REFLECTVLA: EVENT-ANCHORED PAIRED TRAJECTORY CONDITIONING FOR CORRECTIVE RE-EXECUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) methods have achieved strong performance for robotic manipulation. However, models trained on success-only datasets often fail when driven into out-of-distribution states. Existing correction strategies use different intervention interfaces, while paired trajectory conditioning directly represents the deviation between failed and successful behaviors for controllable re-execution. In this paper, we propose ReflectVLA, a framework that learns corrective re-execution from paired successful and failed trajectories. The core of ReflectVLA is an event-anchored Contrastive Trajectory Encoder (CTE) that learns paired trajectory representations, capturing how failed trajectories diverge from successful references. The deviation representation is encoded as correction tokens and injected into the VLA inference process. A correction adapter enables the policy to interpret these correction tokens and generate re-execution actions while preserving the base VLA behavior during normal execution. During inference, an external detector identifies an incipient failure. When the trigger arrives before an irreversible environmental change, ReflectVLA interrupts execution, rolls back to a visited pre-failure keyframe, and retrieves the most relevant successful reference from a Success Bank. The paired success–failure trajectory is jointly encoded to guide correction. Experiments demonstrate that ReflectVLA enhances recovery performance under this early, reversible intervention setting.

1 INTRODUCTION

Vision-Language-Action (VLA) models have become an important paradigm for building robotic agents. Representative models (Kim et al., 2025a; Black et al., 2025b;a; Bjorck et al., 2025) have demonstrated strong manipulation capabilities in complex tasks and scenes.

Despite recent progress, typical VLA models remain limited by success-only imitation. Trained only on successful expert demonstrations, they receive little supervision on execution failures. This concentrates policies on the success manifold and leaves pose drift and missed grasps weakly represented during execution. As shown in Figure 1 b), these deviations push them into out-of-distribution states, where errors compound and ultimately cause task failure. Failure detection and correction therefore remain central components of reliable VLA execution.

Recent methods address complementary parts of this execution problem. Some approaches (Lin et al., 2026; Li et al., 2025) react after failure through action biasing, while recent methods (Yang et al., 2025; Ma et al., 2026) use external predictors, backtracking, and test-time action selection. These methods use distinct recovery interfaces, while paired successful–failed trajectory conditioning provides a complementary way to represent execution deviations for re-execution.

In summary, there are two key challenges: **1)** Success-only imitation leaves VLA policies with limited knowledge of failure semantics and poor recovery ability under out-of-distribution deviations. **2)** Existing methods rarely model execution errors at the trajectory level, making it difficult to capture how small deviations accumulate and lead to failure.

In this paper, we propose ReflectVLA, a detector-triggered corrective re-execution framework that learns deviation-aware correction representations from our synthesized success–failure data. Instead

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

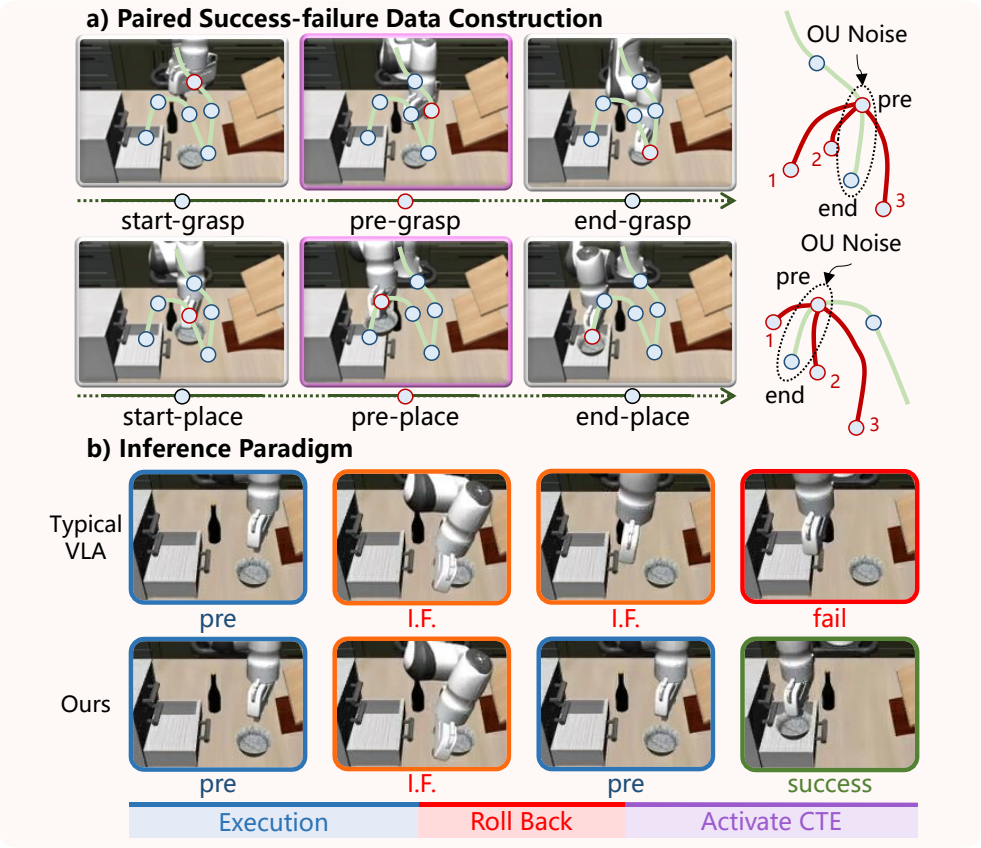


Figure 1: **a)** We add OU noise between keyframes to generate failure data. **b)** Unlike typical VLAs, ReflectVLA detects incipient failures and uses the CTE for corrective re-execution.

of directly regressing corrective displacements from the failure state, ReflectVLA first rolls back to a pre-failure keyframe once an incipient failure is detected, where the task remains controllable before an irreversible environmental change. From this recovered state, ReflectVLA encodes how the failed execution segment deviates from a retrieved successful reference and uses this deviation as soft correction guidance. It enables the VLA policy to re-infer task-consistent actions.

To achieve this, ReflectVLA introduces an event-anchored Contrastive Trajectory Encoder. The failed and successful trajectories are anchored around key execution events. The encoder compares them at the representation level and summarizes their deviation into correction tokens. These tokens are soft conditioning signals interpreted by a correction adapter, which is activated only during correction. Experiments show that ReflectVLA achieves higher task success rates and recovery performance than base methods.

We evaluate correction conditional on a shared early-failure trigger and a recoverable rollback interface. The detector and rollback define the intervention interface, while ReflectVLA learns the paired-trajectory conditioning used for re-execution. This decomposition targets early, reversible execution deviations and separates correction quality from trigger coverage and state restoration.

Our main contributions can be summarized as follows:

- We introduce **ReflectVLA**, a failure-aware framework for VLA policies. It learns deviation-aware correction from synthesized success–failure trajectory pairs and recovers from a pre-failure keyframe upon incipient failure.
- We propose an **event-anchored Contrastive Trajectory Encoder** to model trajectory-level deviations between failed executions and successful references. The encoded correction tokens softly condition VLA inference.

- We develop a **retrieval-augmented correction strategy** that dynamically retrieves relevant successful references from a Success Bank and activates a correction adapter to enable correction-aware policy modulation.

2 Related Work

Vision-Language-Action Models. Robot learning has progressed from diverse datasets and policies (Brohan et al., 2023; Zitkovich et al., 2023; O'Neill et al., 2024) to generalist imitation systems with scalable action generation (Chi et al., 2023; Ghosh et al., 2024). OpenVLA (Kim et al., 2025b) formulates control as autoregressive action-token prediction, and OpenVLA-OFT (Kim et al., 2025a) improves frequency and accuracy via parallel continuous chunks. Continuous generative models such as π_0 and $\pi_{0.5}$ (Black et al., 2025b;a), together with related foundation systems (Bjorck et al., 2025; Liu et al., 2025), advance unstructured manipulation. Complementary designs strengthen spatiotemporal grounding and intermediate reasoning through trajectory prompts, spatial representations, or embodied chain-of-thought (Zheng et al., 2025; Qu et al., 2025; Zhao et al., 2025). However, they are still optimized primarily on successful demonstrations.

Robot Failure Correction. Prior work can be grouped into three correction paradigms: external guidance, direct execution intervention, and policy-integrated correction. External-guidance methods derive explicit corrective guidance from execution observations or external feedback and condition policies on this guidance to generate recovery actions (Ye et al., 2026; Zeng et al., 2026; Dai et al., 2025; Xia et al., 2025). Direct execution intervention methods employ auxiliary supervisory or recovery modules to alter the robot's subsequent execution in response to failure-related information. FailSafe (Lin et al., 2026) predicts executable recovery deltas from verified failure–action pairs, whereas CycleVLA (Ma et al., 2026) combines progress-triggered assessment with subtask backtracking to support corrective re-execution. Related methods also refine action poses before or after execution (Li et al., 2025; Yang et al., 2025). Their recovery interfaces differ in how they use prior execution history and when they intervene. Policy-integrated correction methods develop corrective capabilities within the policy through training (Xiao et al., 2025; Zheng et al., 2026; Liu et al., 2026). Conditioning corrective action generation on differences between failed and successful trajectories remains an open design space. ReflectVLA learns corrective representations from paired trajectories for robot re-execution.

