REVIVE: INFERENCE-TIME RETURN WITH PHASE-AWARE CONTROLLER FOR VLA MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models have shown strong general-purpose robot control by leveraging large vision-language models. However, imitation-learned VLA policies remain brittle in multi-step manipulation: small execution errors can compound off the demonstrated trajectory, and the policy may silently advance to a later phase without completing the current subgoal, which we formalize as *silent phase failure*. Existing recovery approaches often rely on additional human interventions, reward supervision, or offline retraining. We propose REVIVE, an execution-time recovery framework that improves VLA success without retraining the underlying policy. Instead of learning explicit recovery from every failure, REVIVE intervenes before local deviations escalate. An external Return Controller monitors phase-local progress, boundary evidence, and action-effect signals to detect phase-level stalls; upon detection, it performs phase-local rollback to the latest valid phase anchor, clears the current action chunk, and triggers replanning. Failure-directed decoder adaptation generates alternative actions, while failure memory ranks candidates to avoid repeating past mistakes. Experiments on simulated manipulation benchmarks and two real-robot long-horizon tasks show that REVIVE improves task success, including under out-of-distribution task configurations and object changes.

1 INTRODUCTION

Vision-Language-Action (VLA) models have demonstrated strong general-purpose robot control performance by leveraging the semantic capabilities of large vision-language models (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024). Trained by imitation learning on large-scale robot datasets, VLA policies learn to map an instruction and current visual observations to actions that resemble successful demonstrations. However, these policies remain vulnerable to changes in observation and execution conditions (Fei et al., 2025). Once an error occurs at an intermediate step, the policy may encounter two distinct but related failure modes: *compounding errors* and *silent phase failure*. *Compounding errors* arise from a mismatch between the states visited by the learned policy and its training distribution: small execution errors can steer the policy into poorly covered states and induce further errors (Ross et al., 2011). After failures such as dropped or mispositioned objects, the policy may remain in the resulting off-trajectory state without knowing how to return to a valid execution path. We further identify *silent phase failure*, in which the policy proceeds to the next phase without recognizing that the current subgoal remains unfulfilled. For instance, the gripper may not secure the object even as the arm begins moving toward the placement region. In this case, phase progression becomes decoupled from subgoal completion, as the policy continues executing actions for a later phase even though the prerequisite for that phase has not been satisfied. Such premature transitions allow an undetected failure to propagate across subsequent phases.

Existing approaches commonly address these issues by augmenting policy training with recovery experience. DAgger (Ross et al., 2011) iteratively collects expert corrections on learner-induced states, while DART (Laskey et al., 2017) deliberately perturbs expert demonstrations to incorporate recovery behaviors. In robotic manipulation, intervention-based methods such as IWR (Mandlekar et al., 2020) further focus supervision on failure-prone bottleneck states. Reinforcement-learning-based approaches such as HIL-SERL (Luo et al., 2025) combine online robot interaction with human demonstrations and corrective interventions to learn robust manipulation policies. More recently, $\pi^*_{0.6}$ (Intelligence et al., 2025a) uses RECAP to improve VLA policies through autonomous experi-

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.

ence, expert corrections, and reward-guided policy updates. These approaches improve robustness by teaching the policy how to act beyond the nominal demonstration distribution. However, these gains typically require additional real-world data collection and policy retraining, together with human intervention or task-specific reward supervision.

Our intuition is that the VLA does not necessarily need to learn how to explicitly recover from every possible failure state. Instead, we seek to intervene before local execution errors accumulate into more severe deviations. When an early deviation is detected, approximately reversing recent robot motions can move the system back toward a previously valid execution state, preventing the current error from propagating into increasingly difficult-to-recover states. A global reset provides a simple alternative, but is often unnecessarily costly, as it discards progress already completed in earlier phases. We therefore advocate *phase-local rollback*: returning to the latest valid phase anchor before the current phase failed. Such rollback does not by itself correct the policy—querying the same VLA again may simply reproduce the same mistake. Rather, it restores the system to a state where the error remains easier to correct, creating an opportunity to generate an alternative action before deviations accumulate further.

Based on this intuition, we propose *REVIVE*, an execution-time recovery framework that improves task success without offline retraining of the VLA. The central idea is to learn a Return controller outside the VLA, and trigger rollback and replanning when a phase failure is detected. Specifically, REVIVE addresses three questions: 1) **When to return?** The Return controller is trained to monitor whether phase-local progress continues to improve, together with phase-boundary probabilities and global progress, to determine whether the current phase has stalled or failed. 2) **Where to return?** We return to the latest phase boundary rather than resetting to the episode beginning. 3) **How to replan after return?** We preserve failed action patterns to temporarily adapt the action decoder for generating alternatives, then use failure memory to guide candidate selection and reduce the risk of repeating the same mistake.

Around these three questions, we design a phase-aware execution-recovery framework. An external Return Controller continuously monitors phase progress and boundary evidence. Once it detects a phase-level stall, it triggers local rollback and clears the current action chunk. Failure-directed decoder adaptation then steers candidate generation away from previously failed actions, while failure memory ranks the resulting candidates by their similarity to stored failures and their density under the failure distribution. The framework requires neither architectural changes to the VLA nor offline retraining: decoder updates are temporary and reverted at the end of each episode.

