an **action chunk** [57]: in a single forward pass, the policy predicts a sequence of future actions, while the controller only executes the first H steps as the **action horizon** [20, 30, 36, 7, 21, 39]. This design amortizes policy-call frequency and improves temporal smoothness, but it also creates an **open-loop blind spot** [46, 50] that introduces two compounding risks. **First, the policy lacks real-time reactivity**. Fresh observations arrive at every control step, yet the system ignores them until the horizon ends, leaving it unable to respond to unexpected slippage, collision, or pose drift as they occur. **Second, errors can accumulate beyond the point that replanning can recover from during the blind spot**. If deviations go uncorrected long enough, the robot may drift into an out-of-distribution state that the policy has rarely encountered during training, at which point even the next replanning call cannot steer execution back to the intended trajectory [46, 30, 36, 48, 51], and the task ultimately fails. Crucially, both risks worsen as the horizon grows longer. As illustrated in Fig. 1, under a long horizon ($H = 10$), the robot keeps executing stale actions after a deviation and eventually gets stuck; in contrast, with $H = 1$, the policy replans at every step and avoids the same error accumulation.

Yet $H = 1$ is not a practical solution: it requires a full VLA inference at every control step, negating the efficiency gains that action chunking was designed to provide. The contrast between these two extremes thus exposes a fundamental trade-off: smaller horizons improve closed-loop responsiveness but multiply policy-call frequency, while larger horizons amortize computation but widen the blind spot where errors silently accumulate.

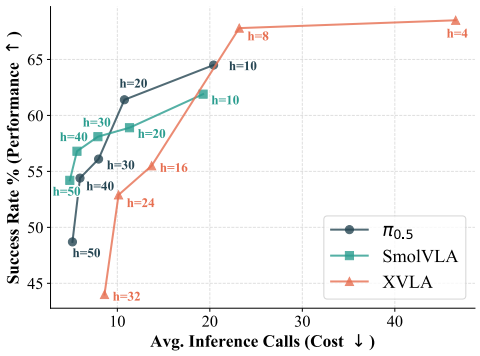


Figure 2: **Performance–efficiency trade-off across fixed action horizons.** Smaller horizon achieves higher success rates, while larger horizon preserves the chunking efficiency.

To quantify this trade-off, we conduct systematic experiments comparing task success rate and policy frequency across a range of action horizons on three VLA backbones. As shown in Fig. 2, larger horizons substantially reduce policy calls but also lead to lower success rates, and this trend is consistently observed across all three backbones. For $\pi_{0.5}$ [16], for example, increasing the horizon reduces policy calls by about $4\times$, but lowers success from about 64% to below 49%; similar trends hold for SmolVLA [38] and X-VLA [58]. Furthermore, since the best horizon depends on task difficulty, environmental dynamics, and sim-to-real mismatch, it is difficult to predefine a single static horizon that remains optimal across scenarios. This suggests that the key is not to choose a better fixed horizon, but to decide when the current chunk should stop being trusted.

The above analysis raises two key questions: (1) How to **detect execution deviations in time** and **terminate stale actions** before errors compound beyond recovery; (2) How to **correct the trajectory after truncation**—since naive replanning alone is often insufficient: the VLA may re-generate actions that still fail to escape the deviated state, leaving the robot trapped again.

Motivated by these insights, we propose **VLA-Corrector**, a lightweight corrective inference framework that addresses both questions. For the first, VLA-Corrector introduces a **Latent-space Vision Monitor** (LVM) that continuously compares predicted and actual visual evolution during open-loop execution; once persistent deviation is detected, it triggers an **interrupt event** that immediately truncates the remaining stale actions and invokes a new action generation. For the second, rather than relying on naive replanning, VLA-Corrector activates **Online Gradient Guidance** (OGG), which exploits the discrepancy between predicted and observed visual dynamics to inject a corrective gradient during new action generation—actively steering the robot back toward the intended trajectory, instead of simply hoping that re-inference will naturally recover. Together, LVM-triggered truncation and OGG-guided re-inference transform a fixed action horizon into an **adaptive action horizon** with **self-correcting re-inference**, preserving long-horizon efficiency and short-horizon responsiveness.

These two mechanisms jointly improve success-per-call efficiency: the robot avoids wasting long chunks on stale actions, so each policy query contributes more effectively to task completion. For $\pi_{0.5}$ at horizon 50, success rises from 48.7% to 58.7% while average policy calls drop from 5.15 to 4.98—a $+24.6\%$ success-per-call efficiency gain. Gains are consistent across all three backbones and grow larger at longer horizons, where the open-loop blind spot is widest. Beyond robustness, VLA-Corrector also improves sample efficiency: on LIBERO, a few-shot fine-tuned model augmented with VLA-Corrector reaches 97.8% average success, *surpassing* the fully fine-tuned baseline (96.9%).

Our contributions are summarized as follows:

• We systematically quantify the performance–efficiency trade-off induced by fixed action horizon in action-chunked VLA policies, showing that the open-loop blind spot consistently affects robustness across different VLA backbones.

• We propose VLA-Corrector, a lightweight inference-time framework that augments a frozen VLA backbone with latent-space monitoring, event-triggered truncation, and guidance toward recovery-oriented inference.

• We validate VLA-Corrector across simulation and real-world tasks, showing that it achieves stronger robustness, higher success-per-call efficiency, and adaptive correction behavior.

2 Preliminaries

VLA Policies with Action Chunks. In embodied continuous control, the current visual observation o_t is first encoded into a latent representation $Z_t^{\text{real}} = \mathcal{E}(o_t)$ by a visual encoder $\mathcal{E}$. To balance policy-call frequency and action smoothness, modern generative VLA policies parameterized by θ commonly predict an **action chunk** in a single inference call,

$$A_t = [a_t, a_{t+1}, \ldots, a_{t+C-1}] \sim \pi_\theta(\cdot \mid Z_t^{\text{real}}, l), \tag{1}$$

where a_t denotes a single control action and l is the language instruction. During deployment, only the first H actions, with $H \leq C$, are executed sequentially, and we refer to this execution window as the *action horizon*. The corresponding execution queue is $Q_t = [a_t, a_{t+1}, \ldots, a_{t+H-1}]$, during which the controller does not query the VLA policy again.

3 VLA-Corrector: A Lightweight Corrective Inference Framework

VLA-Corrector addresses the open-loop blind spot of action-chunked VLA policies. During the action horizon, fresh observations are available, but the policy does not use them until the horizon ends. We therefore decouple *action generation* from *execution monitoring*: the VLA backbone generates action chunks, while VLA-Corrector monitors whether execution remains on track and intervenes only when drift emerges. We organize the method as follows: **(1) Corrector Training** describes the external latent dynamics corrector; **(2) LVM Detection** introduces online visual deviation detection;

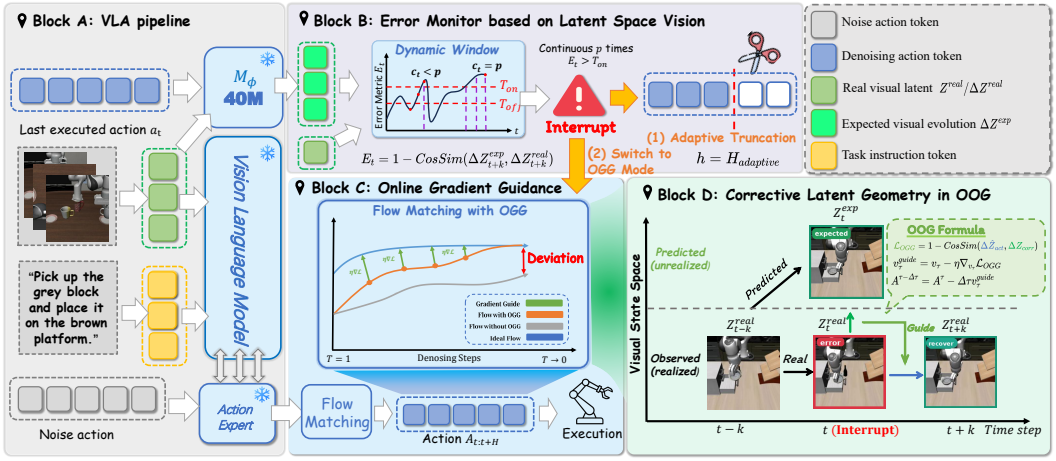


Figure 3: **Overview of VLA-Corrector.** Starting from a standard chunked VLA pipeline (**A**), we add a **Latent-space Vision Monitor** (LVM) that detects persistent execution drift and triggers an interrupt event (**B**). The event truncates stale actions and switches the next replan from normal flow matching to OGG-guided flow matching (**C**). OGG uses the expected and observed latent evolution to guide the replan back toward a recoverable trajectory (**D**).

(3) Event-Triggered Truncation explains how persistent deviations induce adaptive horizons; and **(4) OGG Correction** presents corrective replanning after truncation.

3.1 Training the External Latent Dynamics Corrector

VLA-Corrector is trained after a VLA policy has already been obtained. We first fine-tune the VLA backbone on the benchmark training set, then freeze the VLA and use its visual encoder $\mathcal{E}$ to extract visual latents from demonstration trajectories. Given a transition (o_t, a_t, o_{t+k}), we compute

$$Z_t^{\text{real}} = \mathcal{E}(o_t), \qquad Z_{t+k}^{\text{real}} = \mathcal{E}(o_{t+k}), \qquad \Delta Z_{t+k}^{*} = Z_{t+k}^{\text{real}} - Z_t^{\text{real}}, \tag{2}$$

where ΔZ_{t+k}^{*} is the target short-horizon visual latent evolution induced by the executed action. We then train a lightweight external dynamics corrector M_ϕ to predict this residual evolution from the current latent state and action:

$$\Delta \hat{Z}_{t+k} = M_\phi(Z_t^{\text{real}}, a_t). \tag{3}$$

Rather than predicting the absolute future latent state, M_ϕ instead predicts a residual visual latent evolution, which actively suppresses static scene content and encourages the model to focus on task-relevant dynamics. The training objective combines magnitude matching and directional consistency:

$$\mathcal{L}_{\text{corr}} = \left\| \Delta \hat{Z}_{t+k} - \Delta Z_{t+k}^{*} \right\|_2^2 + \beta \left[1 - \text{CosSim}\left(\Delta \hat{Z}_{t+k}, \Delta Z_{t+k}^{*} \right) \right], \tag{4}$$

where β balances residual accuracy and directional alignment. The trainable component is therefore not the VLA policy itself, but a lightweight latent dynamics module trained separately on top of frozen VLA features. This modular design allows the corrector to be retrained or replaced for each benchmark without re-optimizing the expensive VLA backbone. The corrector is trained on benchmark demonstration trajectories because they are easy to obtain and broadly reflect successful task progress. Its goal is not to model all possible futures as a full world model would do, but to learn whether local latent dynamics remain consistent with on-track execution. Demonstrations are suitable for this purpose: although they may contain small teleoperation jitter or local imperfections, they still capture behavior that keeps the task progressing correctly. Since this is a local and low-dimensional prediction task, a lightweight MLP is sufficient in practice; in our experiments, a 40M corrector already provides an effective monitoring and corrective signal, which can be trained at a very low cost.