3 Methodology

Our goal is to equip a pretrained VLA with paired-trajectory conditioning for corrective re-execution before an execution deviation becomes irreversible.

3.1 Overview

Given a language instruction g and multimodal observation o_t, a pretrained VLA policy π_θ generates robot actions but may become unreliable once execution deviates from the successful demonstration distribution. ReflectVLA formulates failure handling as *early intervention followed by re-execution from a pre-failure keyframe*. As shown in Figure 2, ReflectVLA consists of a frozen base VLA model, a Success Bank, an event-anchored Contrastive Trajectory Encoder (CTE), and a correction adapter. Given a failed trajectory segment τ^f and a retrieved successful reference τ^s, the CTE encodes their event-anchored deviation into correction tokens C, which condition the VLA policy to regenerate actions from the pre-failure state:

$$\begin{aligned} C &= E_\psi\big(\tau^f, \tau^s\big)\,, \\ a^{\mathrm{corr}} &\sim \pi_{\theta,\phi}\big(\cdot \mid o_{t_p}, g, C\big)\,. \end{aligned} \tag{1}$$

Here, E_ψ is the CTE parameterized by ψ, and t_p denotes the online pre-failure keyframe used for correction. The variables θ and ϕ denote the parameters of the frozen base VLA and the trainable correction adapter, respectively, while a^{corr} is the regenerated corrective action sequence. During training, only ψ and ϕ are optimized using our synthesized success–failure trajectory pairs. During normal execution, the CTE, correction adapter, and Success Bank are bypassed.

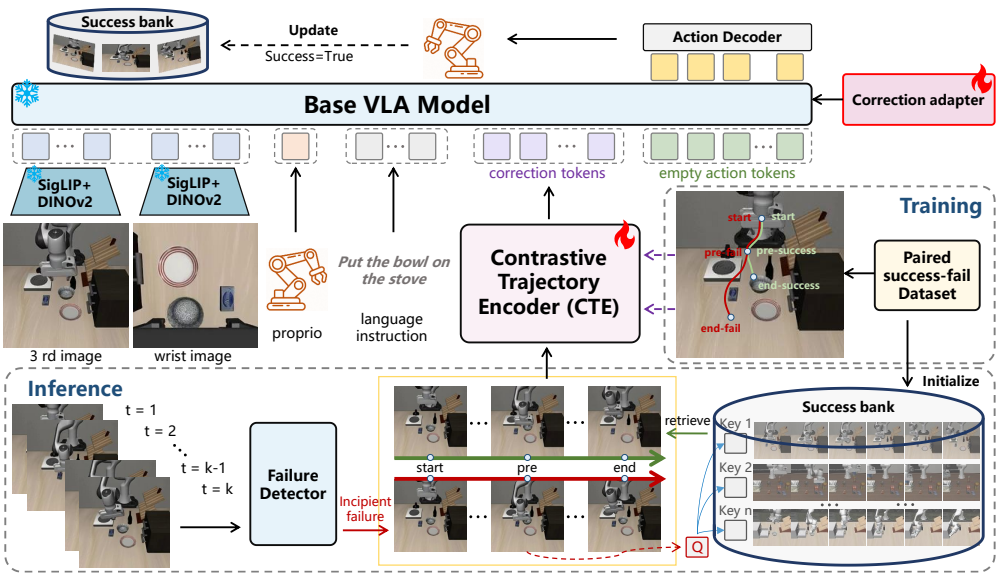


Figure 2: **Overview of ReflectVLA.** A frozen base VLA is augmented with a trainable event-anchored CTE and correction adapter. Paired success–failure trajectories initialize the Success Bank and are encoded into correction tokens.

3.2 Contrastive Trajectory Encoder (CTE)

The Contrastive Trajectory Encoder (CTE) converts an event-anchored success–failure trajectory pair into a fixed set of correction tokens. CTE models the trajectory-level deviation between the two execution branches and provides it as soft conditioning for the VLA policy. Let $\tau^b = \{(I_m^b, s_m^b, a_m^b)\}_{m=1}^M$ denote an anchored trajectory segment, where $b \in \{succ, fail\}$ indicates the successful or failed branch, and I_m^b, s_m^b, and a_m^b represent the multi-view images, robot state, and executed action at the m-th step, respectively. The success and failure encoders share weights.

For each branch, CTE first extracts multimodal trajectory features using the encoder shown in Figure 3. The third-person and wrist images are encoded by the vision encoder, and their features are fused by a trainable visual projector. The robot states and actions are independently embedded by trainable state and action encoders. These features are concatenated and projected into a shared token space. For efficiency, image and state features are extracted only at the anchor frames (start, pre, and end keyframes). Action features are encoded over the full trajectory sequence and compressed into three corresponding tokens.

To preserve temporal order and distinguish the two trajectory branches, positional embeddings and trajectory-type embeddings are added to each token. The resulting token sequences are concatenated and processed by multi-head self-attention blocks, allowing the encoder to jointly capture temporal dependencies within each trajectory and representation-level differences across the successful and failed branches. Finally, a set of K learnable query tokens Q_L attends to the encoded trajectory features through multi-head cross-attention. The query outputs are mapped by a final projector to obtain the correction tokens. The learnable queries compress the variable-length trajectory pair into a fixed number of correction tokens, which summarize correction-relevant deviation cues. These tokens are subsequently inserted into the VLA input sequence and interpreted by the correction adapter during re-execution.

3.3 Training Data Construction

To model execution deviations, we construct paired success–failure trajectories from success-only demonstrations by introducing localized perturbations around manipulation-critical events, including grasping, placement, contact, and goal completion, as illustrated in Figure 1. The construction preserves the initial condition, and task context, producing task-consistent failures.

For each successful trajectory, we identify task-relevant events from gripper transitions, end-effector motion, and task-completion signals, and represent each event using three temporal anchors $(t_{\text{start}}, t_{\text{pre}}, t_{\text{end}})$. Here, t_{start} marks the beginning of the interaction stage, t_{pre} is the latest recoverable state before the deviation becomes critical, and t_{end} marks the end of the event and its immediate consequence. These anchors define semantically consistent segments for comparing successful and failed executions.

Failure trajectories are synthesized by injecting Ornstein–Uhlenbeck (OU) noise into the translational and axis–angle rotational action components within the anchored event window:

$$\widetilde{\mathbf{a}}_t = \begin{cases} \mathbf{a}_t + \boldsymbol{\epsilon}_t^{\text{OU}}, & t_{\text{pre}} \le t \le t_{\text{end}}, \\ \mathbf{a}_t, & \text{otherwise}, \end{cases} \tag{2}$$

where $\boldsymbol{\epsilon}_t^{\text{OU}}$ denotes temporally correlated Ornstein–Uhlenbeck (OU) noise sampled with different translational and rotational scales.

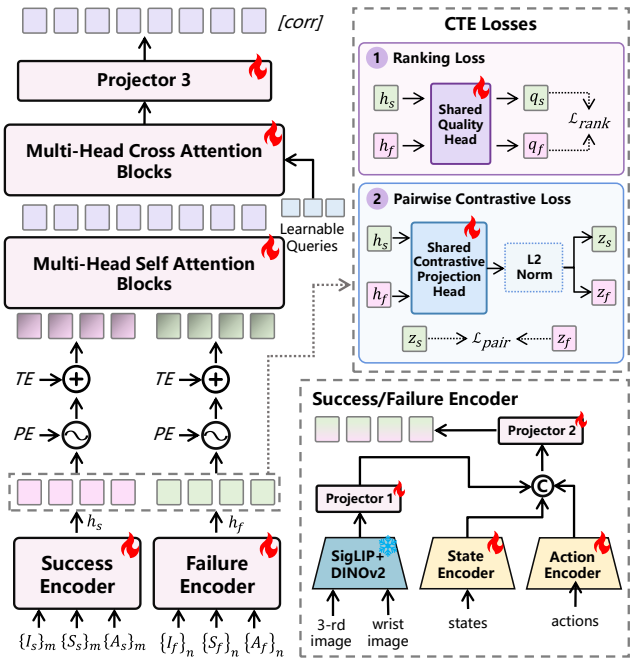


Figure 3: **Framework of CTE.** Event-anchored success–failure trajectories are encoded into correction tokens with auxiliary ranking and pairwise contrastive supervision.

In contrast to independently sampled Gaussian perturbations, OU noise evolves smoothly over time and therefore better reflects temporally accumulated execution errors, such as end-effector drift, pose misalignment, and unstable approach motions. Each perturbed action sequence is re-executed from the same initial state, and only trajectories that fail to satisfy the task-completion condition are retained.

By perturbing different event windows and sampling multiple OU-noise intensities, a single successful demonstration can generate multiple failed trajectories with diverse failure modes and severities. The resulting training unit consists of one successful trajectory and a set of associated failed trajectories, together with their corresponding $(t_{\text{start}}, t_{\text{pre}}, t_{\text{end}})$ anchors. For each successful trajectory in the original dataset, 3 to 9 additional failure trajectories (averaging 6) were constructed. Detailed task-specific event detection rules and perturbation configurations are provided in the Appendix A.

3.4 Training Strategy

We train ReflectVLA on our synthesized paired success–failure dataset. As illustrated in Figure 2, the event-anchored success–failure trajectory pair is processed by the CTE to generate correction tokens, which condition the correction adapter to predict actions. With the base VLA frozen, the corrective action loss updates the CTE through the correction adapter, encouraging it to encode paired-trajectory cues useful for re-execution. We randomly sample the endpoint of each failure trajectory between its recorded pre-frame and end-frame to expose the model to varying trajectory lengths and to match inference-time failure prefixes that may terminate before event completion.