Our contributions are summarized as follows:

- We identify and formalize *silent phase failure* as a characteristic failure mode in VLA-based manipulation, in which phase progression becomes decoupled from the completion of the current subgoal.
- We introduce *REVIVE*, a lightweight execution-time recovery framework that combines progress and boundary monitoring, phase-anchor management, temporary decoder adaptation, and failure-guided candidate selection, without requiring explicit phase labels or architectural changes to the VLA.
- We conduct extensive experiments on robot-manipulation benchmarks and real-world tasks, showing that *REVIVE* effectively recovers from silent phase failures and improves the robustness and success rates of existing VLA policies.

2 Related Work

2.1 Vision-Language-Action Models

VLA models are a major direction toward general-purpose robot foundation models. RT-2, Octo, and OpenVLA transfer vision-language knowledge to action generation, forming generalist policies (Brohan et al., 2023; Team et al., 2024; Kim et al., 2024); π_0, $\pi_{0.5}$, and GR00T N1 further develop generative action modeling and generalist control (Black et al., 2024; Intelligence et al., 2025b; Bjorck et al., 2025). Related action representations include action chunking, diffusion policies, and efficient action tokenization (Zhao et al., 2023; Chi et al., 2025; Pertsch et al., 2025). Despite architectural differences, imitation learning from successful demonstrations remains a com-

mon starting point. Such supervision teaches what to do along nominal executions, but does not necessarily teach how to recover after execution deviates.

2.2 Learning Robust Policies from Failures

One response is to expand the policy's training experience. DAgger collects expert labels on learner-visited states, while DART perturbs demonstrations to increase coverage of recovery behavior (Ross et al., 2011; Laskey et al., 2017). HG-DAgger and Intervention Weighted Regression (IWR) incorporate human interventions into imitation learning, with IWR upweighting intervention data (Kelly et al., 2019; Mandlekar et al., 2020). Reinforcement learning offers another route: SERL supports sample-efficient robot learning, and HIL-SERL combines online interaction with human guidance (Luo et al., 2024; 2025). More recently, $\pi^*_{0.6}$ improves VLA policies through experience and reward-guided updates, while RL Token investigates bootstrapping online RL with VLA models (Intelligence et al., 2025a; Xu et al., 2026). These approaches turn additional interaction or correction into persistent policy improvement. Depending on the method, this requires expert labeling, robot data collection, reward supervision, or further optimization. We instead focus on execution-time recovery without offline VLA retraining, retaining decoder changes only within the current episode.

2.3 Inference-Time Recovery and Replanning

Inference-time methods complement these approaches by revising decisions during deployment. Traditional model predictive control (Rawlings et al., 2017) and the cross-entropy method (Botev et al., 2013; Williams et al., 2017) require suitable dynamics predictions and trajectory objectives; they do not themselves specify when to abandon and roll back a failed phase. Adding inference modules around pretrained generative policies offers another route, exemplified by enhancement through predictive world modeling (Qi et al., 2026). Relatedly, test-time training and TENT adapt to distribution shifts through self-supervision and entropy minimization, respectively (Sun et al., 2020; Wang et al., 2020), but these general objectives do not target failed robot actions. Our focus is complementary: deciding where execution should return, not only what action to execute next.

3 Method

REVIVE is an inference-time recovery framework for VLA models that combines phase-aware monitoring, bounded local rollback, and failure-guided action refinement (Figure 1). We first introduce execution phases and monitoring signals, then describe when and where to return and how to replan after rollback.

3.1 Execution Phases and Monitoring Signals

Let an episode be a textual task g, a sequence of observations o_t and actions from a frozen VLA policy, which predicts an action chunk $\mathbf{a}_{t:t+H-1} \in \mathbb{R}^{H\times d}$ each time, where H is the policy horizon and d is the action dimension.

Phase segmentation. We represent a successful rollout as S operational phases separated by $S-1$ internal boundaries, where a phase is an execution segment with a locally coherent action pattern. We identify boundaries from stable gripper transitions and prominent change peaks in smoothed, normalized arm actions, with the latter capturing motion transitions without gripper opening or closing. Motion candidates too close to existing boundaries are excluded to avoid redundant cuts. These offline boundaries define training targets for phase-local progress and boundary prediction; segmentation details are given in Appendix A.1.

Silent phase failure. We define *silent phase failure* as an execution failure in which a phase subgoal remains unsatisfied, yet the policy proceeds to actions of a subsequent phase that depends on that subgoal, without initiating corrective behavior before the transition. For example, the arm begins transport toward the placement region despite failing to secure the object. For an attempt at phase s, let $C_s(t)$ indicate whether its subgoal is satisfied, $B_s(t)$ whether downstream behavior requiring that subgoal is executed, and $R_s(t)$ whether correction was initiated before the transition.