3.2 Latent Visual Dynamics for Online Anomaly Detection

After training M_ϕ, we use it online to monitor whether the current action chunk remains reliable. During execution, **Latent-space Vision Monitor** (LVM) compares the *expected* latent visual evolution predicted by M_ϕ with the *actual* evolution observed from fresh camera input, detecting deviations before they accumulate into failure.

Expected residual dynamics. At control step t, LVM uses the executed action a_t and current latent state Z_t^{real} to predict the expected short-horizon latent residual, $\Delta Z_{t+k}^{\text{exp}} = M_\phi(Z_t^{\text{real}}, a_t)$, where k is the prediction interval.

Online inconsistency score. During rollout, the latest observation remains available even though the policy is not re-queried. We encode the future observation and compute the actual residual $\Delta Z_{t+k}^{\text{real}} = Z_{t+k}^{\text{real}} - Z_t^{\text{real}}$. LVM then measures the mismatch between expected and actual latent evolution:

$$E_t = 1 - \text{CosSim}\big(\Delta Z_{t+k}^{\text{exp}}, \Delta Z_{t+k}^{\text{real}}\big), \qquad \text{CosSim}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^\top \mathbf{v} / (\|\mathbf{u}\| \, \|\mathbf{v}\|). \tag{5}$$

A larger E_t indicates stronger visual dynamics mismatch, providing a continuous signal for later event-triggered truncation.

3.3 Event-Triggered Truncation under Robust Online Monitoring

Given the continuous inconsistency score E_t, we need to decide when a deviation is persistent enough to justify intervention. Directly thresholding E_t can be unstable, since transient visual outliers may cause false triggers. We therefore use a robust event-triggered rule based on dynamic thresholds and persistence checking. Specifically, we maintain a sliding window of recent scores, $\mathbf{E}_W = \{E_{t-w+1}, \ldots, E_t\}$, and compute its median M_e and median absolute deviation,

$$\text{MAD} = \text{median}(|E_i - M_e|), \qquad E_i \in \mathbf{E}_W. \tag{6}$$

We then define two adaptive thresholds,

$$T_{\text{on}} = M_e + \lambda_{\text{on}} \text{MAD}, \qquad T_{\text{off}} = M_e + \lambda_{\text{off}} \text{MAD}, \qquad \lambda_{\text{on}} > \lambda_{\text{off}}, \tag{7}$$

where T_{on} confirms persistent deviation and T_{off} provides hysteresis for recovery. An interrupt event is triggered only when $E_t > T_{\text{on}}$ holds for p consecutive steps; isolated spikes are ignored.

Once an interrupt is triggered, VLA-Corrector discards the remaining actions in the current queue and queries the policy again under corrective mode. If the controller has executed h actions from the original queue, the realized horizon becomes $H_{\text{adaptive}} = h < H$. Thus, the fixed action horizon is shortened only when sustained visual drift indicates that the current chunk has become stale. The full thresholding state machine is provided in Appendix B.1.

3.4 Online Gradient Guidance for Corrective Inference

Truncation stops stale actions, but recovery still depends on the next replan. After an interrupt event, VLA-Corrector applies **Online Gradient Guidance** (OGG) [34] only to the single policy call immediately following the interrupt event, guiding this recovery replan toward a corrective latent direction.

Predicted action effect. At denoising step τ of flow matching, let A^τ be the noisy action chunk. The VLA predicts a velocity field $v_\tau = \pi_\theta(A^\tau, Z_t^{\text{real}}, \tau)$, from which we estimate the clean chunk $\hat{A}_0 = A^\tau - \tau v_\tau$ and take its first action $\hat{a}_t = \hat{A}_0[0]$. The corrector then predicts the latent effect of this candidate action:

$$\Delta \hat{Z}_{\text{act}} = M_\phi(Z_t^{\text{real}}, \hat{a}_t). \tag{8}$$

Corrective target. Let $t-k$ be the last stable step before an interrupt event. We define the expected residual $\Delta Z_{\text{exp}} = \Delta Z_t^{\text{exp}} = M_\phi(Z_{t-k}^{\text{real}}, a_{t-k})$, which is predicted by M_ϕ from step $t-k$. The accumulated deviation is $\Delta Z_{\text{dev}} = \Delta Z_t^* = Z_t^{\text{real}} - Z_{t-k}^{\text{real}}$. The corrective latent direction is

$$\Delta Z_{\text{corr}} = \Delta Z_{\text{exp}} - \Delta Z_{\text{dev}}. \tag{9}$$

This direction preserves the intended local dynamics while compensating for the drift accumulated during open-loop execution.

Table 1: **Cross-architecture generalization on MetaWorld.** Success rate (%, ↑) across difficulty splits. Green arrows show absolute improvements over the corresponding baseline.

Backbone	Method	Easy↑	Medium↑	Hard↑	Very Hard↑	Avg.↑
$\pi_{0.5}$	Baseline	70.5	45.0	38.3	41.0	48.70
	+ VLA-Corrector	**83.2** ↑ 12.7	**61.7** ↑ 16.7	**47.5** ↑ 9.2	**65.0** ↑ 24.0	**64.35** ↑ 15.65
SmolVLA	Baseline	81.3	53.6	51.7	61.0	61.90
	+ VLA-Corrector	**83.4** ↑ 2.1	**56.0** ↑ 2.4	**64.2** ↑ 12.5	**63.0** ↑ 2.0	**66.65** ↑ 4.75
X-VLA	Baseline	72.5	46.4	48.3	55.0	55.55
	+ VLA-Corrector	**74.4** ↑ 1.9	**50.0** ↑ 3.6	**50.0** ↑ 1.7	**64.0** ↑ 9.0	**59.60** ↑ 4.05

Table 2: **RoboTwin v2.0 evaluation.** Success rate (%, ↑), average policy calls per episode (↓), and per-step wall-clock latency (↓). The benchmark covers all 50 tasks in clean and randomized configurations, with 50 episodes per task/configuration (5,000 episodes per method); the same frozen $\pi_{0.5}$ backbone is used throughout.

Method	Clean↑	Random↑	Overall (95% CI)↑	Calls/ep.↓	ms/step↓
Baseline $H = 50$	68.2	62.6	65.4 [64.1,66.7]	5.74	8.10
Baseline $H = 30$ (latency-matched)	72.0	66.8	69.4 [68.1,70.7]	9.41	9.98
VLA-Corrector $H = 50$	**76.0**	**70.2**	**73.1 [71.9,74.3]**	6.88	10.02

Guided velocity update. OGG aligns the predicted action effect with ΔZ_{corr} through

$$\mathcal{L}_{\text{OGG}} = 1 - \text{CosSim}(\Delta \hat{Z}_{\text{act}}, \Delta Z_{\text{corr}}), \tag{10}$$

and injects the resulting gradient into the flow-matching velocity:

$$v_\tau^{\text{guide}} = v_\tau - \eta \nabla_{v_\tau} \mathcal{L}_{\text{OGG}}, \qquad A^{\tau-\Delta\tau} = A^\tau - \Delta\tau v_\tau^{\text{guide}}. \tag{11}$$

Here, η controls the guidance strength. Since OGG modifies the velocity field rather than directly perturbing action coordinates, it remains compatible with the original flow-matching process and yields smoother corrective replanning.

4 Experiments

We evaluate VLA-Corrector on MetaWorld [52], LIBERO [28], RoboTwin v2.0 [6], and a real AgileX PiPER robot, using $\pi_{0.5}$ [16] as the main backbone and SmolVLA [38]/X-VLA [58] for cross-architecture evaluation. The experiments cover four aspects: **(1) Performance and policy-call efficiency**, evaluating success rate and success-per-call across benchmarks and horizons; **(2) Mechanism analysis**, examining LVM detection, truncation timing, and OGG recovery; **(3) Real-world transfer**, testing the method on physical robot manipulation; and **(4) Ablations**, analyzing component necessity and sensitivity. Detailed protocols, metrics, and compute resources are provided in Appendix C.1.

4.1 Main Results

Cross-Architecture Evaluation on MetaWorld. VLA-Corrector consistently improves all three VLA backbones on MetaWorld, with larger gains on harder tasks. As shown in Table 1, the average success rate improves by 15.65 points for $\pi_{0.5}$, 4.75 points for SmolVLA, and 4.05 points for X-VLA. The strongest gain appears on the Very Hard split of $\pi_{0.5}$, increasing success from 41.0% to 65.0%.

RoboTwin Evaluation. On the broader RoboTwin v2.0 benchmark, VLA-Corrector improves overall success from 65.4% to 73.1% over the original $H = 50$ baseline and from 69.4% to 73.1% over the latency-matched $H = 30$ baseline. The gain over the matched baseline is 3.7 points while per-step latency remains nearly identical (10.02 vs. 9.98 ms); VLA-Corrector also uses 27% fewer policy calls than the matched baseline (6.88 vs. 9.41 calls/episode). The improvement persists in

Table 3: **Sample efficiency on LIBERO.** Success rate (%, ↑). Green arrows compare VLA-Corrector against the few-shot fine-tuned baseline.