We use two auxiliary objectives, as shown in Figure 3. Given the encoded success and failure trajectory summaries h_i^s and h_i^f, a shared quality head predicts scalar scores q_i^s and q_i^f. We employ a ranking loss that assigns a higher quality score to the successful trajectory:

$$\mathcal{L}_{\text{rank}} = \frac{1}{B_c} \sum_{i=1}^{B_c} \text{softplus}\left(q_i^f - q_i^s\right), \tag{3}$$

where B_c is the number of paired samples. This objective encourages $q_i^s > q_i^f$ and provides the CTE with supervision on the relative quality of the two execution branches.

To make the CTE sensitive to success–failure trajectory differences, the summaries are also projected and ℓ_2-normalized as z_i^s and z_i^f. We then apply a margin-based pairwise contrastive loss:

$$\mathcal{L}_{\text{pair}} = \frac{1}{B_c} \sum_{i=1}^{B_c} \max\left(0,\, \cos(z_i^f, z_i^s) - \epsilon\right), \tag{4}$$

where ϵ is the similarity margin. It prevents the success and failure representations from collapsing into indistinguishable embeddings, thereby preserving deviation-aware information. The auxiliary heads are only used during training.

The complete CTE and training objectives are

$$\begin{aligned} \mathcal{L}_{\text{CTE}} &= \mathcal{L}_{\text{rank}} + \lambda_{\text{pair}} \mathcal{L}_{\text{pair}}, \\ \mathcal{L}_{\text{total}} &= \mathcal{L}_{\text{act}} + \lambda_{\text{CTE}} \mathcal{L}_{\text{CTE}}, \end{aligned} \tag{5}$$

where λ_{pair} and λ_{CTE} balance the pairwise contrastive supervision and the overall CTE objective, respectively. We set $\mathcal{L}_{\text{act}}$ to be the same as in the base model and GT to be the actions from successful trajectory. We freeze the base VLA and jointly optimize the CTE and the correction adapter. $\mathcal{L}_{\text{act}}$ provides the primary supervision for learning correction tokens, while λ_{pair} and λ_{CTE} serve as auxiliary regularizers.

3.5 Inference

During normal execution, the CTE and the correction adapter remain inactive, such that the policy follows the base VLA behavior. For the reported system-level runs, each seed/task initializes a task-specific Success Bank from training successes and incrementally adds verified successful evaluation trajectories from the same task. Bank contents are not shared across seeds or tasks. For the shared-trigger audit, the bank is kept fixed to ensure identical retrieval across conditions. Each bank entry is represented as

$$\mathcal{B} = \left\{ \left(k_j^{\text{init}}, k_j^{\text{pre}}, \tau_j^s\right) \right\}_{j=1}^{N}, \tag{6}$$

where k_j^{init} and k_j^{pre} denote the robot states at the initial and pre-event keyframes of the successful trajectory τ_j^s, respectively. For notational simplicity, Eq. (6) suppresses the event index and auxiliary metadata. The full multi-event bank representation is given in Appendix A.2.

We use a lightweight rule-based detector over task-relevant signals and the simulator target-object position. It identifies failure precursors, with the trigger occurring 13.58 frames on average before complete failure. When detected sufficiently early, the trigger precedes an irreversible scene change and permits rollback to a visited pre-frame. The detector returns the failure time t_f and an online pre-frame $t_p < t_f$. In the reported protocol, the rollback target is the same visited frame, denoted t_{rb}. ReflectVLA rolls back to t_{rb} by closed-loop tracking of recorded end-effector poses, then retrieves a reference in two stages: five candidates by initial-state similarity followed by pre-event-state similarity. The detector and rollback protocol are detailed in Appendix A.3 and Appendix A.4.

The failed segment τ^f and reference τ^{s*} are aligned to the same execution event. The CTE maps the pair to correction tokens and the adapter is activated for re-execution. Verified successful re-executions are added to the Success Bank during the system-level evaluation. The shared-trigger audit uses the same fixed bank, detector, and rollback for all conditions.

4 Experiments

4.1 Implementation Details

For the system-level experiments, we instantiate ReflectVLA with the 7B OpenVLA and OpenVLA-OFT backbones: the discrete-action variant uses autoregressive action-token prediction, while the continuous-action variant predicts action chunks from two-camera observations and proprioceptive inputs with an ℓ_1 objective. All dense VLA parameters and the shared CTE visual encoders are

Table 1: Quantitative results on the LIBERO benchmark. Each suite is evaluated using three seeds, with 500 trials per seed (50 per task). Average success rates and sample standard deviations are reported. **R. = Retry.**

Methods	LIBERO-Spatial	LIBERO-Object	LIBERO-Goal	LIBERO-10	Avg.
OpenVLA	0.850 ± 0.008	0.886 ± 0.010	0.753 ± 0.017	0.522 ± 0.012	0.753 ± 0.012
OpenVLA+R.	0.862 ± 0.008	0.890 ± 0.008	0.770 ± 0.011	0.554 ± 0.012	0.769 ± 0.009
ReflectVLA	$\mathbf{0.926 \pm 0.006}$	$\mathbf{0.937 \pm 0.005}$	$\mathbf{0.845 \pm 0.009}$	$\mathbf{0.604 \pm 0.011}$	$\mathbf{0.828 \pm 0.008}$
Δ (pp)	+ 6.4	+ 4.7	+ 7.5	+ 5.0	+ 5.9
OpenVLA-OFT	0.982 ± 0.009	0.977 ± 0.004	0.969 ± 0.003	0.941 ± 0.002	0.968 ± 0.003
OpenVLA-OFT+R.	0.989 ± 0.003	0.980 ± 0.002	0.971 ± 0.001	0.949 ± 0.003	0.972 ± 0.002
ReflectVLA	$\mathbf{0.993 \pm 0.003}$	$\mathbf{0.989 \pm 0.003}$	$\mathbf{0.981 \pm 0.001}$	$\mathbf{0.966 \pm 0.002}$	$\mathbf{0.982 \pm 0.001}$
Δ (pp)	+ 0.4	+ 0.9	+ 1.0	+ 1.7	+ 1.0

frozen. For the continuous-action variant, the pretrained action head and proprioceptive projector are also fixed. The correction adapter is a separate LoRA branch attached to all linear projections of the VLA transformer, with a rank of 32 and a scaling factor of 16. The CTE contains 2 self-attention blocks and 3 cross-attention blocks with 16 heads, where 32 learnable queries generate the correction tokens. We set the contrastive margin to 0.2 and $\lambda_{\text{CTE}} = \lambda_{\text{pair}} = 0.5$. We train the CTE and LoRA parameters for 100K steps on eight NVIDIA A6000 GPUs using a learning rate of 5×10^{-4}, a per-GPU batch size of 1, and an effective batch size of 32. These experiments allow at most two rollback-and-correction attempts per episode.

4.2 Evaluation Results

Quantitative Results. We aim to show that our model consistently improves standard VLA baselines. All methods are evaluated under the same maximum number of environment steps for each task. For each base policy, the Retry and ReflectVLA variants use the same failure detector and state-based rollback procedure. After returning to the same pre-failure state, Retry directly re-executes the base policy, whereas ReflectVLA conditions re-execution on the correction tokens generated by the CTE. Each task suite is evaluated using three random seeds with 500 trials per seed, and we report the mean success rate (SR) and sample standard deviation. The task and episode order follows the fixed benchmark schedule.

As shown in Table 1, ReflectVLA consistently outperforms both the corresponding base policies and their Retry variants across all four LIBERO suites. When integrated with OpenVLA, ReflectVLA achieves an average SR of 82.8%, compared with 75.3% for OpenVLA and 76.9% for OpenVLA+Retry. Relative to the controlled Retry baseline, ReflectVLA improves LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10 by 6.4, 4.7, 7.5, and 5.0 pp, respectively, resulting in an average gain of 5.9 pp. ReflectVLA also improves the stronger OpenVLA-OFT backbone from 97.2% with Retry to 98.2%, with gains of 0.4, 0.9, 1.0, and 1.7 pp across the four suites. Since Retry and ReflectVLA share identical failure detection and rollback mechanisms, these consistent improvements demonstrate that the learned deviation-conditioned guidance provides benefits beyond simply allowing an additional execution attempt. These results indicate that our correction conditioned by paired trajectories remains effective across both discrete- and continuous-action VLA backbones. The shared-trigger analysis below uses the same OpenVLA evaluation set as Appendix Table 7: three seeds and 500 rollouts per suite. Table 3 accounts for all 6,000 rollouts and identifies 1,364 true-positive triggers. Table 2 compares correction pathways on exactly those triggers.

Qualitative Results. Figure 4 shows corrective re-execution during grasping, placement, drawer opening, stove activation, and object transfer. Each rollout preserves the progress before the triggered interaction and re-executes the affected event from the rolled-back state.

4.2.1 Shared-trigger mechanism audit

We use the OpenVLA evaluation set underlying Appendix Table 7: three seeds and 500 rollouts per suite, or 1,500 rollouts for each LIBERO suite. The detector labels the first trigger from the uninterrupted base-policy continuation. Conditions share each true-positive trigger, rollback state,

Table 2: Conditional recovery (%) on the same true-positive triggers for inference-time information interventions. Each LIBERO suite contributes the true-positive triggers counted in Table 3. The pooled column aggregates all 1,364 triggers. Counts are shown in parentheses.