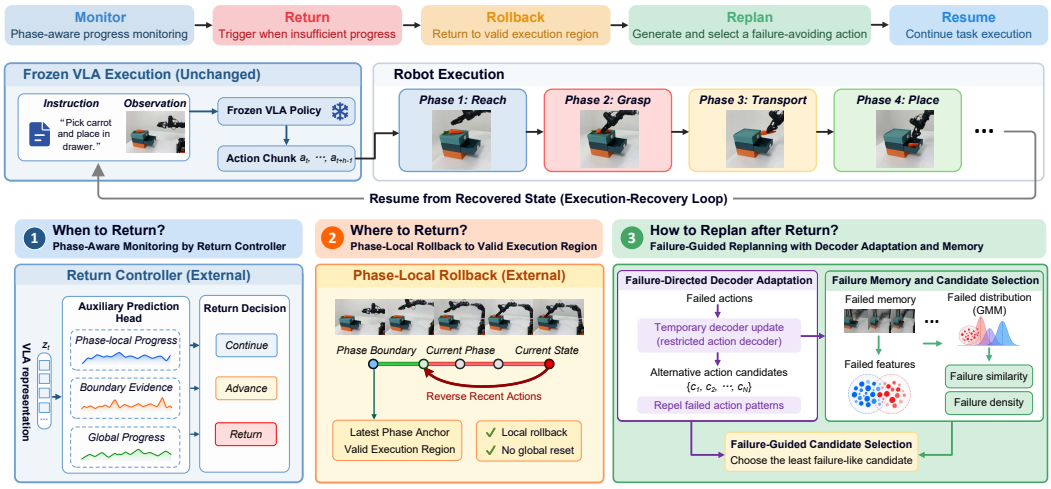

Figure 1: Overview of REVIVE's inference-time recovery framework for pretrained VLA policies. The Return Controller combines phase-local progress, boundary evidence, and global progress to decide among Continue, Advance, and Return. When Return is triggered, recent actions are reversed within the current phase. Failed actions then guide Decoder Adaptation (DA), which temporarily updates the action decoder to generate candidate chunks. These candidates are selected using the Failure-Guided Selection (FGS), which leverages failure memory.

Then, silent phase failure can be represented as:

$$\mathrm{SPF}_s(t) = \mathbf{1}[C_s(t) = 0 \ \wedge \ B_s(t) = 1 \ \wedge \ R_s(t) = 0]\,. \tag{1}$$

We report how often the pattern occurs in a frozen policy in Experiment 4.1.

Controller monitoring signals and maintaining buffer. We employ an external Return controller that executes each action chunk in a receding-horizon manner and monitors the resulting interaction. To decide between continuing, advancing, and returning, REVIVE predicts three quantities from a pooled VLA representation z_t together with proprioceptive features:

$$\hat{u}_t = f_u(z_t), \qquad \text{phase-local progress,} \tag{2}$$

$$\hat{h}_t = f_h(z_t), \qquad \text{boundary or phase-advance evidence,} \tag{3}$$

$$\hat{v}_t = f_v(z_t), \qquad \text{global episode progress.} \tag{4}$$

They resolve different ambiguities: $\hat{u}_t$ describes whether the current phase is still advancing, $\hat{h}_t$ whether the observation is near a phase boundary, and $\hat{v}_t$ provides an episode-level signal that complements a phase-local plateau. All three come from one lightweight auxiliary head, whose details are given in Appendix A.1. At inference, the head receives only the current VLA representation and proprioceptive state, and its outputs are consumed by the controller rather than fed back to the VLA. A stable gripper transition can additionally serve as a non-learned boundary cue. For the current phase s, the controller maintains a within-phase progress window W_t and an action buffer A_t recorded since the latest confirmed boundary, which are used to return to the last boundary.

3.2 When to Return

We measure progress change within the current phase rather than across a boundary. Splitting W_t into early and late halves, we define the predicted phase-local progress gain as:

$$g_t = \max_{i \in W_t^{\text{late}}} \hat{u}_i - \max_{i \in W_t^{\text{early}}} \hat{u}_i, \tag{5}$$

and treat the phase as stalled when $g_t < \delta_u$; the mean global progress over the late half provides a second failure signal when it falls below δ_v. The controller then applies three ordered rules. **Advance**: when boundary evidence together with high phase-local progress confirms completion,

or a stable gripper transition confirms a boundary, the controller moves the anchor forward and clears W_t and A_t without rolling back. **Return**: when no boundary is confirmed and either failure signal fires, it triggers Return once the cooldown and window conditions hold, and aborts if the per-phase return budget is exhausted. **Continue**: otherwise it executes the current plan. Confirmed boundaries are latched until their evidence decays, so a completed phase cannot repeatedly advance the controller.

3.3 Where to Return: Bounded Local Rollback

A global reset discards completed phases, so we use the phase action cache A_t recorded by the Controller to perform a partial rollback within the current phase, without requiring the simulator's internal state or human intervention. Let $n = |A_t|$ be the number of actions recorded since the latest confirmed boundary. When Return is triggered, we will reverse A_t to go back to the previous boundary. However, if n is large, reversing all A_t will cause the effective progress to be cleared. Therefore, we scale the rollback length with n and cap it by the maximum length K:

$$k = \min\left(\max\left(1, \lceil \rho n \rceil\right), K\right). \tag{6}$$

The last k actions are executed in reverse order: relative arm displacements are negated, while the gripper is commanded to the midpoint of its normalized range as a neutral action. Noticeably, the operation is an approximate rollback rather than a state restoration, since contacts, object motion, and grasp status may be irreversible.

3.4 How to Replan After Return

Rollback creates an opportunity for correction, but querying the same VLA from a similar observation may reproduce the same unsuccessful actions. REVIVE therefore reuses failed action sequences $\{\mathbf{A}_m^-\}_{m=1}^M$, where M is the number of previous attempts and failures in the current phase, to guide both candidate generation and candidate selection.