Model	Object↑	Spatial↑	Goal↑	Long↑	Avg.↑
$\pi_{0.5}$ (Full Fine-tuned)	99.4	98.2	97.8	92.4	96.95
$\pi_{0.5}$ (Few-shot Fine-tuned)	97.8	95.4	96.2	86.6	94.00
$\pi_{0.5}$ (Few-shot) **+ VLA-Corrector**	**99.8** ↑ 2.0	**100.0** ↑ 4.6	**98.0** ↑ 1.8	**93.4** ↑ 6.8	**97.80** ↑ 3.80

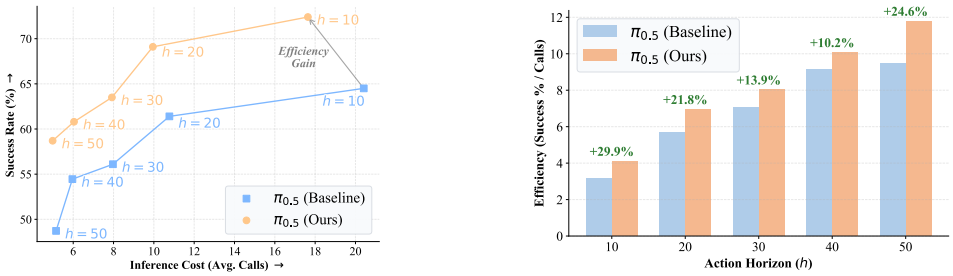


Figure 4: **Performance–efficiency analysis on $\pi_{0.5}$. Left**: performance–efficiency trade-off across action horizons. **Right**: success-per-call efficiency. VLA-Corrector improves success rate across action horizons and yields consistent efficiency gains.

both clean and randomized configurations. A more comprehensive and detailed analysis can be found in Appendix D.6.

Sample Efficiency on LIBERO. We further evaluate whether VLA-Corrector can compensate for limited task-specific data. Table 3 compares a fully fine-tuned $\pi_{0.5}$ model with a few-shot fine-tuned counterpart on LIBERO. Starting from the publicly released LeRobot [3] few-shot checkpoint `pi05_libero_base`, VLA-Corrector improves the average success rate from 94.00% to 97.80%, even surpassing the fully fine-tuned baseline (96.95%).

These results suggest that few-shot fine-tuning can already learn much of the normal task trajectories, but may lack coverage of drifted states and their recovery behaviors. Instead of improving recovery by exposing the backbone to more rare failure cases during training, VLA-Corrector makes recovery easier at inference time by interrupting error accumulation early and guiding corrective replanning, thereby alleviating the dependence on additional post-training data for robust recovery.

Performance–Efficiency Trade-off. VLA-Corrector improves success-per-call efficiency across action horizons rather than simply querying the policy more often. As shown in Figure 4, the largest success-per-call gains reach 29.9% for $\pi_{0.5}$, 45.3% for SmolVLA, and 39.1% for X-VLA. For example, on SmolVLA with horizon 10, success improves from 61.90% to 73.00% while policy calls decrease from 19.27 to 15.64. This suggests that VLA-Corrector makes each policy query more useful by interrupting stale chunks and guiding recovery. Full results are provided in Appendix D.3.

4.2 Mechanism Analysis

In this section, we will analyze the mechanisms behind VLA-Corrector from three perspectives: **1) LVM detection**, whether LVM provides a useful and well-timed drift signal; and **2) OGG correction**, whether OGG improves recovery after an interrupt event; **3) controlled recovery**, whether the full framework improves recovery in a controlled failure case.

LVM Detection. LVM provides an informative drift signal. As shown in Figure 5, successful episodes concentrate at low E_t, while failed episodes show a heavier high-score tail and trigger more interrupt events. This indicates that E_t captures failure-prone visual dynamics mismatch.

LVM also intervenes at meaningful moments. We manually divide MetaWorld trajectories into *critical* phases, such as precise grasping and alignment, and *non-critical* phases, such as tolerant transport after a stable grasp. Figure 6 shows that 83.7% of truncations occur in critical phases. This result verifies the core adaptive-horizon intuition of VLA-Corrector: it does not simply shorten the

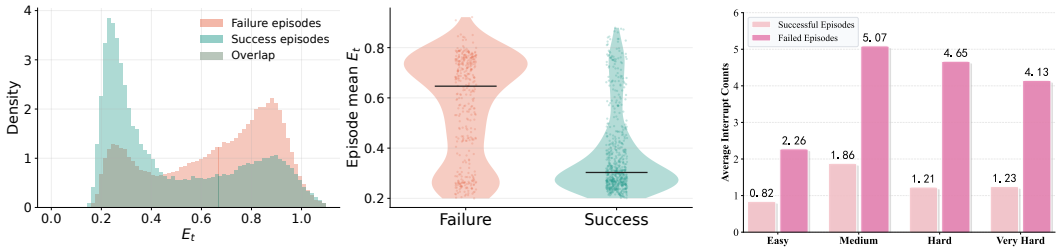
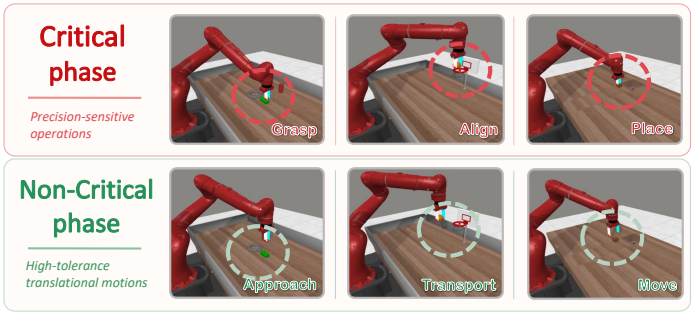


Figure 5: **LVM detection analysis. Left**: distribution of the inconsistency score E_t, where successful episodes concentrate at low values and failed episodes show a heavier high-score tail. **Right**: interrupt frequency, where failed episodes trigger more interrupt events than successful ones.



Truncation Distribution by Task Phase

Phase	Rate
Critical	**83.7%**
Non-critical	16.3%

5.1× more truncations occur in critical phases than in non-critical phases.

Figure 6: **Task-phase analysis of LVM-triggered truncation.** We manually divide MetaWorld trajectories into *critical* and *non-critical* phases. **Left**: representative trajectories show that truncation is triggered much more frequently during critical phases. **Right**: 83.7% of truncations occur in critical phases, while only 16.3% occur in non-critical phases.

horizon everywhere, but instead preserves long-horizon efficiency in tolerant phases and restores short-horizon precision when the task enters error-sensitive phases.

OGG Correction. We isolate OGG by comparing standard re-inference and OGG-guided re-inference
after the same interrupt-triggered truncation. Recovery is counted when $E_t < T_{\text{off}}$ within the next 10 steps. As shown in Fig. 8, OGG improves recovery across all difficulty levels, with an average gain of 0.23. This indicates that, after truncation stops stale actions, OGG-guided re-inference further improves the quality of the recovery replan compared to standard re-inference.

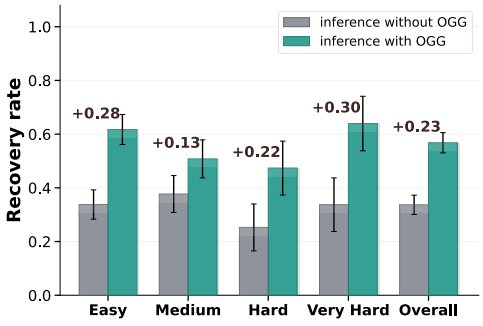


Figure 8: **Post-interrupt recovery.** OGG-guided inference consistently outperforms standard inference.

Controlled Recovery Case. Figure 7 shows a representative LIBERO episode where both methods start from the same initial state and encounter the same grasping error when approaching the cup handle. In the top row, the baseline rollout is only monitored by LVM for error alignment without truncating the remaining actions; after the deviation is detected, it continues executing the original action chunk, leading to an unstable grasp, cup drop, and task failure. In the bottom row, VLA-Corrector detects the same deviation but actively truncates the stale actions and triggers corrective replanning, allowing the robot to recover a stable grasp and place the cup at the target location. This controlled comparison highlights the overall effectiveness of VLA-Corrector under the same initial condition and the same detected execution error.

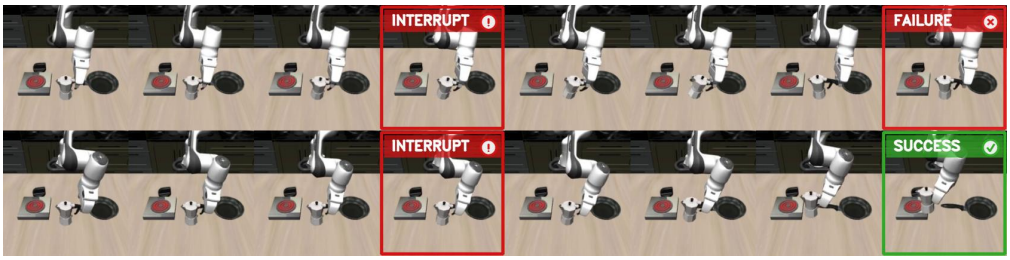


Figure 7: **Controlled recovery case.** Given the same initial state and detected grasping error, the monitored baseline continues the original chunk and fails (**top**), while VLA-Corrector truncates stale actions, replans with OGG, and completes the task (**bottom**).

4.3 Real-World Evaluation

Setup. We evaluate VLA-Corrector on an AgileX PiPER 6-DoF arm using $\pi_{0.5}$ as the backbone. The benchmark includes three task groups, each with three tasks and 20 trials: **1) Pick-and-place** tests standard manipulation, **2) Alignment** tests precision-sensitive placement or insertion, and **3) Disturbance recovery** tests recovery when objects or targets are manually shifted during execution. Full task definitions, protocols, failure analysis, and representative demonstrations are provided in Appendix E.

Table 4: **Real-world evaluation on AgileX PiPER.** Success rate (%, ↑) over three task groups. Each group contains three tasks with 20 trials per task. We report 95% binomial confidence intervals.

Method	Pick-place↑	Alignment↑	Disturbance↑	Avg.↑
$\pi_{0.5}$ Baseline	70.0 ± 11.6	56.7 ± 12.5	40.0 ± 12.4	55.6 ± 7.3
+ VLA-Corrector	**78.3** $\pm$ **10.4** ↑ 8.3	**73.3** $\pm$ **11.2** ↑ 16.6	**68.3** $\pm$ **11.8** ↑ 28.3	**73.3** $\pm$ **6.5** ↑ 17.7

Results. Table 4 shows that VLA-Corrector improves all three task groups, increasing average success from 55.6% to 73.3%. The gain is modest on pick-and-place (+8.3), larger on alignment (+16.6), and largest on disturbance recovery (+28.3). This trend matches the intended role of VLA-Corrector: it preserves standard execution performance, improves precision-sensitive manipulation, and is most beneficial when online disturbances make the remaining action chunk outdated.