Condition	Spatial	Object	Goal	10	Pooled
Full ReflectVLA	**54.5** (114/209)	**49.4** (77/156)	**40.7** (138/339)	**18.6** (123/660)	**33.1** (452/1364)
Without success actions	45.0 (94/209)	42.3 (66/156)	35.1 (119/339)	15.0 (99/660)	27.7 (378/1364)
Success-reference only	33.5 (70/209)	28.8 (45/156)	25.1 (85/339)	10.0 (66/660)	19.5 (266/1364)
Failure-prefix only	35.9 (75/209)	32.1 (50/156)	27.1 (92/339)	11.1 (73/660)	21.3 (290/1364)
Retrieved-action replay	30.1 (63/209)	26.9 (42/156)	22.1 (75/339)	8.9 (59/660)	17.5 (239/1364)
Without type embeddings	43.1 (90/209)	39.7 (62/156)	33.0 (112/339)	13.0 (86/660)	25.7 (350/1364)
Retry	8.6 (18/209)	3.8 (6/156)	7.4 (25/339)	7.3 (48/660)	7.1 (97/1364)

Table 3: Failure-detector accounting on the OpenVLA evaluation set used for the shared-trigger analysis. Each LIBERO suite contains 1,500 evaluation rollouts. TP counts define the suite denominators in Table 2. P, R, and FPR denote precision, recall, and false-positive rate.

Suite	TP	P (%)	R (%)	FPR (%)
Spatial	**209**	94.14	92.89	1.02
Object	**156**	91.76	91.23	1.05
Goal	**339**	96.58	91.62	1.06
LIBERO-10	**660**	98.51	92.05	1.28

retrieved reference, and remaining action budget. Only the correction information pathway changes. Appendix A.7 specifies this accounting.

Full obtains the highest pooled conditional recovery, 452/1,364 true-positive triggers (33.1%), compared with 97/1,364 (7.1%) for Retry. The recovered-count differences are consistent with Table 1: Full exceeds Retry by 96, 71, 113, and 75 episodes on Spatial, Object, Goal, and LIBERO-10, corresponding to the reported 6.4, 4.7, 7.5, and 5.0 pp suite-level gains over 1,500 rollouts. The interventions remove one information pathway at a time, while retrieved-action replay tests whether reference actions alone account for the gain. Table 3 supplies the shared detector accounting. The pooled 33.1% is micro-averaged over all true-positive triggers. The 40.81% OpenVLA recovery in Appendix Table 7 is the macro-average of the four suite recoveries.

4.3 Mechanism and Component Analysis

We conduct five component ablations to evaluate the main components, as reported in Table 4. All ablations are evaluated using the same single seed, and success rates are reported.

W/o Learnable Queries Q_L. We replace the learnable queries in the CTE cross-attention blocks with the encoded pre-frame feature while keeping the remaining architecture unchanged. This variant achieves an average success rate of 81.45%, 1.45 pp below the full model, with the largest decrease on LIBERO-Goal (84.6% to 82.4%). The result suggests that a single pre-frame representation is less effective at summarizing deviations distributed across the paired trajectories, whereas learnable queries can extract multiple correction-relevant cues.

W/o Auxiliary Losses $\mathcal{L}_{\mathrm{CTE}}$. We remove the ranking and pairwise contrastive losses and train the CTE and correction adapter using only the action-prediction objective. The average success rate decreases from 82.90% to 80.45%. Although action supervision alone learns useful correction tokens, the improvement from the full model shows that explicit success–failure discrimination provides complementary regularization for deviation-aware representation learning.

W/o Event Anchoring. We remove the start, pre-event, and end-event anchors and instead encode the complete trajectories. This variant obtains an average success rate of 79.40%, a 3.50 pp decrease from the full model, including a drop from 84.6% to 79.8% on LIBERO-Goal. Complete trajectories introduce task-irrelevant history and weaken the temporal alignment between corresponding execution events, while event anchoring focuses the comparison on manipulation-critical stages.

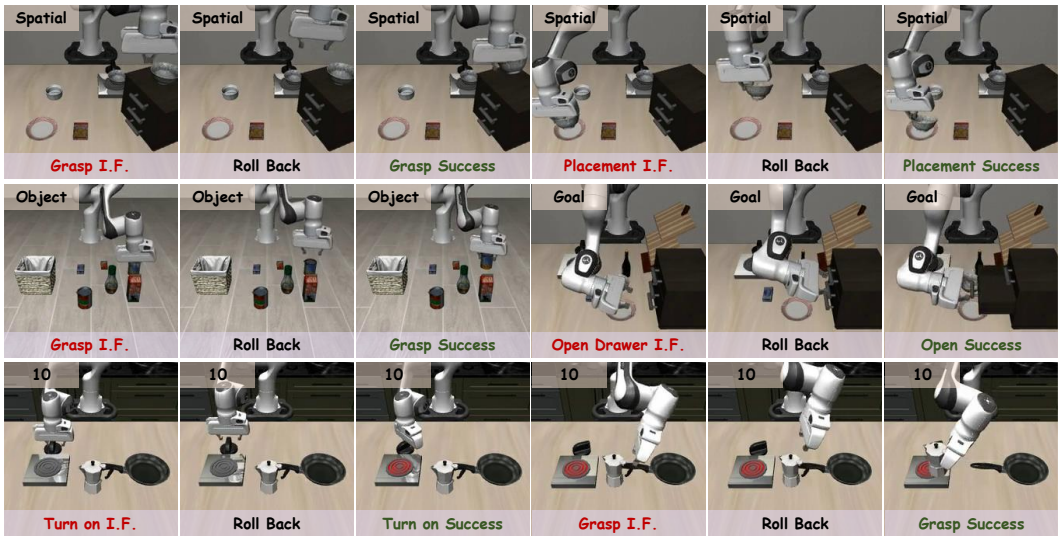


Figure 4: Qualitative results across four LIBERO task suites. I.F. denotes a detected incipient failure.

Settings	Spatial	Object	Goal	10	Avg
W/o Q_L	91.6	92.2	82.4	59.6	81.45
W/o $\mathcal{L}_{\text{CTE}}$	90.2	91.6	81.8	58.2	80.45
W/o Event Anchoring	88.8	90.4	79.8	58.6	79.40
W/o Paired Conditioning	89.6	91.0	78.8	57.2	79.15
Rand. Ret.	86.4	88.6	76.8	54.0	76.45
Full model	**92.6**	**93.8**	**84.6**	**60.6**	**82.90**

Table 4: Single-seed component ablations (success rate, %). Rand. Ret. denotes Random Retrieval.

W/o Paired Conditioning. We remove paired success–failure comparison and generate guidance tokens using only the retrieved successful trajectory. The average success rate decreases to 79.15%, 3.75 pp below the full model, with a pronounced reduction on LIBERO-Goal (84.6% to 78.8%). A successful reference indicates the desired behavior but does not identify how the current failed execution deviates from it, supporting the need for explicit trajectory-level comparison.

Random Retrieval during Inference. We retain the complete CTE architecture but randomly select a successful trajectory from the same task setting instead of using similarity-based retrieval. This produces the lowest average success rate of 76.45%, corresponding to a 6.45 pp decrease from the full model and a 7.8 pp drop on LIBERO-Goal. The result shows that successful references are not interchangeable and that alignment between the retrieved pre-event state and the current recovery state is critical for generating correction tokens.

5 Conclusion

ReflectVLA provides event-anchored paired trajectory conditioning for corrective VLA re-execution. Across OpenVLA and OpenVLA-OFT, it improves the shared-interface Retry baseline, and the shared-trigger analysis on the OpenVLA evaluation set shows that the complete paired pathway has the highest pooled conditional recovery among the tested conditions. The method's operating setting is early, reversible execution: a shared trigger identifies the intervention point, rollback restores a previously visited robot configuration, and the learned adapter uses the observed failure prefix together with a successful reference to regenerate the affected interaction. This interface cleanly separates trigger coverage, state restoration, and paired conditioning, providing a direct basis for extending the method to learned triggering and scene-state-aware recovery.

6 AI USE STATEMENT

In this work, we used generative AI tools for proofreading, improving the clarity and readability of the manuscript, and assisting with the analysis and organization of experimental results. We have not used generative AI tools for generating research ideas, designing the methodology, implementing the core code, conducting experiments, or making scientific claims and conclusions. Additionally, we used generative AI tools for language refinement and checking the overall consistency of the manuscript and experimental descriptions. We have reviewed all AI-assisted work and verified the correctness of the final manuscript, experimental analysis, and reported results. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

7 REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide detailed descriptions of the proposed framework, training procedure, and experimental settings in the main paper and supplementary materials. The implementation details, including model architectures, training configurations, optimization settings, and inference procedures, are described in Section 4.1 and Appendix A. The complete source code used for training and evaluation is provided in the Supplementary Material. The supplementary materials also include additional details of dataset construction, trajectory generation procedures, and experimental protocols necessary to reproduce our results.

REFERENCES

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, et al. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert 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. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025b. doi: 10.15607/RSS.2025.XXI.010.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S Ryoo, Grecia Salazar, Pannag R Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.025.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.026.