Decoder Adaptation (DA). To generate alternatives, DA temporarily adapts a restricted part of the action decoder by ascending the masked loss $\mathcal{L}_{\text{fail}}(\theta)$ on the failed sequence $\{\mathbf{A}_m^-\}_{m=1}^M$, with gradient clipping, weight shrinkage, and a trust-region projection:

$$\theta^+ = \Pi_{\|\theta-\theta_0\|_2 \le r}\left[\theta + \eta\ \text{clip}_c(\nabla_\theta \mathcal{L}_{\text{fail}}) - \eta\lambda(\theta - \theta_0)\right], \tag{7}$$

where θ_0 denotes the decoder parameters at episode start. The update repels the failed actions rather than imitating them; all other VLA parameters remain frozen, and the adapted policy generates candidate chunks $\{c_i\}_{i=1}^N$ from the post-rollback observation.

Failure-Guided Selection (FGS). To select among the candidates, FGS normalizes the failure actions and candidates to normalized action features, $\mathbf{f}_m^{\text{fail}} = \phi(\mathbf{A}_m^-)$ and $\mathbf{f}_i^{\text{cand}} = \phi(\mathbf{c}_i)$, and fits an episode-local spherical Gaussian mixture model (GMM) to the failure features only:

$$p_{\text{fail}}(\mathbf{u}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{u}; \boldsymbol{\mu}_k, \sigma^2 I). \tag{8}$$

For candidate i, $d_i = \log p_{\text{fail}}(\mathbf{f}_i^{\text{cand}})$ is its failure log-density, and the selected chunk trades that density against its similarity to the stored failures:

$$q_i = \exp\left(d_i - \max_j d_j\right), \tag{9}$$

$$s_i = \max_m \frac{1 + (\mathbf{f}_i^{\text{cand}})^\top \mathbf{f}_m^{\text{fail}}}{2}, \tag{10}$$

$$i^* = \arg\min_i \left(w_s s_i + w_u q_i\right). \tag{11}$$

The GMM models the density of observed failure patterns rather than a calibrated success probability. Both steps are episode-local: the original decoder parameters are restored and the failure memory cleared afterwards. Rollback followed by a direct query to the frozen VLA serves as the corresponding baseline, and Appendix A.3 details feature construction, fitting, and default weights.

Table 1: Failure mode on the grasp phase with frozen GR00T N1.7 on LIBERO-plus object-suite variants. *Variants* counts screened variants, *Grasp* counts those with a grasp attempt, and *Grasp Fail* counts those whose grasp subgoal was unmet. *Silent* counts failures that advanced downstream without correction, and *Stall* counts those that stack at grasp subgoal. Both rates use *Grasp Fail* as the denominator.

Category	Variants	Grasp	Grasp Fail	Silent	R_{silent}	Stall	R_{stall}
Camera Viewpoints	117	113	16	6	0.375	10	0.625
Robot Initial States	177	113	20	9	0.450	11	0.550
All hard ($d \geq 3$)	654	553	61	15	0.246	46	0.754

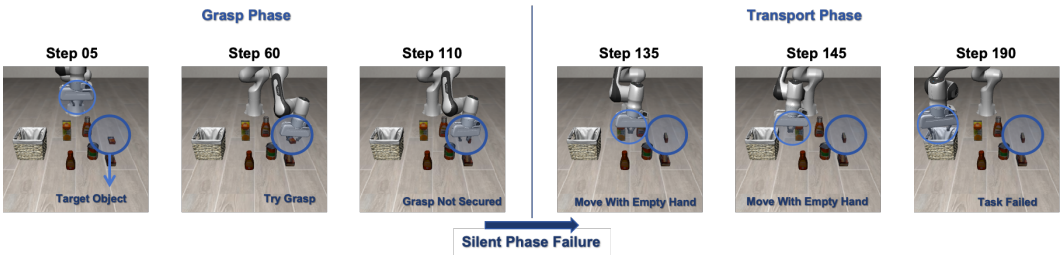

Figure 2: Frozen GR00T N1.7 on a silent-phase-failure episode: the grasp phase (steps 5–110) never secures the object, yet the transport phase (steps 135–190) proceeds with an empty hand and the task fails. The arrow marks the silent transition. Rendering details are given in Appendix B.3.

4 Experiments

In this section, we first report the diagnostics of silent phase failures (SPF), then present the results of our methods compared with others on two simulated benchmarks and real-robot evaluation, and perform component and recovery-scope ablations, including interaction costs.

4.1 Silent Phase Failure in a Frozen Policy

We evaluate the SPF definition on the unmodified LIBERO-plus (Fei et al., 2025) object suite with a frozen GR00T N1.7 (Bjorck et al., 2025); labels are offline simulator-state annotations, and the protocol is given in Appendix B.3.

Silent phase failure occurs without injection. As shown in Table 1, both rates use grasp failures as the denominator: R_{silent} measures silent downstream transitions, and R_{stall} measures stalls at the grasp subgoal. Without injected failures, the hard-band rates are 24.6% and 75.4%, respectively. Figure 2 shows a failed grasp followed by empty-handed transport; the panel contains the frozen policy only. *Distinct from a stalled task:* Silent transitions are concentrated in Camera Viewpoints and Robot Initial States, whereas Objects Layout mainly produces stalled phases. SPF therefore differs from failure that prevents further progress, although this screened diagnostic does not by itself exclude variability.