4.4 Ablation Studies

Table 5: **Component ablation on MetaWorld.** Success rate (%, ↑). Green arrows indicate average improvement over the baseline.

Variant	Easy↑	Medium↑	Hard↑	Very Hard↑	Avg.↑
Baseline (Open-loop)	70.5	45.0	38.3	41.0	48.70
+ Truncation Only	81.8	53.6	**50.0**	56.0	60.35 ↑ 11.65
+ Truncation + OGG	**83.2**	**61.7**	47.5	**65.0**	**64.35** ↑ 15.65

We ablate VLA-Corrector on $\pi_{0.5}$ in MetaWorld, focusing on **(1) components**: necessity of truncation and OGG; **(2) detector design**: decoupling monitoring from backbone fine-tuning; and **(3) sensitivity**, with detailed results in Appendix D.1.

Component effects. Table 5 shows that truncation alone raises average success from 48.70% to 60.35%, confirming the importance of stopping stale actions. Adding OGG further improves the average to 64.35%, showing that post-truncation recovery also benefits from guided re-inference.

Detector design and sensitivity. We compare our decoupled LVM with an internal detection head trained during $\pi_{0.5}$ fine-tuning using the same auxiliary latent-residual prediction target. The internal head takes the final-layer hidden state of the last token, combines it with the executed action, and predicts the same task as the external LVM. As shown in Appendix D.1, decoupled LVM improves average success from 49.55% to 64.35%, suggesting that separating monitoring from action generation preserves the backbone representation and yields a cleaner drift signal. The same appendix also shows that moderate guidance strength and a 40M LVM are sufficient; overly strong guidance or larger monitors provide little additional benefit.

5 Conclusion

This paper studies the open-loop blind spot in action-chunked VLA policies, where fixed horizons improve policy-call efficiency but allow stale actions to accumulate errors. VLA-Corrector addresses this issue with a lightweight detect-and-correct layer that monitors latent visual dynamics, truncates stale actions when drift persists, and guides the next inference toward recovery. Our results show that small inference-time modules can provide targeted robustness gains without retraining the VLA backbone. Rather than replacing action chunking, VLA-Corrector makes it adaptive: long-horizon execution is preserved when reliable, while corrective replanning is invoked when the current chunk should no longer be trusted.

Acknowledgments and Disclosure of Funding

This work was supported by Ningbo Global Innovation Center, Zhejiang University (Grant Nos. NBRY2025X0015 and 702J26007) and Ningbo Yongjiang Talent Programme (No. 2025A-386-G).

References

[1] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[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] Remi Cadene, Simon Aliberts, Francesco Capuano, Michel Aractingi, Adil Zouitine, Pepijn Kooijmans, Jade Choghari, Martino Russi, Caroline Pascal, Steven Palma, et al. Lerobot: An open-source library for end-to-end robot learning. *arXiv preprint arXiv:2602.22818*, 2026.

[4] Jun Cen, Siteng Huang, Yuqian Yuan, Kehan Li, Hangjie Yuan, Chaohui Yu, Yuming Jiang, Jiayan Guo, Xin Li, Hao Luo, et al. Rynnvla-002: A unified vision-language-action and world model. *arXiv preprint arXiv:2511.17502*, 2025.

[5] Hao Chen, Jiaming Liu, Chenyang Gu, Zhuoyang Liu, Renrui Zhang, Xiaoqi Li, Xiao He, Yandong Guo, Chi-Wing Fu, Shanghang Zhang, et al. Fast-in-slow: A dual-system foundation model unifying fast manipulation within slow reasoning. *arXiv preprint arXiv:2506.01953*, 2025.

[6] Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

[7] 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.

[8] Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *arXiv preprint arXiv:2505.23705*, 2025.

[9] Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. *arXiv preprint arXiv:2508.15260*, 2025.

[10] George Jiayuan Gao, Tianyu Li, and Nadia Figueroa. Out-of-distribution recovery with object-centric keypoint inverse policy for visuomotor imitation learning. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9816–9828. IEEE, 2025.

[11] Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, et al. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems (RSS)*, 2024.

[12] Ruiyu Gou. *Learning temporal action chunking for motor control*. PhD thesis, University of British Columbia, 2024.

[13] 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.

[14] Zheyuan Hu, Robyn Wu, Naveen Enock, Jasmine Li, Riya Kadakia, Zackory Erickson, and Aviral Kumar. Rac: Robot learning for long-horizon tasks by scaling recovery and correction. *arXiv preprint arXiv:2509.07953*, 2025.

[15] Nils Ingelhag, Jesper Munkeby, Michael C Welle, Marco Moletta, and Danica Kragic. Real-time operator takeover for visuomotor diffusion policy training. *arXiv preprint arXiv:2502.02308*, 2025.

[16] 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.

[17] Arnav Kumar Jain, Vibhakar Mohta, Subin Kim, Atiksh Bhardwaj, Juntao Ren, Yunhai Feng, Sanjiban Choudhury, and Gokul Swamy. A smooth sea never made a skilled sailor: Robust imitation via learning to search. *arXiv preprint arXiv:2506.05294*, 2025.

[18] Sunshine Jiang, Xiaolin Fang, Nicholas Roy, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Siddharth Ancha. Streaming flow policy: Simplifying diffusion/flow-matching policies by treating action trajectories as flow trajectories. *arXiv preprint arXiv:2505.21851*, 2025.

[19] Yuming Jiang, Siteng Huang, Shengke Xue, Yaxi Zhao, Jun Cen, Sicong Leng, Kehan Li, Jiayan Guo, Kexiang Wang, Mingxiu Chen, et al. Rynnvla-001: Using human demonstrations to improve robot manipulation. *arXiv preprint arXiv:2509.15212*, 2025.

[20] Dong Jing, Gang Wang, Jiaqi Liu, Weiliang Tang, Zelong Sun, Yunchao Yao, Zhenyu Wei, Yunhui Liu, Zhiwu Lu, and Mingyu Ding. Mixture of horizons in action chunking. *arXiv preprint arXiv:2511.19433*, 2025.

[21] Sarosh Khan and Ellie Tanimura. Test-time stochasticity estimation for adaptive action chunk selection, 2025.

[22] 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.

[23] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[24] Wei Li, Renshan Zhang, Rui Shao, Jie He, and Liqiang Nie. Cogvla: Cognition-aligned vision-language-action model via instruction-driven routing & sparsification. *arXiv preprint arXiv:2508.21046*, 2025.

[25] Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, et al. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023.

[26] Yuanchang Liang, Xiaobo Wang, Kai Wang, Shuo Wang, Xiaojiang Peng, Haoyu Chen, David Kim Huat Chua, and Prahlad Vadakkepat. Adaptive action chunking at inference-time for vision-language-action models. *arXiv preprint arXiv:2604.04161*, 2026.

[27] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[28] 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.

[29] Isabella Liu, An-Chieh Cheng, Rui Yan, Geng Chen, Ri-Zhao Qiu, Xueyan Zou, Sha Yi, Hongxu Yin, Xiaolong Wang, and Sifei Liu. Long-horizon manipulation via trace-conditioned vla planning. *arXiv preprint arXiv:2604.21924*, 2026.

[30] Yuejiang Liu, Jubayer Ibn Hamid, Annie Xie, Yoonho Lee, Maximilian Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via guided test-time sampling. *arXiv preprint arXiv:2408.17355*, 2024.

[31] Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision-language-action models for embodied ai. *arXiv preprint arXiv:2405.14093*, 2024.

[32] Cyrus Neary, Omar G Younis, Artur Kuramshin, Ozgur Aslan, and Glen Berseth. Improving pre-trained vision-language-action policies with model-based search. *arXiv preprint arXiv:2508.12211*, 2025.

[33] 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.

[34] Minho Park, Kinam Kim, Junha Hyung, Hyojin Jang, Hoiyeong Jin, Jooyeol Yun, Hojoon Lee, and Jaegul Choo. Acg: Action coherence guidance for flow-based vla models. *arXiv preprint arXiv:2510.22201*, 2025.

[35] 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.

[36] Kohei Sendai, Maxime Alvarez, Tatsuya Matsushima, Yutaka Matsuo, and Yusuke Iwasawa. Leave no observation behind: Real-time correction for vla action chunks. *arXiv preprint arXiv:2509.23224*, 2025.

[37] Archit Sharma, Ahmed M Ahmed, Rehaan Ahmad, and Chelsea Finn. Self-improving robots: End-to-end autonomous visuomotor reinforcement learning. *arXiv preprint arXiv:2303.01488*, 2023.

[38] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[39] Junhyuk So, Chiwoong Lee, Shinyoung Lee, Jungseul Ok, and Eunhyeok Park. Improving generative behavior cloning via self-guidance and adaptive chunking. *arXiv preprint arXiv:2510.12392*, 2025.

[40] Haoming Song, Delin Qu, Yuanqi Yao, Qizhi Chen, Qi Lv, Yiwen Tang, Modi Shi, Guanghui Ren, Maoqing Yao, Bin Zhao, et al. Hume: Introducing system-2 thinking in visual-language-action model. *arXiv preprint arXiv:2505.21432*, 2025.

[41] Wenxuan Song, Jiayi Chen, Pengxiang Ding, Han Zhao, Wei Zhao, Zhide Zhong, Zongyuan Ge, Jun Ma, and Haoang Li. Accelerating vision-language-action model integrated with action chunking via parallel decoding. *arXiv preprint arXiv:2503.02310*, 2025.

[42] Zhanyi Sun and Shuran Song. Latent policy barrier: Learning robust visuomotor policies by staying in-distribution. *arXiv preprint arXiv:2508.05941*, 2025.

[43] Libo Wang. Sigma: The key for vision-language-action models toward telepathic alignment. *arXiv preprint arXiv:2512.00783*, 2025.

[44] Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, et al. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. *Proceedings of the AAAI conference on artificial intelligence*, 40(22):18638–18646, 2026.

[45] Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, et al. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024.

[46] Zihua Wang, Zhitao Lin, Ruibo Li, Yu Zhang, Xu Yang, Siya Mi, and Xiu-Shen Wei. Open-loop planning, closed-loop verification: Speculative verification for vla. *arXiv preprint arXiv:2604.02965*, 2026.

[47] Junjie Wen, Yichen Zhu, Minjie Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Xiaoyu Liu, Chaomin Shen, Yaxin Peng, and Feifei Feng. Diffusionvla: Scaling robot foundation models via unified diffusion and autoregression. In *Forty-second International Conference on Machine Learning*, 2025.