Yinpei Dai, Jayjun Lee, Nima Fazeli, and Joyce Chai. Racer: Rich language-guided failure recovery policies for imitation learning. In *IEEE International Conference on Robotics and Automa-*

tion (ICRA), pp. 15657–15664, 2025. URL https://doi.org/10.1109/ICRA55743.2025.11127799.

Dibya Ghosh, Homer Rich 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 *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.090.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025a. doi: 10.15607/RSS.2025.XXI.017.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, et al. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025b. URL https://proceedings.mlr.press/v270/kim25c.html.

Chenxuan Li, Jiaming Liu, Guanqun Wang, Xiaoqi Li, Sixiang Chen, Liang Heng, Chuyan Xiong, Jiaxin Ge, Renrui Zhang, Kaichen Zhou, and Shanghang Zhang. A self-correcting vision-language-action model for fast and slow system manipulation, 2025. URL https://arxiv.org/abs/2405.17418.

Zijun Lin, Jiafei Duan, Haoquan Fang, Dieter Fox, Ranjay Krishna, Cheston Tan, and Bihan Wen. Failsafe: Reasoning and recovery from failures in vision-language-action models, 2026. URL https://arxiv.org/abs/2510.01642.

Chenyv Liu, Wentao Tan, Lei Zhu, Fengling Li, Jingjing Li, Guoli Yang, and Heng Tao Shen. Self-correcting vla: Online action refinement via sparse world imagination, 2026. URL https://arxiv.org/abs/2602.21633.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1b: a diffusion foundation model for bimanual manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=yAzN4tz7oI.

Chenyang Ma, Guangyu Yang, Kai Lu, Shitong Xu, Bill Byrne, Niki Trigoni, and Andrew Markham. Cyclevla: Proactive self-correcting vision-language-action models via subtask backtracking and minimum bayes risk decoding, 2026. URL https://arxiv.org/abs/2601.02295.

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. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, May 2024. doi: 10.1109/icra57147.2024.10611477. URL http://dx.doi.org/10.1109/ICRA57147.2024.10611477.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Jiayuan Gu, Zhigang Wang, Yan Ding, Bin Zhao, Dong Wang, and Xuelong Li. SpatialVLA: Exploring Spatial Representations for Visual-Language-Action Models. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.011.

Wenke Xia, Ruoxuan Feng, Dong Wang, and Di Hu. Phoenix: A motion-based self-reflection framework for fine-grained robotic action correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6981–6990, June 2025.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi "Jim" Fan, Guanya Shi, and Yuke Zhu. Self-improving vision-language-action models with data generation via residual rl, 2025. URL https://arxiv.org/abs/2511.00091.

Yifan Yang, Zhixiang Duan, Tianshi Xie, Fuyu Cao, Pinxi Shen, Peili Song, Piaopiao Jin, Guokang Sun, Shaoqing Xu, Yangwei You, and Jingtai Liu. Fpc-vla: A vision-language-action framework with a supervisor for failure prediction and correction, 2025. URL https://arxiv.org/abs/2509.04018.

Zewei Ye, Weifeng Lu, Minghao Ye, Tao Lin, Shuo Yang, Junchi Yan, and Bo Zhao. Robofac: A comprehensive framework for robotic failure analysis and correction, 2026. URL https://arxiv.org/abs/2505.12224.

Xianchao Zeng, Xinyu Zhou, Youcheng Li, Jiayou Shi, Tianle Li, Liangming Chen, Lei Ren, and Yong-Lu Li. Diagnose, correct, and learn from manipulation failures via visual symbols. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 42386–42395, June 2026.

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 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1702–1713, June 2025.

Meng Zheng, Samhita Marri, Anwesa Choudhuri, Benjamin Planche, Zhongpai Gao, Van Nguyen Nguyen, Terrence Chen, Girish Chowdhary, and Ziyan Wu. Failing forward: Adaptive failure-informed learning for vision-language-action models, 2026. URL https://arxiv.org/abs/2605.08434.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=b1CVu9l5GO.

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 Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A Appendix

A.1 Details of Paired Dataset Construction

Construction overview. We construct paired successful and failed trajectories from the original success-only LIBERO demonstrations. Given a successful trajectory $\tau^s = \{(o_t^s, s_t^s, a_t^s)\}_{t=1}^T$, we generate failed branches by perturbing the action sequence around manipulation-critical events. Each pair is represented as

$$\mathcal{P}_j = \left(\tau^s, \tau_j^f, e_j, m_j, \ell_j\right), \qquad e_j = \left(t_{\text{start}}^j, t_{\text{pre}}^j, t_{\text{end}}^j\right), \tag{7}$$

where m_j and ℓ_j denote the failure mode and severity, respectively. The two branches share the instruction, task identity, initial simulator state, and unperturbed execution prefix. Therefore, their primary difference is the execution outcome around the selected event, reducing unrelated variation when learning trajectory-level deviations.

Successful-trajectory replay. Each source demonstration is replayed in its corresponding LIBERO environment from the recorded initial simulator state. We first execute ten no-op control steps to stabilize the simulator and controller, and then replay the demonstrated actions. We record the realized actions, simulator and robot states, gripper and joint states, end-effector pose, task-completion signal, and third-person and wrist-camera observations when available. RGB observations are resized to 256×256. Only demonstrations whose replay satisfies the environment-defined task condition are retained as successful references. Successful and failed branches are generated through the same simulator, controller, and observation pipeline.

For long-horizon LIBERO-10 trajectories, small numerical errors can accumulate before a late-stage event. During data generation, we synchronize the simulator with the recorded source states along the unperturbed prefix. Synchronization is disabled at the intervention point, after which the perturbed branch evolves freely under the environment dynamics. This preserves previously completed subtasks without directly specifying the failure outcome.

Event anchoring. We localize manipulation events using gripper commands, physical gripper states, end-effector motion, and task-completion signals. A grasp or release is detected from open–close or close–open command transitions. For multi-stage trajectories, transitions within a 15-step neighborhood are debounced. When physical gripper observations are available, stable closure is defined as the first frame whose gripper position is below 0.005 and changes by less than 4×10^{-4} over each of the following two frames.

For interactions without a unique gripper transition, including drawer opening, plate pushing, and stove activation, event windows are derived from peaks in the six-dimensional end-effector action together with the first task-completion frame. We anchor object grasp and placement events in LIBERO-Spatial and LIBERO-Object; grasp, placement, drawer, plate, and switch interactions in LIBERO-Goal; and first/second grasps, intermediate/final placements, insertion, switch interaction, and release events in LIBERO-10.

The interval $[t_{\mathrm{start}}, t_{\mathrm{pre}})$ contains clean pre-event context. Perturbations begin at t_{pre}, while t_{end} includes the interaction and its immediate consequence. Invalid, duplicated, or degenerate windows are discarded. Importantly, t_{pre} here is the source-demonstration alignment anchor $t_{\mathrm{pre}}^{\mathrm{data}}$, determined before generating a failed rollout. It is distinct from the online detector anchor $t_p \equiv t_{\mathrm{pre}}$ and the rollback target t_{rb}.

Temporally correlated perturbations. The primary perturbation is Ornstein–Uhlenbeck (OU) noise, sampled independently for translational and rotational action components:

$$\begin{aligned} \epsilon_{t+1}^{d} &= \epsilon_t^d + \theta(\mu - \epsilon_t^d)\Delta t + \sigma_d^{(\ell)}\sqrt{\Delta t}\,\xi_t^d, \quad \xi_t^d \sim \mathcal{N}(0, I), \\ \widetilde{a}_t &= \begin{cases} a_t + [\epsilon_t^{\mathrm{trans}}, \epsilon_t^{\mathrm{rot}}, 0], & t_{\mathrm{pre}} \le t \le t_{\mathrm{end}}, \\ a_t, & \text{otherwise}, \end{cases} \end{aligned} \tag{8}$$

where $\epsilon_0^d = 0$, $\theta = 0.15$, $\mu = 0$, and $\Delta t = 1$. The basic perturbation modifies the three translational and three rotational dimensions while preserving the gripper command. Compared with frame-independent noise, OU noise produces smooth, temporally accumulated deviations such as approach drift, pose misalignment, and unstable contact.

We use three severity levels. The translational/rotational diffusion scales are $(0.08, 0.16, 0.24)/(0.02, 0.04, 0.06)$ for LIBERO-Spatial, $(0.08, 0.12, 0.16)/(0.02, 0.03, 0.04)$ for LIBERO-Object, $(0.08, 0.16, 0.24)/(0.02, 0.04, 0.06)$ for grasp and placement events in LIBERO-Goal, $(0.08, 0.13, 0.18)/(0.03, 0.045, 0.06)$ for its other interactions, and $(0.20, 0.22, 0.24)/(0.04, 0.05, 0.06)$ for LIBERO-10. These values are expressed in the normalized controller action coordinates.

Task-conditioned failure modes. Because different interactions fail in different ways, OU noise is combined with localized task-conditioned action modifications when needed. The constructed modes include lateral, vertical, or rotational approach errors for missed grasps; accumulated pose offsets for inaccurate or tilted placement; contact offsets for handles, switches, and articulated objects; attenuation or reversal of the demonstrated pulling, pushing, or turning direction; and modified release or insertion behavior. For multi-stage tasks, the perturbation is applied only to the designated interaction, preserving earlier task progress. Directional modifications are estimated relative to the successful trajectory rather than using a globally fixed direction, and continuous actions are clipped to the valid controller range.