4.2 Experimental Setup

Benchmarks and policies. (1) Standard LIBERO (Liu et al., 2023) uses 10 tasks and 50 episodes per task in Spatial, Object, Goal, and Long (LIBERO-10); (2) LIBERO-Plus Fei et al. (2025) uses one episode per task variant across seven perturbation axes. We report success rates, *Avg* as the mean across suites or categories, *costs* as per-episode means, *Steps* as the total executed steps including rollback, *Ret.* as the times that Return triggers, and *Share(%)* as rollback-step fraction. Bold and underline mark the best and second-best values within each comparison group. We use the officially released checkpoints for π_0 Black et al. (2024), $\pi_{0.5}$ Intelligence et al. (2025b), and GR00T N1.7 Bjorck et al. (2025), without additional offline fine-tuning.

Table 2: LIBERO success rates (%) for the base policy and the base policy augmented with the method, across three VLA backbones. Intermediate configurations, including rollback with direct replanning, are analyzed in the ablation study. A plus prefix marks the baseline augmented with the method. † marks GR00T N1.7, which is evaluated with an officially released per-suite specialist checkpoint for each suite, whereas the π-family uses officially released single generalist checkpoints.

		Success rate (%) ↑				
Model	Method	Spatial	Object	Goal	Long	Avg
GR00T N1.7†	Baseline	**98.00**	97.20	95.80	87.80	94.70
	+REVIVE	97.80	**98.20**	**96.00**	**89.40**	**95.35**
π_0	Baseline	93.40	96.40	92.20	81.20	90.80
	+REVIVE	**99.40**	**100.00**	**98.60**	**93.60**	**97.90**
$\pi_{0.5}$	Baseline	99.20	99.20	97.20	92.80	97.10
	+REVIVE	**100.00**	**100.00**	**99.60**	**98.40**	**99.50**

Table 3: LIBERO-plus success rates (%) per perturbation category for the base policy and the base policy augmented with the method. A plus prefix marks the augmented configuration.

		Success rate (%) ↑							
Model	Method	Noise	Light	Layout	Camera	Background	Language	Robot	Avg
π_0	Baseline	5.25	64.54	47.80	3.63	39.41	37.61	3.81	28.86
	+REVIVE	**25.12**	**86.53**	**71.71**	**17.42**	**57.26**	**76.84**	**14.07**	**49.85**
$\pi_{0.5}$	Baseline	86.82	96.23	86.16	71.11	94.98	85.36	75.03	85.10
	+REVIVE	**92.44**	**99.91**	**90.36**	**81.30**	**99.07**	**90.24**	**84.00**	**91.05**

Compared methods. **Baseline** executes the base VLA without explicit recovery. **Return** performs local rollback and directly replans with the unchanged policy. **Return + FGS** adds Failure-Guided Selection, while **Return + DA** adds Decoder Adaptation; these are separate single-component variants. **REVIVE** combines DA and FGS. Neither single-component variant alone represents the complete method.

4.3 Main Results

We evaluate whether REVIVE improves task success across VLA backbones on LIBERO and whether its benefits extend to LIBERO-plus, which is used to evaluate the robustness of VLA.

LIBERO. Table 2 shows higher average success with REVIVE on all three backbones, with gains of 7.10, 2.40, and 0.65 percentage points for π_0, $\pi_{0.5}$, and GR00T N1.7. For each backbone, the largest absolute gain occurs in the Long suite, reaching 12.40, 5.60, and 1.60 points. This demonstrates that REVIVE greatly contributes to the closed-loop success in long-horizon tasks. However, GR00T declines by 0.20 points on Spatial and shows a modest overall scores.

LIBERO-plus. Table 3 shows substantial improvements across all seven perturbation categories for both models, π_0, $\pi_{0.5}$, across all seven perturbation categories. The average performance rises dramatically from 28.86% to 49.85% for π_0, and from 85.10% to 91.05% . Despite the notable gains, the hardest perturbations in LIBERO-plus,robot and camera,are still not fully resolved. π_0 in particular remains inadequate, achieving only 17.42% on Camera and 14.07% on Robot.

4.4 Ablation Study

Components, recovery scope, and interaction cost. Table 4 reports the component and recovery-scope ablations of π_0, with each suite in one column followed by per-episode interaction costs. Baseline and Return serve as references, Return + FGS and Return + DA add one component each, and REVIVE combines them. The **Return-to-Start (RTS)** variant, labeled +REVIVE (RTS), retains

Table 4: Component and recovery-scope ablations with π_0 on LIBERO. Each suite is one column, followed by per-episode cost columns. RTS costs are averaged over 100 episodes per suite (400 total), whereas the other methods use 500 episodes per suite (2,000 total).

	Success rate (%) ↑					Cost per episode		
Method	Spatial	Object	Goal	Long	Avg	Steps	Ret.	Share (%)
Baseline	93.40	96.40	92.20	81.20	90.80	173.93	0.00	0.00
Return	95.60	98.40	91.00	81.80	91.70	220.92	0.75	6.07
Return + FGS	99.40	**100.00**	98.40	92.00	97.45	223.23	0.77	6.20
Return + DA	99.60	**100.00**	**98.60**	92.80	97.75	219.80	0.75	6.07
+REVIVE (RTS)	**100.00**	95.00	91.00	32.00	79.50	310.01	1.28	9.86
+REVIVE	99.40	**100.00**	**98.60**	**93.60**	**97.90**	225.69	0.79	6.24