[48] Pengyuan Wu, Pingrui Zhang, Zhigang Wang, Dong Wang, Bin Zhao, and Xuelong Li. Closed-loop action chunks with dynamic corrections for training-free diffusion policy. *arXiv preprint arXiv:2603.01953*, 2026.

[49] Zhichao Wu, Junyin Ye, Zhilong Zhang, Yihao Sun, Haoxin Lin, Jiaheng Luo, Haoxiang Ren, Lei Yuan, and Yang Yu. Speedup patch: Learning a plug-and-play policy to accelerate embodied manipulation. *arXiv preprint arXiv:2603.20658*, 2026.

[50] Changhua Xu, Jie Lu, Junyu Xuan, and En Yu. Vgas: Value-guided action-chunk selection for few-shot vision-language-action adaptation. *arXiv preprint arXiv:2602.07399*, 2026.

[51] Shaoqing Xu, Fang Li, Zhixiang Duan, Yifan Yang, Tianshi Xie, and Zhi-Xin Yang. Vla-in-the-loop: Online policy correction with world models for robust robotic grasping. OpenReview manuscript, n.d. URL `https://openreview.net/forum?id=aT4LG8c6DE`. ICLR 2026 submission.

[52] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[53] Thomas TCK Zhang, Daniel Pfrommer, Chaoyi Pan, Nikolai Matni, and Max Simchowitz. Action chunking and data augmentation yield exponential improvements in behavior cloning for continuous spaces. In *The Fourteenth International Conference on Learning Representations*, 2026.

[54] Chen Zhao, Zhuoran Wang, Haoyang Li, Shifeng Bao, Guanlin Li, Youhe Feng, Yang Li, Jie Tang, and Jing Zhang. Action draft and verify: A self-verifying framework for vision-language-action model. *arXiv preprint arXiv:2603.18091*, 2026.

[55] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1702–1713, 2025.

[56] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.

[57] 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.

[58] Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. *arXiv preprint arXiv:2510.10274*, 2025.

[59] 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 Related Work

A.1 Generative VLA Models

Embodied intelligence is shifting from task-specific policies toward unified Vision-Language-Action (VLA) foundation models. Representative works have shown that combining large-scale vision-language pretraining with robot control can substantially improve cross-task generalization, semantic understanding, and cross-platform transfer [33, 2, 59, 22, 11, 25, 1, 35, 41]. On the action modeling side, modern VLA and general robot policies often adopt generative frameworks to model high-dimensional, continuous, and multi-modal action distributions: diffusion models generate actions through iterative denoising [7, 47, 2, 5, 24, 55, 13, 23, 40], while flow matching directly learns velocity fields along continuous trajectories. These approaches typically offer stronger action expressivity, but they also incur higher policy-call cost[45]. Consequently, reducing control latency without sacrificing generation quality has become a central challenge in deploying VLA models.

A.2 Action Chunk and Horizon Trade-off

To alleviate the tension between high-frequency control and expensive policy inference, **action chunk** has been widely adopted in continuous control. The core idea is to predict multiple future actions in a single inference call and then execute them sequentially with a downstream controller [7, 56, 2, 38, 1, 43, 8]. This design effectively amortizes policy-call frequency and improves temporal smoothness, but it also reduces the use of fresh observations within a fixed action horizon. A longer horizon further lowers inference frequency and maintains action consistency, but weakens responsiveness to environmental changes; a shorter horizon more closely resembles strict closed-loop control and often provides better local recovery, yet requires much more frequent policy calls [20, 53, 30, 21, 39, 9, 54, 29]. Most existing approaches therefore seek a compromise under a static horizon, whereas our work focuses on an event-triggered dynamic horizon: the system retains the efficiency of long chunks during stable phases and truncates them on demand to recover closed-loop responsiveness during local failures.

A.3 Failure Recovery in Visuomotor Policies

Long-horizon visuomotor control commonly suffers from compounding errors and covariate shift: once execution drifts away from the training distribution, the policy often cannot recover on its own. Existing approaches can be broadly divided into two categories. One line of work expands the training distribution through additional interaction, human intervention, or recovery data collection, and then updates the policy accordingly [37, 15, 14, 32]. Another line explicitly introduces recovery policies, search mechanisms, or distributional constraints to pull the system back to a safe region once it departs from the expert manifold [10, 42, 17, 26, 12, 49, 51]. These works collectively show that explicit recovery mechanisms can substantially improve the robustness of visuomotor policies; however, they typically rely on additional recovery data, separately trained recovery modules, or joint optimization tailored to a specific policy backbone.

B Method Details

B.1 Details of Event-Triggered Truncation

This section provides the full event-triggered truncation rule used by LVM. Given the online inconsistency score E_t, the monitor maintains a sliding window of recent scores,

$$\mathbf{E}_W = \{E_{t-w+1}, \dots, E_t\}.$$

Let

$$M_e = \mathrm{median}(\mathbf{E}_W), \qquad \mathrm{MAD} = \mathrm{median}(|E_i - M_e|), \quad E_i \in \mathbf{E}_W.$$

We use the median absolute deviation because it is less sensitive to transient score spikes than the mean and variance. Based on this robust statistic, we define asymmetric on/off thresholds,

$$T_{\mathrm{on}} = M_e + \lambda_{\mathrm{on}}\mathrm{MAD}, \qquad T_{\mathrm{off}} = M_e + \lambda_{\mathrm{off}}\mathrm{MAD}, \qquad \lambda_{\mathrm{on}} > \lambda_{\mathrm{off}}.$$

The higher threshold T_{on} is used to confirm abnormal deviation, while the lower threshold T_{off} provides hysteresis and prevents rapid oscillation between normal and abnormal states.

To avoid triggering on isolated spikes, we maintain a persistence counter:

$$c_t = \begin{cases} c_{t-1} + 1, & \text{if } E_t > T_{\text{on}}, \\ 0, & \text{if } E_t < T_{\text{off}}, \\ c_{t-1}, & \text{otherwise.} \end{cases} \tag{12}$$

An interrupt event is triggered when

$$c_t \geq p, \tag{13}$$

where p is the patience parameter. After an interrupt event, the remaining actions in the current execution queue are discarded, the counter is reset, and the next policy query is performed under corrective mode. If h actions have already been executed from the current queue, the realized action horizon is

$$H_{\text{adaptive}} = h < H.$$

In this way, the system preserves long-horizon execution during stable phases, while shortening the horizon when persistent visual drift indicates that the current action chunk is no longer reliable.

B.2 LVM and OGG Runtime Parameters

LVM monitoring parameters. The Latent-space Vision Monitor uses a robust online thresholding state machine. We maintain a sliding window of recent inconsistency scores with window size 15. The dynamic thresholds are computed using $\lambda_{\text{on}} = 3.0$ and $\lambda_{\text{off}} = 2.0$. An interrupt event is triggered only when the inconsistency score exceeds the activation threshold for $p = 5$ consecutive steps. Similarly, the monitor requires 5 consecutive safe steps for reset. To avoid repeated triggers immediately after intervention, we use an interrupt cooldown of 10 steps.

OGG parameters. Online Gradient Guidance is invoked only for the single policy query immediately following an interrupt-triggered truncation. When an interrupt event occurs, the remaining action queue is cleared, the next replan is marked as a guidance replan, and OGG is applied during that next action-chunk generation. Subsequent policy calls return to standard inference unless another interrupt event is detected. The guidance strength is controlled by η. In our experiments, we use $\eta = 1$ as the default setting for $\pi_{0.5}$, SmolVLA, and X-VLA. The sensitivity of performance to η is analyzed in Table 7.

C Training and Implementation Details

C.1 Benchmarks and Evaluation Protocol

We evaluate VLA-Corrector on three simulation benchmarks: MetaWorld [52], LIBERO [28], and RoboTwin v2.0. MetaWorld tests contact-rich manipulation robustness across difficulty splits, LIBERO evaluates language-conditioned long-horizon task execution, and RoboTwin broadens the evaluation to 50 tasks under clean and randomized scene configurations. We use $\pi_{0.5}$ [16] as the main backbone, and further test SmolVLA [38] and X-VLA [58] for cross-architecture evaluation. We report task success rate as the primary metric, together with policy calls, success-per-call efficiency, and post-interrupt recovery rate when applicable.

For MetaWorld, tasks are grouped into four difficulty splits: *Easy*, *Medium*, *Hard*, and *Very Hard*. All online evaluations are conducted task by task within each split, and we report the average success rate over tasks in the corresponding split. Unless otherwise specified, each task is evaluated with 20 episodes. For the corrector data-scaling study, we use a fixed episode-level split before subsampling the training data: 10% of episodes are held out for validation, 10% are held out for testing, and the remaining episodes form the training pool. We then subsample this training pool according to the training ratio r. In the multi-seed data-scaling analysis, we use split seeds $\{42, 123, 999\}$ and require at least 20 episodes in the train, validation, and test partitions. All simulation experiments are conducted on 8 NVIDIA A100-SXM4-40GB GPUs.

For RoboTwin v2.0, we evaluate all 50 tasks in both the clean and randomized configurations, using 50 episodes per task and configuration. Initial conditions and evaluation seeds are paired across methods, yielding 5,000 episodes per method. We compare the original fixed-horizon baseline ($H = 50$), a latency-matched baseline ($H = 30$), and VLA-Corrector ($H = 50$), all with the

Table 6: **Coupled vs. decoupled detection.** Success rate (%, ↑). Both variants use OGG; only the detector design differs.

Detector	Easy↑	Medium↑	Hard↑	Very Hard↑	Avg.↑
Internal Head + OGG	65.2	46.0	36.5	50.5	49.55
Decoupled LVM + OGG	**83.2** ↑ 18.0	**61.7** ↑ 15.7	**47.5** ↑ 11.0	**65.0** ↑ 14.5	**64.35** ↑ 14.80

same frozen $\pi_{0.5}$ backbone. We report clean, randomized, and pooled success rates with 95% Wilson confidence intervals, together with average policy calls per episode and measured wall-clock latency per environment step.

C.2 External Corrector Architecture

The external corrector M_ϕ is implemented as a residual MLP for short-horizon latent dynamics prediction. We first map the executed action through a linear embedding layer, concatenate the resulting action feature with the current visual latent Z_t^{real}, and feed the combined representation into a stack of residual MLP blocks. The network predicts the short-horizon latent residual rather than the absolute future latent state. In all experiments, we use a four-layer hidden configuration with width $[2048, 2048, 2048, 2048]$. The exact number of parameters varies slightly across benchmarks and VLA backbones because both the action dimensionality and the visual latent dimensionality differ across settings. Overall, the correctors used in our experiments contain approximately 38–42M parameters, and we refer to this external module as a lightweight ∼40M MLP corrector.