Execution-based verification and storage. Failure acceptance is determined by environment completion rather than perturbation magnitude. Every candidate is re-executed from the same source

initial state and is retained when it fails to satisfy the environment task-completion condition:

$$\mathrm{Accept}(\tau^f) = \mathbb{I}\left[\sum_{t=1}^{T} d_t^{\mathrm{env}} = 0\right], \tag{9}$$

where d_t^{env} is the environment completion signal. Perturbed executions that still succeed are discarded and resampled. We additionally verify that the successful branch completes the task, all trajectory fields have consistent lengths, anchor indices are valid and ordered, and the event, failure-mode, and severity metadata are complete.

Each successful demonstration is associated with 3–9 verified failed branches, with six branches on average. Each branch includes actions, observations, robot and simulator states, completion signals, and `keyframe_ids` $= [t_{\mathrm{start}}, t_{\mathrm{pre}}, t_{\mathrm{end}}]$. This organization preserves an explicit one-to-many correspondence and enables direct sampling of aligned success–failure pairs during CTE training.

A.2 Details of Bank Retrieval

Success Bank construction. The Success Bank is initialized using successful trajectories from the training split. Entries are organized by task instruction, such that retrieval is performed only among demonstrations of the same task. Each entry is represented as

$$\mathcal{B}_j = \left(k_j^{\mathrm{init}}, \left\{k_{j,r}^{\mathrm{pre}}, e_{j,r}\right\}_{r=1}^{R_j}, \tau_j^s, \eta_j\right), \tag{10}$$

where k_j^{init} is the initial robot state, $k_{j,r}^{\mathrm{pre}}$ is the state at the pre-event anchor of event r, $e_{j,r}$ contains the corresponding $(t_{\mathrm{start}}, t_{\mathrm{pre}}, t_{\mathrm{end}})$ indices, τ_j^s is the successful trajectory, and η_j stores its episode identifier and data location. Only trajectories that satisfy the environment task-completion condition are inserted into the bank. The initial bank contains training demonstrations and is updated online with verified successful evaluation trajectories within each seed/task run. Bank contents are not carried across seeds or tasks. For the shared-trigger audit, the bank is kept fixed throughout all 6,000 evaluation rollouts to ensure identical retrieval across conditions. The bank construction retains the task identity, initial state, and source-episode metadata required to resolve the complete successful trajectory during inference.

Two-stage state retrieval. Once a failure precursor is detected, retrieval is performed using the current task instruction, the episode initial state k_{init}^q, and the detected pre-failure state k_{pre}^q. We first compute the initial-state distance

$$d_{\mathrm{init}}(q, j) = \left\| \bar{k}_{\mathrm{init}}^q - \bar{k}_{\mathrm{init}}^j \right\|_2, \tag{11}$$

where $\bar{k}$ denotes a state represented in the same normalized robot-state coordinates used by the CTE. The five entries with the smallest d_{init} are retained. This coarse stage removes references with substantially different scene and robot configurations without requiring trajectory-level encoding of the full bank.

Within these candidates, we consider pre-event anchors compatible with the detected interaction stage and select

$$(j^*, r^*) = \arg \min_{\substack{j \in \mathrm{Top5}(d_{\mathrm{init}}) \\ r \in \mathcal{E}_j}} \left\| \bar{k}_{\mathrm{pre}}^q - \bar{k}_{\mathrm{pre}}^{j,r} \right\|_2, \tag{12}$$

where $\mathcal{E}_j$ contains the candidate's event anchors that match the detected event type or execution stage. Initial-state matching preserves global robot-configuration similarity within the task, whereas the second stage explicitly matches the local state from which corrective re-execution begins. This is preferable to selecting solely by the initial state, particularly when successful demonstrations with similar initial configurations reach an interaction through different poses or motion histories.

Reference materialization. After selecting (j^*, r^*), we load only the associated successful event segment

$$\tau^{s*} = \tau_{j^*}^s \left[t_{\mathrm{start}}^{j^*, r^*} : t_{\mathrm{end}}^{j^*, r^*}\right]. \tag{13}$$

The start, pre-event, and end observations and robot states, together with the full event-level action sequence, are used as the successful branch of the CTE. The failed branch is anchored to the corresponding detected event before the two branches are jointly encoded. Retrieval is deterministic for a fixed bank and query, and the bank is bypassed during normal execution. Following a verified successful completion, the resulting trajectory is added as a new bank entry during system-level evaluation. This online update is disabled only for the shared-trigger audit.

A.3 Rule-based Detector

Detector interface. We use a lightweight rule-based detector as an external trigger for the correction pipeline. The detector is evaluated after every executed action and uses only the observation and execution history available up to the current step. Its inputs include the executed actions, robot and end-effector states, gripper state, and task-relevant interaction signals, such as object contact, object motion, articulated-joint position, and task completion. In the LIBERO implementation, these task-state quantities are obtained from the simulator instrumentation. No future observations, successful reference trajectory, or CTE output is used for detection.

At step t, the detector either returns no event or produces a failure plan

$$\mathcal{D}_t = (m,\, t_{\mathrm{start}},\, t_{\mathrm{pre}},\, t_f,\, t_{\mathrm{rb}})\,, \tag{14}$$

where m is the detected failure type, t_f is the detection frame, and $t_{\mathrm{pre}} < t_f$ is the online pre-failure anchor (a previously visited recoverable frame). The variable t_{rb} denotes the rollback-controller target. In the reported protocol $t_{\mathrm{rb}} = t_{\mathrm{pre}}$, and t_p in the main text is a shorthand for this online anchor. These online variables are distinct from the dataset alignment anchor $t_{\mathrm{pre}}^{\mathrm{data}}$ defined in Appendix A.1. The interval $[t_{\mathrm{start}}, t_f]$ is subsequently used as the failed branch of the CTE.

Event-specific detection. The rules are organized around manipulation stages rather than absolute episode time. For grasping, a missed grasp is detected when a closing command has been issued but the gripper remains near its empty configuration for two consecutive steps. The empty-gripper threshold is 0.0015. Placement-related rules are activated only after a stable grasp has been observed. They identify precursors such as sustained contact with the boundary of the target region while the gripper continues descending, negligible object displacement under persistent contact, or outward retreat following an unsuccessful insertion.

For articulated and goal-conditioned interactions, the detector measures task progress over a short temporal window. Drawer failures include rebound after the closest handle approach and prolonged lack of drawer motion, plate-pushing failures include missing the expected contact or object displacement, and stove failures are detected when a sustained switch interaction does not produce task progress. LIBERO-10 uses an event-state detector that tracks the current operation—e.g., the first or second grasp, placement, insertion, or switch interaction—and returns both the failed stage and its ordinal position in the long-horizon instruction.

Temporal validation and pre-frame selection. Transient contacts or single-frame gripper fluctuations do not immediately trigger correction. Each rule requires either a sustained condition or a consistent progress trend over multiple frames. Once an event is triggered, repeated detections from the same interaction are suppressed until the detector observes a new manipulation attempt.

The pre-failure frame is selected causally from the stored execution history using event-dependent offsets. By default, missed-grasp detections use a 20-step pre-frame and a 30-step context window, while placement detections use a 35-step pre-frame and a 45-step context window. Articulated-object rules select the pre-frame relative to the closest-contact or minimum-progress point. All indices are clipped to the observed history, and the rollback target is always a state that was actually visited before detection. Thus, selecting t_{pre} does not require access to the eventual terminal failure.

Role in evaluation. The detector determines when correction is activated and which historical segment is supplied to the correction pipeline. Correction tokens and recovery actions are generated by the learned pathway. For each backbone, the Retry and ReflectVLA variants use the same detector, thresholds, rollback target, maximum environment-step budget, and correction budget. Retry re-executes the base policy from the selected pre-frame, whereas ReflectVLA additionally conditions re-execution on the retrieved success–failure trajectory pair. This controlled protocol isolates

the benefit of deviation-conditioned correction from that of early intervention and an additional execution attempt.

A.4 Rollback Process

After detecting an incipient failure, ReflectVLA returns the robot to the selected rollback target t_{rb} (equal to the online pre-failure anchor t_{pre} in the reported protocol) by tracking the robot end-effector states observed during execution in reverse temporal order. The simulator remains active and the controller uses closed-loop commands:

$$\mathcal{X}_{\mathrm{rb}} = \left[x_{t_f}, x_{t_f-1}, \ldots, x_{t_{\mathrm{rb}}}\right], \qquad x_t = (p_t, r_t), \tag{15}$$

where p_t and r_t denote the recorded end-effector position and axis–angle orientation. All targets correspond to states actually visited by the robot, so rollback follows a locally executed path within the workspace.

For each historical target x_k, we compute the position error $e_p = p_k - p_t$ and orientation error e_r between the current and target end-effector orientations. The closed-loop rollback command is

$$a_t^{\mathrm{rb}} = \left[\mathrm{clip}\Big(\frac{e_p}{0.05}, -1, 1\Big), \mathrm{clip}\Big(\frac{e_r}{0.5}, -1, 1\Big), g_m\right], \tag{16}$$

where g_m is an event-dependent gripper command. The gripper is kept open when returning from a failed grasp and closed when rolling back a placement-related failure, thereby avoiding an unintended grasp or release during the return motion. For other interaction types, the most recent executed gripper command is preserved.