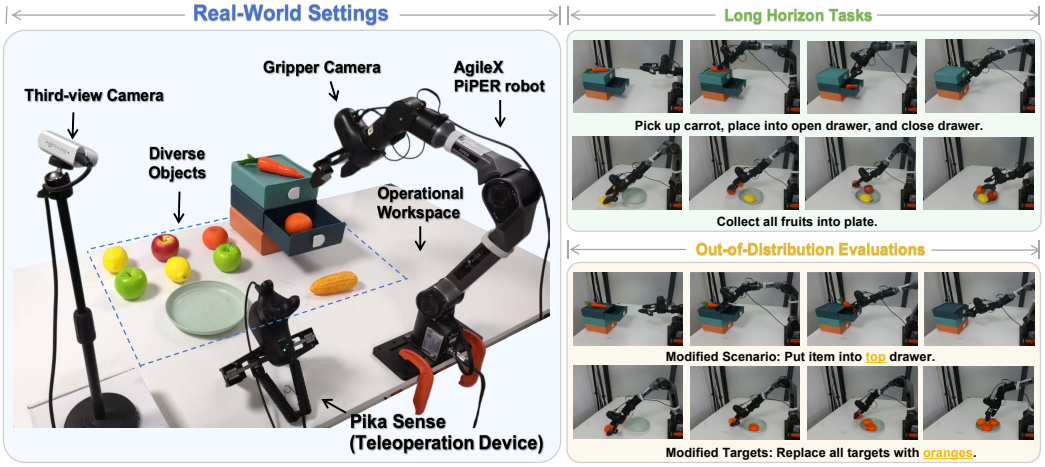

Figure 3: Real-world evaluation setup and task scenarios. Left: the robot manipulation platform consisting of an AgileX PiPER robot, multi-view cameras, and diverse objects in the workspace. Right: evaluated task scenarios, including long-horizon manipulation tasks and out-of-distribution variants with modified task configurations and object targets.

both DA and FGS but returns the arm directly to its episode-initial position and treats the entire executed action sequence as a failed record. The component decomposition for the other backbones and for LIBERO-plus is given in Appendix B.2.

The complete method reaches 97.90%, compared with 97.45% for Return + FGS and 97.75% for Return + DA. It uses 225.69 steps per episode versus 173.93 for the baseline, with rollback accounting for 6.24% of interactions; RTS uses a smaller evaluation set, as noted in the caption.

Phase-local versus return-to-start recovery. RTS retains DA and FGS but uses return-to-start rollback and full-trajectory failure records. Its Avg and Long success rates are 79.50% and 32.00%, versus 97.90% and 93.60% for REVIVE. Because both recovery scope and failure-record scope change, this comparison does not isolate either factor.

4.5 Real-Robot Evaluation

We evaluate $\pi_{0.5}$ on an AgileX PiPER. The Carrot task requires grasping a carrot from the top of a cabinet, placing it in an open drawer, and closing the drawer; the Fruit task requires placing a lemon, an apple, and an orange into a plate. Both Baseline and REVIVE use $N = 20$ trials per task and setting. The two out-of-distribution settings swap in a top drawer and replace all fruits with oranges; neither adds training.

Task	Method	Phase SR (%) S1	S2	S3	Cond. SR (%) $P(\text{S2} \mid \text{S1})$	$P(\text{S3} \mid \text{S1}, \text{S2})$	Task SR
Long-horizon tasks (in distribution)							
Carrot	Baseline	45.00	75.00	35.00	77.80 (7/9)	57.10 (4/7)	20.00
	+REVIVE	100.00	70.00	95.00	70.00 (14/20)	100.00 (14/14)	70.00
Fruit	Baseline	40.00	70.00	80.00	75.00 (6/8)	100.00 (6/6)	30.00
	+REVIVE	80.00	80.00	85.00	87.50 (14/16)	85.70 (12/14)	60.00
Out-of-Distribution Evaluations							
Carrot	Baseline	90.00	70.00	30.00	66.70 (12/18)	8.30 (1/12)	5.00
	+REVIVE	60.00	70.00	85.00	58.30 (7/12)	100.00 (7/7)	35.00
Fruit	Baseline	20.00	75.00	65.00	75.00 (3/4)	100.00 (3/3)	15.00
	+REVIVE	85.00	75.00	60.00	76.50 (13/17)	61.50 (8/13)	40.00

Table 5: Real-robot success rates (%) for Baseline and the complete REVIVE pipeline on in-distribution long-horizon tasks and OOD tasks. OOD variants retain the three-phase structure but use the top drawer for Carrot and all oranges for Fruit. Each method uses 20 trials per task and setting. Metric definitions are given in the text.

Phase SR is a phase's success rate on its own: phases skipped in a rollout are retested in isolation and counted as (main + retest)/N. Task SR is the fraction of trials that complete all three phases. Cond. SR is computed on the same rollouts and conditions each phase on its predecessors: $P(\text{S2} \mid \text{S1})$ is the share of trials with a successful first phase that also succeed at the second, and $P(\text{S3} \mid \text{S1}, \text{S2})$ is the share of trials with a successful first two phases that also succeed at the third. The counts behind each conditional rate are given in parentheses. REVIVE has no skipped phases, so its phase rates require no supplementary retests.

For Baseline, in-distribution Carrot (Table 5) is limited by its last phase: Task SR is 20% while the third phase alone succeeds in 35% of trials. The top-drawer shift makes grasping easier (phase S1: 45%→90%), but the chain collapses ($P(\text{S3} \mid \text{S1}, \text{S2})$: 57.1%→8.3%, Task SR: 20%→5%): the robot can still close a drawer, but rarely reaches one to close. Fruit Task SR falls from 30% in distribution to 15% under OOD, alongside a drop in S1 from 40% to 20%; $P(\text{S3} \mid \text{S1}, \text{S2})$ remains 100% in both small conditional subsets.