C.3 Corrector Training

The corrector is trained after the VLA backbone has been fine-tuned on the benchmark training set. We freeze the VLA backbone and use its visual encoder to extract latent representations from demonstration trajectories. The corrector is then trained on these frozen latents to predict the short-horizon visual latent residual induced by the executed action. We use AdamW as the optimizer with learning rate 3×10^{-4} and weight decay 10^{-4}. The learning rate is scheduled by cosine annealing with $\eta_{\min} = 0.01 \times \text{lr}$. We train for 30 epochs with early stopping patience 5. For the ratio-sweep training runs, we use batch size 128 and evaluation batch size 256. For the deployed h1-k10 correctors used in evaluation, we use batch size 512, train for 30 epochs, and optimize a cosine-based training loss. The number of optimization steps depends on the number of training samples under each data ratio:

$$\text{steps per epoch} = \left\lceil \frac{N_{\text{train}}}{B} \right\rceil, \qquad \text{total steps} = 30 \times \text{steps per epoch},$$

where N_{train} is the number of training samples and B is the batch size.

C.4 Compute Resources

All simulation experiments are conducted on NVIDIA A100-SXM4-40GB GPUs. The main training and evaluation jobs are run on a server with 8 such GPUs. Corrector training is lightweight compared with VLA backbone fine-tuning because it only optimizes the external MLP on frozen VLA features. During online evaluation, LVM monitoring adds only a forward pass through the external corrector, while OGG introduces additional gradient computation only for the single recovery query following an interrupt event.

D Additional Experimental Results

D.1 Ablation Sensitivity

We provide additional ablations on detector design, OGG guidance strength, and LVM capacity. All experiments are conducted with $\pi_{0.5}$ on MetaWorld.

For the internal detector, we fine-tune $\pi_{0.5}$ with an additional auxiliary loss that matches the prediction task of LVM: the final-layer hidden state of the last token is combined with the executed action

Table 7: **Ablation on OGG guidance strength η.** Success rate (%, ↑) on MetaWorld. The highlighted row is the default setting.

Guidance Strength	Easy↑	Medium↑	Hard↑	Very Hard↑	Avg.↑
$\eta = 0.1$	82.7	56.8	45.2	**66.0**	62.68
$\eta = 1$ **(default)**	**83.2** ↑ 0.5	**61.7** ↑ 4.9	**47.5** ↑ 2.3	65.0 ↓ 1.0	**64.35** ↑ 1.67
$\eta = 10$	82.5	56.8	38.3	65.0	60.65 ↓ 3.70
$\eta = 100$	80.9	55.0	36.7	63.0	58.90 ↓ 5.45

Table 8: **Ablation on LVM capacity.** Success rate (%, ↑) on MetaWorld. The highlighted row is the default setting.

Monitor Capacity	Easy↑	Medium↑	Hard↑	Very Hard↑	Avg.↑
LVM-10M	78.5	52.4	39.8	55.6	56.58
LVM-40M (default)	**83.2** ↑ 4.7	61.7 ↑ 9.3	47.5 ↑ 7.7	**65.0** ↑ 9.4	**64.35** ↑ 7.77
LVM-160M	82.8 ↓ 0.4	**62.1** ↑ 0.4	**48.0** ↑ 0.5	64.2 ↓ 0.8	64.28 ↓ 0.07

and used to predict the same short-horizon latent residual. Table 6 shows that this coupled design performs much worse than the decoupled external LVM. This suggests that joint auxiliary detection training can interfere with the action-generation representation, whereas the external LVM preserves the backbone policy and provides a cleaner monitoring signal.

Tables 7 and 8 show that VLA-Corrector does not require overly strong guidance or excessively large monitors. $\eta = 1$ achieves the best average success, while larger guidance weakens performance on harder tasks. Increasing LVM capacity from 10M to 40M substantially improves success, but 160M provides almost no additional average gain.

D.2 Cross-Domain Corrector Generalization

To examine whether the external corrector learns a transferable latent-dynamics signal rather than only memorizing benchmark-specific trajectories, we evaluate cross-domain corrector transfer on MetaWorld. We use the same $\pi_{0.5}$ MetaWorld baseline policy and compare two VLA-Corrector variants: one using a corrector trained on LIBERO demonstrations, and the other using a corrector trained on MetaWorld demonstrations. Both are evaluated on MetaWorld under the same protocol.

Table 9 shows that the LIBERO-trained corrector still improves the MetaWorld baseline from 48.7% to 51.8%, suggesting that the learned latent-dynamics consistency signal has limited but non-trivial cross-domain transfer. However, the MetaWorld-trained corrector achieves a much larger gain, improving the average success rate to 58.7%. This indicates that while the corrector can capture partially transferable notions of on-track visual dynamics, domain-matched demonstrations remain important for accurate deviation detection and effective corrective guidance.

D.3 Full Performance–Efficiency Results

Table 10 provides the full performance–efficiency results across all evaluated VLA backbones and action horizons. These results complement the main-text analysis by reporting the exact success rates, average policy calls, and success-per-call efficiency gains for each setting. Overall, VLA-Corrector consistently improves task success while maintaining or reducing the number of policy calls in most cases, leading to positive success-per-call efficiency gains across all backbones. The gains are especially pronounced at longer horizons, where the open-loop blind spot is larger and stale actions are more likely to accumulate errors before the next policy query.

D.4 Data Efficiency of Corrector Training

Analysis. To quantify how much data is needed to train the external corrector, we subsample the MetaWorld demonstration set with training ratios $r \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$ and evaluate the resulting VLA-Corrector on the same task suite. As shown in Table 11, the corrector is already effective

Table 9: **Cross-domain corrector generalization on MetaWorld.** Success rate (%, ↑) of the same $\pi_{0.5}$ MetaWorld baseline equipped with correctors trained from different demonstration domains. Green arrows indicate absolute improvement over the baseline.

Method	Easy↑	Medium↑	Hard↑	Very Hard↑	Avg.↑
$\pi_{0.5}$ Baseline	70.5	45.0	38.3	41.0	48.7
+ LIBERO-trained Corrector	70.9 ↑ 0.4	50.5 ↑ 5.5	40.7 ↑ 2.4	45.0 ↑ 4.0	51.8 ↑ 3.1
+ MetaWorld-trained Corrector	**74.1** ↑ 3.6	**53.8** ↑ 8.8	**52.9** ↑ 14.6	**54.0** ↑ 13.0	**58.7** ↑ 10.0

Table 10: **Performance–efficiency trade-off across VLA backbones and action horizons.** Success rate, average policy calls per episode, and relative efficiency gains are reported. The arrows (↑/↓) in the header explicitly indicate whether a higher or lower metric is optimal.

Model	Horizon	Baseline		Ours (LVM+OGG)		Improvements		
		Succ (%)↑	Calls ↓	Succ (%)↑	Calls ↓	ΔSucc (%)↑	ΔCalls ↓	Eff. Gain (%)↑
$\pi_{0.5}$	10	64.50	20.41	**72.40**	**17.64**	↑ 7.90	↓ 2.77	**+29.9%**
	20	61.40	10.77	**69.10**	**9.95**	↑ 7.70	↓ 0.82	**+21.8%**
	30	56.10	7.97	**63.50**	**7.92**	↑ 7.40	↓ 0.05	**+13.9%**
	40	54.45	5.96	**60.80**	6.04	↑ 6.35	↑ 0.08	**+10.2%**
	50	48.72	5.15	**58.70**	**4.98**	↑ 9.98	↓ 0.17	**+24.6%**
SmolVLA	10	61.90	19.27	**73.00**	**15.64**	↑ 11.10	↓ 3.63	**+45.3%**
	20	58.90	11.32	**68.90**	**9.50**	↑ 10.00	↓ 1.82	**+39.4%**
	30	58.10	7.90	**65.20**	**7.60**	↑ 7.10	↓ 0.30	**+16.6%**
	40	56.80	5.64	**67.20**	**5.13**	↑ 10.40	↓ 0.51	**+30.0%**
	50	54.20	4.86	**62.90**	**4.68**	↑ 8.70	↓ 0.18	**+20.6%**
X-VLA	4	68.50	46.58	**72.00**	**35.20**	↑ 3.50	↓ 11.38	**+39.1%**
	8	67.80	23.18	**72.10**	**19.48**	↑ 4.30	↓ 3.70	**+26.5%**
	16	55.50	13.71	**60.90**	13.92	↑ 5.40	↑ 0.21	**+8.1%**
	24	52.90	10.13	**59.20**	**9.92**	↑ 6.30	↓ 0.21	**+14.3%**
	32	44.00	8.61	**54.40**	**8.34**	↑ 10.40	↓ 0.27	**+27.6%**

ΔSucc: absolute success-rate improvement. **ΔCalls**: absolute change in policy calls per episode. **Eff. Gain**: relative improvement in success-per-call over the baseline.

with limited data: using only 20% of the demonstrations yields a 48.32% macro-average success rate. Performance improves as more data is added and reaches 52.20% at $r = 0.6$, after which the gain becomes marginal. This suggests that the corrector does not require exhaustive coverage of all task variations; once it has seen enough examples of on-track local dynamics, it can provide a useful signal for detecting deviations and guiding recovery.

Table 11: **Data efficiency of corrector training on MetaWorld.** We compare the original open-loop baseline at NAS=50 with VLA-Corrector trained using different fractions of demonstration trajectories. The results show that moderate corrector data already surpasses the baseline and that performance largely saturates around 60% of the training set. Bold indicates the best result in each column.

Method / Training Ratio	Easy	Medium	Hard	Very Hard	Macro Avg.
Baseline (NAS=50)	70.54	45.00	38.33	41.00	48.72
VLA-Corrector ($r = 0.2$)	71.07	49.55	36.67	36.00	48.32
VLA-Corrector ($r = 0.4$)	70.36	45.91	38.33	42.00	49.15
VLA-Corrector ($r = 0.6$)	70.89	52.73	39.17	**46.00**	52.20
VLA-Corrector ($r = 0.8$)	71.07	53.64	38.33	**46.00**	52.26
VLA-Corrector ($r = 1.0$)	**73.21**	**55.91**	**44.17**	44.00	**54.32**

Table 12: **LVM timing during deployment on MetaWorld.** We measure 500 held-out observations after removing 20 warm-up observations. "Full LVM computation" is all work done by the monitor; "wait added to control loop" is the part that the controller actually waits for.