Each intermediate target is tracked for at most 10 controller steps and is considered reached when the position and orientation errors are below $0.004\,\mathrm{m}$ and $0.03\,\mathrm{rad}$, respectively. The final pre-failure target is given up to 60 steps and must remain within both tolerances for two consecutive steps, providing a stable starting configuration for corrective re-execution. Position and rotation errors, the number of controller steps, and target-convergence status are logged for every rollback.

Because the simulator remains active throughout this procedure, object contacts and dynamics evolve normally. No privileged restoration of the scene state is performed. When the detector triggers before a substantial irreversible scene change, rollback operates at an early, recoverable stage. Delayed triggers can fall outside this operating condition, as illustrated in Fig. 8. After reaching t_{rb}, ReflectVLA activates the correction adapter and regenerates the affected interaction. The Retry baseline uses the identical detector, rollback targets, controller, and convergence criteria, but resumes with the unconditioned base policy.

A.5 More Experimental Results

A.5.1 Per-Subtask Results

Figure 5 presents three complete correction episodes from failure detection to task completion. Each row shows the original rollout, the detected incipient-failure point (*I.F.*), rollback to a previously visited recoverable state, and corrected re-execution. The execution preserves progress before the failed interaction and regenerates the affected manipulation stage from the recovered state.

Tables 5 and 6 report the success rate of every subtask, measured over three evaluation runs. ReflectVLA improves all 40 subtasks with the OpenVLA backbone, increasing the suite-level averages by 7.6, 5.1, 9.2, and 8.2 percentage points on LIBERO-Spatial, Object, Goal, and LIBERO-10, respectively. With the stronger OpenVLA-OFT backbone, ReflectVLA improves all four suite averages and 38 of the 40 subtasks, despite the base policy already achieving success rates above 94% on every suite. The largest OFT gain is obtained on LIBERO-10 ($+2.5$ points), indicating that correction remains useful when failures occur at later stages of long-horizon execution. The suite-level standard deviations remain comparable to the corresponding base policies.

A.5.2 Evaluation of Failure Detection and Correction

Table 7 evaluates the detector and the subsequent correction process. Detection accuracy and recall are defined as $(\mathrm{TP} + \mathrm{TN})/N$ and $\mathrm{TP}/(\mathrm{TP} + \mathrm{FN})$, respectively. Recovery rate is the fraction of correctly detected failures that are subsequently completed after ReflectVLA correction.

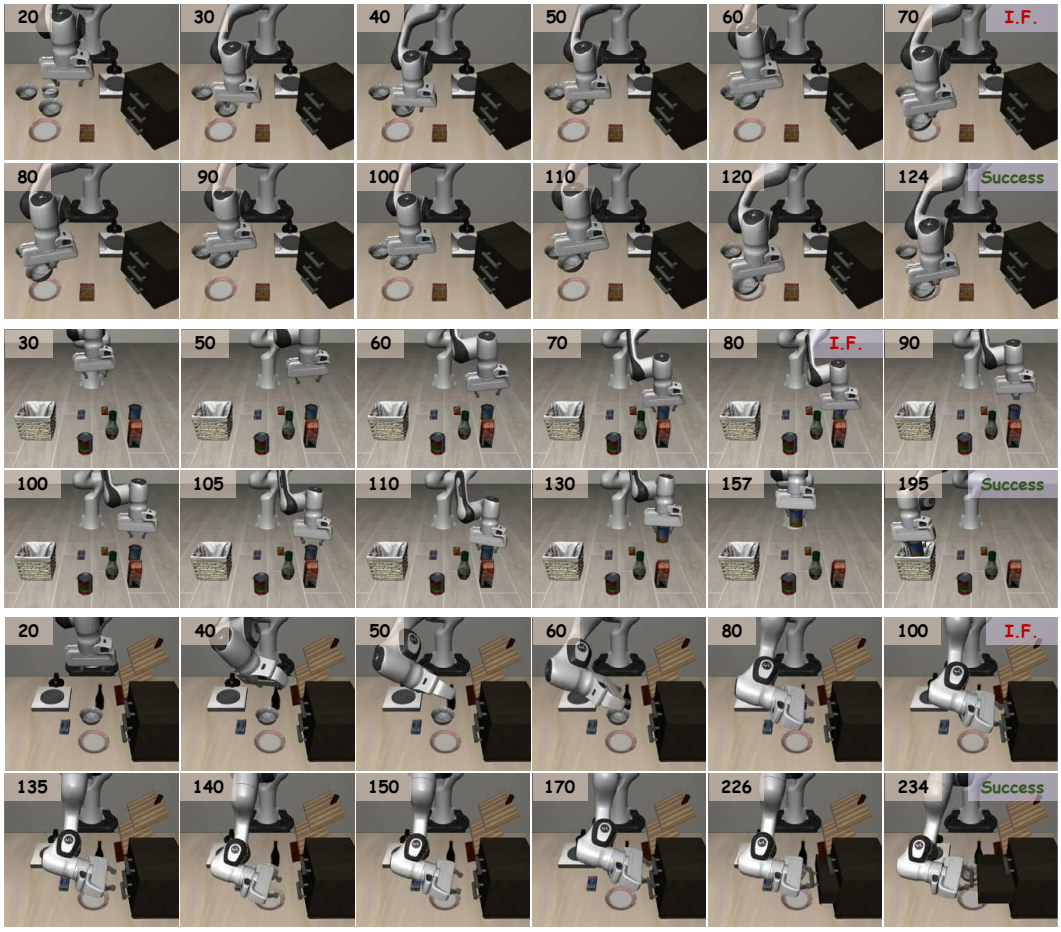


Figure 5: Execution rollouts of ReflectVLA on three different tasks. Every two rows show a complete correction trace, highlighting the detected incipient failure (I.F.), rollback to a recoverable frame, and successful corrected re-execution.

	LIBERO-Spatial		LIBERO-Object	
Task	OpenVLA	Reflect	OpenVLA	Reflect
T1	.953±.023	**.973±.012**	.907±.023	**.973±.012**
T2	.853±.023	**.953±.012**	.840±.040	**.913±.031**
T3	.887±.023	**.960±.020**	.940±.020	**.947±.012**
T4	.940±.020	**.973±.012**	.660±.040	**.813±.031**
T5	.800±.020	**.873±.012**	.913±.023	**.920±.020**
T6	.900±.020	**.953±.012**	.900±.020	**.973±.012**
T7	.860±.020	**.987±.012**	.953±.031	**.960±.020**
T8	.807±.023	**.927±.012**	.920±.040	**1.000±.000**
T9	.860±.040	**.933±.031**	.887±.023	**.927±.012**
T10	.640±.020	**.727±.012**	.940±.040	**.947±.031**
Avg.	.850±.008	**.926±.006**	.886±.010	**.937±.005**

	LIBERO-Goal		LIBERO-10	
Task	OpenVLA	Reflect	OpenVLA	Reflect
T1	.613±.023	**.760±.020**	.653±.012	**.700±.000**
T2	.860±.020	**.933±.012**	.673±.031	**.833±.023**
T3	.820±.040	**.893±.012**	.540±.040	**.733±.031**
T4	.460±.040	**.653±.031**	.447±.031	**.487±.023**
T5	.940±.040	**1.000±.000**	.500±.020	**.507±.012**
T6	.760±.020	**.813±.012**	.727±.031	**.920±.020**
T7	.667±.023	**.807±.012**	.467±.023	**.520±.020**
T8	.960±.020	**.987±.012**	.620±.020	**.627±.012**
T9	.833±.023	**.940±.020**	.160±.020	**.273±.012**
T10	.620±.040	**.667±.023**	.433±.012	**.440±.000**
Avg.	.753±.017	**.845±.009**	.522±.012	**.604±.011**

Table 5: Per-subtask success rates using the OpenVLA backbone. Each suite uses 500 trials per seed (50 per subtask), over three seeds. Values are mean ± sample standard deviation across seeds. "Reflect" denotes ReflectVLA. The better result within each task is shown in bold.

Across both backbones, the detector achieves macro-averaged accuracy of 97.50% and recall of 90.94%. Conditional recovery is 40.81% for OpenVLA and 53.33% for OpenVLA-OFT, yielding 47.07% when all eight backbone–suite settings receive equal weight. Each setting conditions recovery on true-positive triggers. Across all eight backbone–suite settings, 541 of 1,678 ground-truth

	LIBERO-Spatial		LIBERO-Object	
Task	OFT	Reflect	OFT	Reflect
T1	.993±.012	**1.000±.000**	.993±.012	**1.000±.000**
T2	.993±.012	**1.000±.000**	.980±.020	**.987±.012**
T3	.993±.012	**1.000±.000**	.960±.020	**.967±.012**
T4	.993±.012	**1.000±.000**	.960±.020	**.993±.012**
T5	.960±.020	**.967±.012**	.993±.012	**1.000±.000**
T6	.967±.023	**.973±.012**	.973±.012	**.980±.000**
T7	.993±.012	**1.000±.000**	.993±.012	**1.000±.000**
T8	.940±.020	**.987±.012**	.973±.012	**.980±.000**
T9	.993±.012	**1.000±.000**	.967±.031	**1.000±.000**
T10	.993±.012	**1.000±.000**	.980±.020	**.987±.012**
Avg.	.982±.009	**.993±.003**	.977±.004	**.989±.003**

	LIBERO-Goal		LIBERO-10	
Task	OFT	Reflect	OFT	Reflect
T1	.973±.012	**.980±.000**	.913±.012	**.920±.000**
T2	**.927±.012**	.920±.000	.953±.012	**.980±.000**
T3	.973±.023	**1.000±.000**	.973±.012	**.980±.000**
T4	.907±.031	**.973±.012**	.973±.012	**.980±.000**
T5	**.987±.012**	.980±.000	.927±.012	**.980±.000**
T6	.993±.012	**1.000±.000**	.993±.012	**1.000±.000**
T7	.993±.012	**1.000±.000**	.913±.023	**.920±.020**
T8	.993±.012	**1.000±.000**	.967±.012	**1.000±.000**
T9	.973±.012	**.980±.000**	.827±.012	**.920±.000**
T10	.973±.012	**.980±.000**	.973±.012	**.980±.000**
Avg.	.969±.003	**.981±.001**	.941±.002	**.966±.002**

Table 6: Per-subtask success rates using the OpenVLA-OFT backbone. Each suite uses 500 trials per seed (50 per subtask), over three seeds. Values are mean ± sample standard deviation across seeds. "OFT" denotes OpenVLA-OFT and "Reflect" denotes ReflectVLA. The better result within each task is shown in bold.