As shown in Table 5, REVIVE improves Task SR in all four settings: from 20% to 70% for Carrot and from 30% to 60% for Fruit in distribution, and from 5% to 35% and 15% to 40%, respectively, under OOD shifts. Phase-level gains are not uniform: on OOD Carrot, S1 falls from 90% to 60% despite higher task completion.

Phase success is reported unconditionally over all trials for each condition and task, whereas the conditional columns condition each phase on its predecessors, so a failure in one phase propagates to the phases that follow. All outcomes are judged from observable results in the recordings.

5 Conclusion

We present REVIVE, an inference-time recovery framework that requires neither offline retraining nor architectural changes to the VLA. REVIVE combines phase-aware monitoring, bounded local rollback, Decoder Adaptation (DA), and Failure-Guided Selection (FGS) to generate and select alternative actions during execution. On LIBERO, REVIVE improves the average success rate of all three VLA backbones. On LIBERO-plus, it improves performance for both evaluated models across all seven perturbation categories. In real-robot evaluation, it increases complete-task success from 20% to 70% for Carrot and from 30% to 60% for Fruit; under the corresponding OOD settings, success increases from 5% to 35% and from 15% to 40%, respectively. REVIVE implements bounded rollback by reversing recent actions, and its effectiveness depends on the monitoring signals and recorded failure actions; it therefore does not guarantee recovery from every failure state.

6 AI USE STATEMENT

Generative AI tools were used to assist with literature search, language editing, code development, and discussion of paper writing. All AI-assisted outputs, including technical claims, implementations, and manuscript text, were independently reviewed and verified by the authors. The authors take full responsibility for the content of this work.

REFERENCES

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.

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.

Zdravko I. Botev, Dirk P. Kroese, Reuven Y. Rubinstein, and Pierre L'Ecuyer. Chapter 3 - the cross-entropy method for optimization. In C.R. Rao and Venu Govindaraju (eds.), *Handbook of Statistics*, volume 31 of *Handbook of Statistics*, pp. 35–59. Elsevier, 2013. doi: https://doi.org/10.1016/B978-0-444-53859-8.00003-5. URL https://www.sciencedirect.com/science/article/pii/B9780444538598000035.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

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.

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

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: a VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025a.

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*, 2025b.

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

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.

Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on robot learning*, pp. 143–156. PMLR, 2017.

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.

Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. Serl: A software suite for sample-efficient robotic reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16961–16969. IEEE, 2024.

Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033, 2025.

Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. Human-in-the-loop imitation learning using remote teleoperation. *arXiv preprint arXiv:2012.06733*, 2020.

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.

Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. Inference-time enhancement of generative robot policies via predictive world modeling. *IEEE Robotics and Automation Letters*, 2026.

J.B. Rawlings, D.Q. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017. ISBN 9780975937730. URL `https://books.google.com.hk/books?id=MrJctAEACAAJ`.

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

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pp. 9229–9248. PMLR, 2020.

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

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 1714–1721. IEEE, 2017.

Charles Xu, Jost Tobias Springenberg, Michael Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. Rl token: Bootstrapping online rl with vision-language-action models. *arXiv preprint arXiv:2604.23073*, 2026.

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.

A Protocol Details

A.1 Auxiliary Prediction Head and Training Targets

The monitoring signals are predicted by a lightweight auxiliary head. It concatenates pooled VLM features with proprioceptive state features, applies LayerNorm and a shared Linear–GELU trunk, and then uses separate scalar output layers. The phase-local progress and boundary heads are trained with their corresponding offline phase labels, while the global-progress head is trained with a normalized episode-time target. For phase s spanning $[b_{s-1}, b_s)$ in a trajectory of length T, these targets are

$$\tilde{u}_t = \frac{t - b_{s-1}}{b_s - b_{s-1}}, \;\; b_{s-1} \le t < b_s, \qquad \tilde{v}_t = \frac{t}{T}, \;\; 0 \le t \le T, \tag{12}$$

where $\tilde{u}_t$ is clipped to $[0, 1]$. The phase-local label resets at every phase, whereas the global label varies monotonically over the episode. In data processing, t/T is implemented using normalized timestamps when available, with frame indices or row positions as fallbacks. None of these labels is provided to the VLA as an additional observation.

The phase-local progress head uses binary cross-entropy with logits by default, with Smooth-L1 available as an alternative. The boundary and global-progress heads use binary cross-entropy with logits. At inference, each logit is passed through a sigmoid to produce the normalized prediction consumed by the controller.

A.2 Controller Defaults

The controller uses a 16-step progress window, an 8-step cooldown, and at most two returns per phase. The default phase-local progress-gain threshold is 0.04, the global-progress threshold is 0.35, and either failure signal can trigger Return. Rollback uses a fraction of 0.5 and at most 20 steps; arm commands are negated and the gripper is commanded to the midpoint value 0.5 of its normalized range.

After each Return, the controller clears the execution buffer and resets the policy inference state, but retains the episode's failure memory. It then temporarily adapts the action decoder, generates candidate actions, and uses the GMM-based failure memory to select a chunk for execution. Rollback actions are included in the episode environment-step statistics.