Component	Mean (ms)	p95 (ms)
Preprocessing	1.79	2.06
Frozen vision encoder	21.31	21.46
40M residual MLP	2.79	2.93
Mismatch + threshold	0.91	1.08
Full LVM computation	**26.96**	**27.57**
Wait added to control loop	**0.008**	**0.009**

Table 13: **OGG timing during deployment on an NVIDIA A100-SXM4-40GB.** We compare 200 matched ordinary/OGG pairs after 20 warm-up runs. The random state is restored for each pair, the order is balanced, and CUDA timing is synchronized; every OGG call uses two guidance steps.

Call	Mean latency
Ordinary	232.490 ms
OGG	243.223 ms
Increment per trigger	**10.733 ms**
95% CI	[10.430, 11.053] ms

The trend also shows that the corrector benefits from additional data, but with diminishing returns. Increasing the training ratio from 0.2 to 0.6 brings a clear improvement, while increasing it further from 0.6 to 0.8 changes the macro average only slightly. Full-data training still achieves the best overall result, especially on the Hard split, but the relatively strong performance at moderate data ratios supports our design motivation: VLA-Corrector learns a local consistency signal rather than a full policy or heavy world model, making it data-efficient to train. On the Very Hard split, the small fluctuation across higher ratios indicates that extremely challenging contact-rich tasks are influenced by factors beyond corrector data scale alone, such as disturbance severity and backbone recovery capability.

D.5 Inference-Time Cost and Control Delay

The monitor runs in parallel with the robot's current action. We therefore report two different times: (1) the time needed to complete the monitor's computation, and (2) the part of that time that actually makes the controller wait. We also measure the extra time used by the OGG recovery call. All measurements use synchronized CUDA timing. The LVM measurements are listed in Table 12, the OGG measurements in Table 13, and the physical-control check in Table 14.

As shown in Table 12, the monitor performs 26.96 ms of computation on average and 27.57 ms at p95. Because this work runs while the robot is executing, it adds only 0.008 ms of average waiting time (0.009 ms at p95) to the control loop. Thus, the computation is real, but almost all of it is hidden by parallel execution; the 0.008 ms number is only the extra wait, not the total monitor cost.

Table 13 shows that an ordinary policy call takes 232.490 ms, while an OGG-guided call takes 243.223 ms. OGG therefore adds 10.733 ms only when it is used for recovery after an interrupt; it does not slow down ordinary policy calls.

Discussion. Table 14 summarizes the physical check. MetaWorld is evaluated without a real-time clock, but PiPER must complete one action every 100 ms. The monitor's p95 computation time is 27.57 ms, leaving 72.43 ms in each control period, and none of the 500 measured steps missed its deadline. The monitor therefore remains responsive in the physical setup: its full computation is accounted for, while only 0.008 ms of average waiting is added to the controller. OGG adds 10.733 ms only on the occasional recovery call.

Table 14: **Timing check on AgileX PiPER.** PiPER executes 10 actions per second, so the controller has 100 ms for each action.

Quantity	Result
Control period	100 ms/action (10 Hz)
Full LVM	26.96 ms mean / 27.57 ms p95
p95 remaining margin	72.43 ms
Deadline misses	0 / 500
Wait added to control loop	0.008 ms mean

Table 15: **Asynchronous deployment-path accounting.** Intrinsic computation is measured with CUDA synchronization; exposed wait is the delay visible to the control loop after asynchronous pipelining. OGG is applied only to the single recovery query following an interrupt.

Quantity	Measured result
Full LVM intrinsic computation	26.96 ms mean / 27.57 ms p95
Exposed asynchronous control-path wait	0.008 ms mean / 0.009 ms p95
Ordinary policy call	232.490 ms
OGG-guided policy call	243.223 ms
Increment on the recovery call	10.733 ms/trigger, 95% CI [10.430, 11.053] ms
PiPER control period	100 ms/action (10 Hz)
Observed deadline misses	0/500

D.6 Asynchronous and RTC Comparisons

Because Real-Time Chunking (RTC) periodically refreshes action chunks while the previous chunk is executing, it overlaps with the asynchronous execution path of VLA-Corrector and is a natural comparison point. The two mechanisms nevertheless address different parts of the control problem: RTC schedules the next policy query, whereas VLA-Corrector uses the observed-vs. predicted visual-dynamics mismatch to produce an explicit error signal, stop stale actions, and guide the recovery replan. We therefore report both the deployment-path timing and a direct overlapped-execution success comparison. Table 15 collects the asynchronous timing measurements, Table 16 gives the RTC timing context, and Table 17 reports the direct overlapped-execution experiment.

Table 15 makes the deployment claim precise: asynchronous execution does not remove the LVM's 26.96 ms of intrinsic computation, but only 0.008 ms of mean waiting time is exposed on the control path. The OGG increment is similarly localized to the recovery call, rather than being charged to every action step.

Table 16 gives an analytical timing context rather than a hardware benchmark. It shows that periodic RTC can deliver a newer ordinary chunk, but does not itself provide an explicit failure signal or a recovery-directed target. At the same call budget, RTC must distribute policy calls uniformly, whereas LVM-gated execution allocates the additional replan after an observed persistent mismatch. The direct experiment below tests whether this distinction remains visible in task success when all methods use overlapped execution.

For clarity, all main-text experiments and the previously reported VLA-Corrector results use `rtc_config=None`: RTC itself is disabled in those experiments. This setting does not make LVM monitoring synchronous; the deployment-path measurements above still run the monitor asynchronously while the current action executes. To compare against RTC directly, we additionally run the experiment in Table 17 with asynchronous inference enabled for every method and with an RTC baseline included only for this comparison. For context, the RTC analysis reports 97 ms for inference-time RTC versus 76 ms for vanilla $\pi_{0.5}$ on an RTX 4090, because prefix inpainting requires gradient computation; its training-time variant removes that inference operation only through a separately trained prefix-conditioned $\pi_{0.6}$ checkpoint. These context figures are kept separate from the direct overlapped-execution experiment and the A100 deployment-path measurements above.

As shown in Table 17, the additional experiment directly evaluates VLA-Corrector under overlapped execution rather than inferring its behavior from sequential timing. At comparable policy-call rates,

Table 16: **RTC context under the 10 Hz control-period model.** The response intervals are obtained from the ordinary/guided call latencies and the persistence rule ($p = 5$); they clarify the difference between periodic scheduling and failure-aware truncation.

Configuration	Requery rule	Calls/step	Relative calls	Response under the timing model
LVM-gated truncation	$H = 50$; extra replan after persistent mismatch	0.0379	1.0×	Stale suffix stopped at ≈400 ms
Reviewer-style RTC	Periodic, $s = 3$	0.333	8.8×	Replacement at 297–597 ms
Call-matched RTC	Periodic, $s \approx 26.4$	0.0379	1.0×	Replacement at ≈297–2,937 ms

Table 17: **Direct overlapped-execution comparison on MetaWorld-50.** All methods use the same frozen $\pi_{0.5}$, horizon $H = 50$, asynchronous execution path, and 50 paired episodes per task (2,500 episodes per method). The RTC baseline periodically alternates $s = 26$. The three E_t-gated variants share the same detector and truncation settings and differ only in the triggered replan; the random direction is scaled to the mean OGG guidance norm.

Method	Triggered guidance	Success (95% CI)	Calls/100 steps
RTC, $s = 26$	None	61.3 [59.4, 63.2]	3.85
E_t-gated replan	None	62.4 [60.5, 64.3]	4.08
E_t-gated random	Random	61.9 [60.0, 63.8]	4.12
E_t-gated OGG	**OGG**	**67.1 [65.3, 68.9]**	**3.88**

E_t-gated OGG reaches 67.1% success versus 61.3% for periodic RTC, an improvement of 5.8 points (paired 95% CI [+4.4, +7.2], McNemar $p < 10^{-13}$). Thus, the advantage persists when policy inference is overlapped with action execution and is not explained by a higher call frequency. The matched-norm random control reaches 61.9%, which is 5.2 points below E_t-gated OGG (95% CI [+3.8, +6.6], $p < 10^{-11}$), and does not improve over the ordinary E_t-gated replan (61.9% vs. 62.4%). These results isolate the contribution of the recovery direction: the gain comes from using the LVM-derived error signal to guide OGG toward a corrective target, rather than from arbitrary perturbation or simply requesting another policy chunk.

E Real-World Details

E.1 Robot Platform and Task Suite

Robot platform. All real-world experiments are conducted on an **AgileX PiPER** 6-DoF robotic arm. We use $\pi_{0.5}$ as the action-chunked VLA baseline and evaluate VLA-Corrector on top of the same fine-tuned backbone. Both methods share the same RGB camera observation, language instruction, action horizon, and initial task conditions. Each task is evaluated with 20 trials per method. A trial is counted as successful if the robot completes the specified task without human reset after execution starts.

Task design. We design three groups of real-world tasks with increasing difficulty. The first group evaluates standard pick-and-place behavior, where the original action chunk often remains valid. The second group evaluates precise alignment, where small accumulated errors can cause missed placement, collision, or failed insertion. The third group introduces online human disturbances during precision-sensitive phases, directly testing whether VLA-Corrector can recover when the remaining action chunk becomes stale. Table 18 summarizes the task definitions.

E.2 Per-Task Real-World Results

Table 19 reports per-task success rates. The gains are modest on pick-and-place tasks because the baseline can often complete the task when the object and target remain static. The gains become larger on alignment tasks, where small pose errors are less tolerable. The largest improvement appears in disturbance recovery tasks, where the current action chunk often becomes stale after the object or target is manually moved.

Table 18: **Real-world task suite on AgileX PiPER.** The benchmark contains three task groups and nine tasks in total. The disturbance group reuses manipulation skills from the first two groups but introduces online pose changes during precision-sensitive phases.

Group	Task	Description
Pick-and-place	Cube-to-region	Pick up a cube and place it inside a marked target region.
	Cube-to-drawer-top	Pick up a cube and place it on top of a small drawer platform.
	Object-to-container	Pick up a cube and place it into a specific bowl.
Alignment	Corner placement	Place a cube precisely at the upper-left corner of the drawer top.
	Square-hole insertion	Align and insert a small cube into a square hole or slot.
	Edge alignment	Align an object with a narrow marked edge or target strip.
Disturbance recovery	Moving-object grasp	Move the target object when the gripper is about to grasp it.
	Moving-placement target	Move the placement region after the object is grasped.
	Moving-insertion target	Move the square hole or alignment target before insertion.