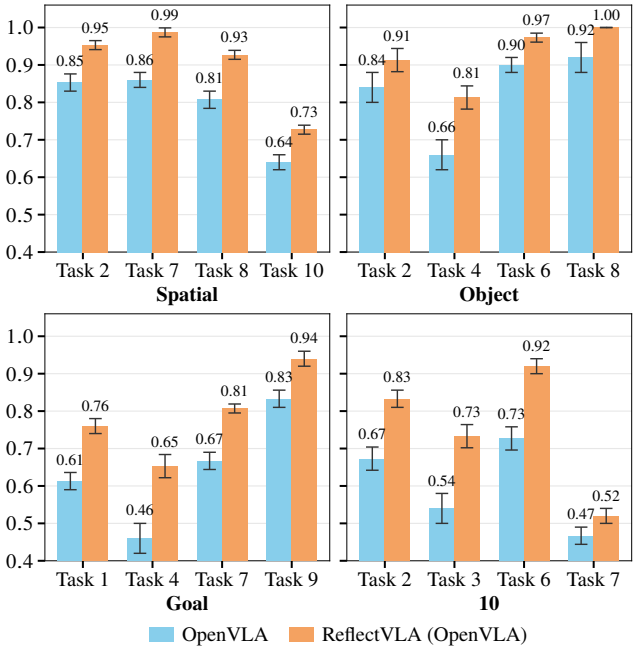


Figure 6: Four representative subtasks with different improvement magnitudes in each suite.

failures (TP+FN) are recovered, corresponding to an end-to-end recovery rate of 32.24%. Tables 2–3 use the same OpenVLA evaluation set with 1,500 rollouts per suite. Table 2 reports a micro-average over the 1,364 true-positive triggers, whereas the 40.81% OpenVLA recovery in Table 3 is the unweighted macro-average across the four suites.

A.6 Task-level and qualitative plots

Figures 6 and 7 provide task-level success-rate comparisons for the OpenVLA and OpenVLA-OFT backbones, respectively, complementing the aggregate results and shared-trigger mechanism analysis in the main text.

With OpenVLA, ReflectVLA improves Spatial Task 7 from 86% to 99% and Object Task 4 from 66% to 81%. The largest gains are 19 pp on Goal Task 4 (46% to 65%), LIBERO-10 Task 3 (54% to 73%), and LIBERO-10 Task 6 (73% to 92%). ReflectVLA also improves the OpenVLA-OFT

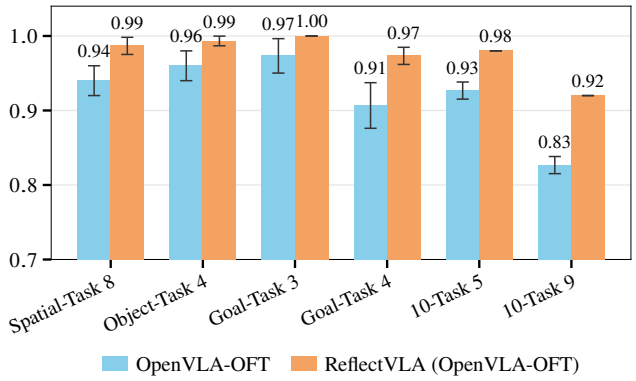


Figure 7: Success rates on six selected subtasks using OpenVLA-OFT.

Base Model	Benchmark	TP	FN	FP	TN	Accuracy (%)	Recall (%)	Recovery (%)
OpenVLA	LIBERO-Spatial	209	16	13	1262	98.07	92.89	54.55
	LIBERO-Object	156	15	14	1315	98.07	91.23	49.36
	LIBERO-Goal	339	31	12	1118	97.13	91.62	40.71
	LIBERO-10	660	57	10	773	95.53	92.05	18.64
	Average	–	–	–	–	97.20	91.95	40.81
OpenVLA-OFT	LIBERO-Spatial	25	2	23	1450	98.33	92.59	68.00
	LIBERO-Object	30	4	25	1441	98.07	88.24	53.33
	LIBERO-Goal	41	5	29	1425	97.73	89.13	43.90
	LIBERO-10	79	9	35	1377	97.07	89.77	48.10
	Average	–	–	–	–	97.80	90.35	53.33
Overall average		–	–	–	–	**97.50**	**91.15**	**47.07**

Table 7: Failure detection and correction results. Each benchmark row aggregates 1,500 evaluation rollouts. Recovery is measured among true-positive detections. Average rows are unweighted macro-averages across settings.

backbone with gains on all six highlighted tasks and a maximum improvement of 9 pp on LIBERO-10 Task 9 (83% to 92%).

A.6.1 Failure Cases

Figure 8 presents representative episodes that define the operating domain of the event-local recovery interface. The examples separate scene-state changes and task-grounding errors from trigger-timing errors.

Object slippage. The top-row examples show object slippage from a previously stable grasp or contact configuration. Slippage changes the object pose or contact mode while the rollback controller tracks the end-effector trajectory. The recovered robot configuration therefore no longer specifies the scene state required by the successful reference. Extending the interface to object-state estimation, grasp-stability monitoring, or re-grasp planning would cover this failure class.

Incorrect target-object interaction. The bottom-left example is a target-object grounding error. ReflectVLA conditions local re-execution within the retrieved task and event context. Correcting target identity requires a grounding-verification or task-level replanning module before event-local re-execution.

Irreversible scene change caused by missed or delayed detection. The bottom-right example is a delayed trigger after an irreversible scene change. It is a detector-timing case: earlier triggering would expose the episode to the rollback and paired-conditioning pathway. Learned temporal prediction and uncertainty-aware thresholding are natural extensions of the shared trigger interface.



Figure 8: Representative failure cases of ReflectVLA. The top row shows object-slippage failures, where returning to a previously visited pose leaves the changed object or contact state unresolved. The bottom-left example shows an incorrect target-object interaction caused by erroneous task grounding. These categories identify scene-state and task-grounding extensions for the event-local correction interface. The bottom-right example shows an irreversible scene change before an incipient failure is detected, corresponding to a missed or delayed trigger.

The evaluated operating domain is event-local execution deviation with preserved task semantics and a trigger before substantial irreversible state change. Scene-state estimation, target verification, and learned trigger timing extend this interface to additional failure classes.

A.7 Shared-trigger Audit Protocol

Evaluation set and detector accounting. The shared-trigger analysis uses the OpenVLA evaluation set underlying Table 7: three seeds and 500 rollouts per LIBERO suite, yielding 1,500 rollouts per suite and 6,000 overall. The first detector trigger is labeled from the uninterrupted base-policy continuation, independently of correction outcomes. The resulting $(\mathrm{TP}, \mathrm{FN}, \mathrm{FP}, \mathrm{TN})$ counts are (209,16,13,1262), (156,15,14,1315), (339,31,12,1118), and (660,57,10,773) for Spatial, Object, Goal, and LIBERO-10. Thus the shared-trigger population contains 1,364 true-positive detections. Precision, recall, and false-positive rate are computed as $\mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$, $\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$, and $\mathrm{FP}/(\mathrm{FP}+\mathrm{TN})$, respectively.

Shared-trigger comparison. For every true-positive trigger, all conditions receive the same observed failure prefix, rollback state, retrieved successful reference, and remaining action budget. The bank, trigger rule, and rollback procedure are held fixed. The intervention changes only the information pathway provided to the correction model. Conditional recovery is the fraction of these 1,364 triggers that satisfy the environment success condition after the assigned intervention. Table 2 reports the recovered counts and the micro-averaged pooled rate. The Full condition recovers 114, 77, 138, and 123 episodes across the four suites, totaling 452/1,364. Retry recovers 18, 6, 25, and 48, totaling 97/1,364. The suite-level Full recovery values agree with the OpenVLA rows in Table 7. Tts reported 40.81% is their unweighted macro-average, whereas 33.1% pools all triggers.

Relation to system-level evaluation. The shared-trigger comparison uses the same 1,500-rollout OpenVLA suite accounting as Table 1. The differences between Full and Retry recovered counts are 96, 71, 113, and 75 episodes, which correspond to 6.4, 4.7, 7.5, and 5.0 percentage points when divided by 1,500 suite rollouts. This links the conditional mechanism comparison to the suite-level success-rate gains while retaining the true-positive trigger population as the denominator for recovery.