Table 19: **Per-task real-world success rates.** Success rate (%, ↑) over 20 trials per task on the AgileX PiPER 6-DoF arm. Group and overall rows report mean ± standard deviation across tasks. Green arrows indicate absolute improvement over the $\pi_{0.5}$ baseline.

Group	Task	$\pi_{0.5}$ Baseline↑	+ VLA-Corrector↑
Pick-and-place	Cube-to-region	75.0	**85.0** ↑ 10.0
	Cube-to-drawer-top	70.0	**80.0** ↑ 10.0
	Object-to-container	65.0	**70.0** ↑ 5.0
	Group Avg.	**70.0 ± 5.0**	**78.3 ± 7.6** ↑ 8.3
Alignment	Corner placement	60.0	**75.0** ↑ 15.0
	Square-hole insertion	50.0	**70.0** ↑ 20.0
	Edge alignment	60.0	**75.0** ↑ 15.0
	Group Avg.	**56.7 ± 5.8**	**73.3 ± 2.9** ↑ 16.6
Disturbance recovery	Moving-object grasp	45.0	**75.0** ↑ 30.0
	Moving-placement target	40.0	**70.0** ↑ 30.0
	Moving-insertion target	35.0	**60.0** ↑ 25.0
	Group Avg.	**40.0 ± 5.0**	**68.3 ± 7.6** ↑ 28.3
Overall	**Avg. across 9 tasks**	**55.6 ± 13.8**	**73.3 ± 7.1** ↑ 17.7

E.3 Real-World Protocol and Task Details

Evaluation protocol. For each real-world task, we collect 10–30 human teleoperated demonstration trajectories and use them to fine-tune the $\pi_{0.5}$ backbone for that task. The same fine-tuned backbone is then used by both the baseline and VLA-Corrector, so the comparison isolates the effect of the inference-time correction module. For each task and each method, we run 20 evaluation trials. Initial object poses are randomized within a small predefined region while keeping the task feasible for the 6-DoF PiPER arm.

Pick-and-place tasks. The pick-and-place group includes placing a cube into a marked region, placing a cube on top of a drawer platform, and placing a small object into a bowl. These tasks are relatively tolerant to small pose errors, so the original action chunk often remains usable. VLA-Corrector mainly helps when occasional drift occurs during approach or release.

Alignment tasks. The alignment group increases the precision requirement. In corner placement, the robot must place a cube at a specific corner of the drawer top. In square-hole insertion, the cube must be aligned with a square opening before insertion. In edge alignment, the robot must align an object with a narrow visual marker or edge. These tasks are more sensitive to accumulated error, making stale action execution more harmful near the final contact.

Disturbance recovery tasks. The disturbance group directly evaluates the open-loop blind spot. We introduce manual perturbations at predefined task phases rather than at fixed time steps. For moving-object grasp, the target object is shifted when the gripper is close to grasping it. For moving-placement target, the robot first grasps the object normally, and the target platform or placement region is shifted before release. For moving-insertion target, the square hole or alignment target is shifted when the robot is about to align or insert. The disturbance is applied once per trial, kept within the camera view, and constrained to a physically recoverable range. The same timing rule and disturbance range are used for both the baseline and VLA-Corrector.

E.4 Observed Real-World Failure Modes

Despite the improvements, VLA-Corrector still fails in several real-world cases. If the target object or fixture is moved beyond the reachable region, or if the disturbance happens after the gripper has entered an unfavorable pose, the robot may no longer have enough workspace or time to recover through a single corrective replan. In tight alignment tasks, a 6-DoF arm without force feedback may fail due to contact geometry, friction, or small height errors even when the visual target is corrected. We also observe failures caused by visual ambiguity, such as partial occlusion by the gripper or poor contrast between the object and the target region. Finally, OGG still depends on the frozen $\pi_{0.5}$ action prior: it can bias the next generation toward a better corrective direction, but it cannot create a recovery behavior that the backbone policy cannot represent.

E.5 Real-World Demonstration Examples

We provide representative real-world demonstration examples mainly for the disturbance recovery tasks. In these cases, the object or target fixture is manually shifted during execution, allowing us to visualize the practical robustness of VLA-Corrector under human-induced disturbances.

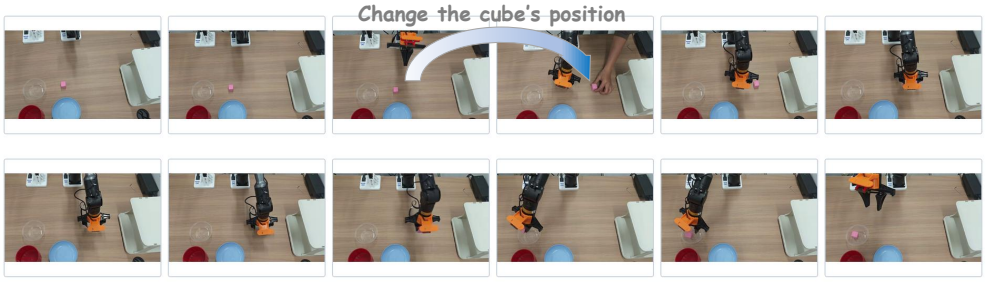


Figure 9: **Demo: Moving-object grasp.** The robot picks up the cube and places it into the white bowl, while a human manually changes the cube's position during the process.

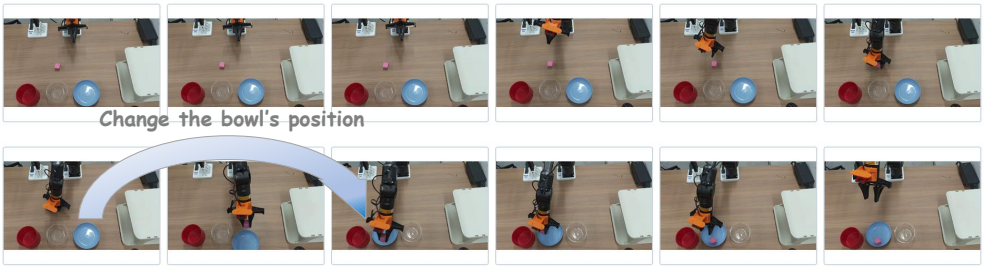


Figure 10: **Demo: Moving-placement target.** The robot picks up the cube and places it into the blue bowl, while a human manually changes the bowl's position during the process.

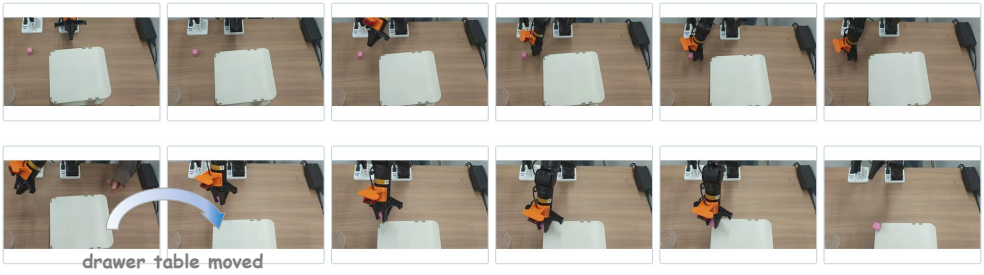


Figure 11: **Demo: Moving-insertion target.** The robot picks up the cube and places it at the upper-left corner of the drawer top, while a human manually changes the drawer's position during the process.

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 Section 1 describe the detect-and-correct framework and its action-horizon trade-off; Section 4 evaluates these claims on simulation benchmarks and real-world 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: Appendix E.4 discusses limits from reachability, contact and occlusion, and the frozen backbone's action prior; Appendix D.2 reports limited cross-domain transfer and the importance of domain-matched demonstrations.

   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 presents an empirical method and its defining equations, rather than theoretical results requiring formal assumptions and proofs.

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: Sections 2–4 define the method and evaluation; Appendices B, C, and E give runtime parameters, corrector architecture and training, benchmark protocols, and real-world task details.

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: [No]

   Justification: The code and installation, training, and evaluation instructions are available at `https://github.com/ZJU-OmniAI/vla-corrector`. However, raw demonstrations and experiment-specific fine-tuned policy and corrector checkpoints are not included in the release, so the complete experimental artifacts are not openly provided.

   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: Appendices B.2, C.1, and C.3 describe inference parameters, evaluation episodes and data splits, and training hyperparameters; Appendix E.3 details the real-world 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: Table 2 reports 95% Wilson confidence intervals and Appendix C.1 specifies paired evaluation seeds; Table 19 reports standard deviations across tasks. These statistics characterize the indicated experiments and do not constitute a significance test for every reported comparison.

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: [No]

   Justification: Appendix C.4 identifies the eight A100-SXM4-40GB GPUs, and Appendix D.5 reports inference timings, but per-run training time and total project compute are not fully reported.

   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 authors have reviewed the NeurIPS Code of Ethics and confirm that the research conforms to its requirements.

   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 paper evaluates robotic manipulation robustness and efficiency, but does not include an explicit discussion of both positive and negative societal impacts.

    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 manuscript does not describe the release of a new general-purpose pre-trained model, scraped dataset, or other asset with the high misuse risks addressed by this question.

    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: [No]

Justification: The manuscript cites the source works for existing VLA backbones and benchmarks, but does not explicitly enumerate the licenses and terms of use of all assets.

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: [Yes]

Justification: The released code at `https://github.com/ZJU-OmniAI/vla-corrector` includes an Apache-2.0 license, environment specifications, and installation, training, and evaluation instructions. Its documentation explicitly identifies the datasets and checkpoints that users must obtain or prepare separately.

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 reported evaluations measure robot performance, not human behavior; manual object disturbances are part of the robot evaluation protocol in Appendix E.3. No crowdsourcing or human-subject study is described in the supplied manuscript.

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: No human-subject study is described in the manuscript; the real-world protocol evaluates a robotic arm under manually introduced object disturbances (Appendix E.3), rather than collecting participant outcomes.

    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: [Yes]

    Justification: Sections 2–3 identify and describe the pretrained vision-language-action backbones used in the method, with implementation details in Appendix C. The source manuscript reports generative-AI assistance for writing, source organization, and formatting only, without changes to the method, experiments, or results.

    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.