A boundary can be confirmed by predicted phase-advance evidence together with phase-local progress, or by a stable gripper transition satisfying the dwell condition. Once confirmed, the progress history and action buffer are cleared and a new phase count begins.

The phase-aware monitor, bounded action-buffer rollback, and failure-conditioned replanning form a single execution-time control loop. Temporary decoder updates are restored at the end of the episode; failure-memory-guided selection does not itself update VLA parameters. Neither step adds phase labels to the VLA observations.

A.3 Failure-Conditioned Replanning

Failure records support two connected steps: adapting candidate generation and guiding candidate selection. Given a failed attempt, its action loss is first used to adapt the decoder temporarily, which then generates four candidate chunks from the post-rollback observation. Failed-action features accumulated within the episode form a failure memory. A spherical GMM models this memory and scores the candidates, selecting the one with the lowest failure density. Subsequent failures update both the temporary decoder and the failure memory.

Adapting candidate generation. Let θ_0 denote the action-decoder parameter rows associated with the current robot embodiment at the beginning of an episode, and let θ denote their temporary values. For a failed action sequence $\mathbf{a}^-$, we replay the failed observation and action labels and evaluate only the VLA's original masked action loss; all auxiliary progress losses are disabled for this update. The failed-action objective is

$$\mathcal{L}_{\text{fail}}(\theta) = \frac{\sum_j m_j \, \ell_j(\theta; \mathbf{a}^-)}{\max(1, \sum_j m_j)}, \tag{13}$$

where m_j denotes the valid-action mask.

To repel the failed action sequence, we ascend rather than minimize $\mathcal{L}_{\text{fail}}$. Define the clipped gradient as

$$\bar{g} = \text{clip}_c(\nabla_\theta \mathcal{L}_{\text{fail}}) = \nabla_\theta \mathcal{L}_{\text{fail}} \min\left(1, \frac{c}{\|\nabla_\theta \mathcal{L}_{\text{fail}}\|_2}\right). \tag{14}$$

We then apply shrinkage toward the episode-initial parameters and project the resulting displacement onto a ball of radius r:

$$\theta' = \Pi_{\|\theta-\theta_0\|_2 \le r}\left[\theta + \eta\bar{g} - \eta\lambda(\theta - \theta_0)\right]. \tag{15}$$

Only the embodiment-specific rows of the final action-decoder weight and bias are trainable; every other VLA parameter remains frozen. We use one projected update with $\eta = 0.01$, $\lambda = 0.1$, $c = 1.0$, and $r = 0.05$. The original rows are restored at the end of the episode.

Failure-memory-guided candidate selection. The failure memory contains action-space information only. For each failed trajectory, we take the first P steps of every common non-gripper action group and zero-pad sequences shorter than P. Each action group is flattened and independently ℓ_2-normalized; the groups are then concatenated and normalized again to obtain a unit-length failure feature $\mathbf{f}_m^{\text{fail}}$. Candidate chunks are mapped to candidate features $\mathbf{f}_i^{\text{cand}}$ by the same transformation. Thus, the GMM uses neither visual features nor task-specific environment state.

Given M failure features $\{\mathbf{f}_m^{\text{fail}}\}_{m=1}^M$, we fit a Gaussian mixture with a shared fixed spherical covariance. We use $\mathbf{u} \in \mathbb{R}^D$ for an arbitrary query point in the shared feature space; the GMM is fitted only to the failure features, while candidate features are used only to query the fitted failure density.

$$p_{\text{fail}}(\mathbf{u}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{u}; \boldsymbol{\mu}_k, \sigma^2 I), \qquad K = \min\left(K_{\max}, \max\left(1, \left\lfloor \frac{M}{M_{\min}} \right\rfloor\right)\right). \tag{16}$$

The component means are initialized by farthest-point selection. For each failure feature $\mathbf{f}_m^{\text{fail}}$, EM computes the responsibility of component k under the fixed spherical variance,

$$r_{mk} = \frac{\pi_k \exp\left(-\|\mathbf{f}_m^{\text{fail}} - \boldsymbol{\mu}_k\|_2^2/(2\sigma^2)\right)}{\sum_{\ell=1}^{K} \pi_\ell \exp(-\|\mathbf{f}_m^{\text{fail}} - \boldsymbol{\mu}_\ell\|_2^2/(2\sigma^2))}, \qquad N_k = \sum_{m=1}^{M} r_{mk}, \tag{17}$$

and updates

$$\boldsymbol{\mu}_k = \frac{\sum_m r_{mk} \mathbf{f}_m^{\text{fail}}}{\max(N_k, \epsilon)}, \qquad \pi_k = \frac{N_k + 1}{M + K}. \tag{18}$$

The variance σ^2 remains fixed throughout fitting.

For candidate i, the GMM-based selector computes the log density

$$d_i = \log p_{\text{fail}}(\mathbf{f}_i^{\text{cand}}) = \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{f}_i^{\text{cand}}; \boldsymbol{\mu}_k, \sigma^2 I), \qquad q_i = \exp\left(d_i - \max_j d_j\right). \tag{19}$$

Because the features are normalized, its maximum similarity to the stored failures is

$$s_i = \max_m \frac{1 + (\mathbf{f}_i^{\text{cand}})^\top \mathbf{f}_m^{\text{fail}}}{2}. \tag{20}$$

The final score is

$$J_i = w_s s_i + w_u q_i, \qquad i^* = \arg\min_i J_i. \tag{21}$$

The selected chunk is the candidate with the lowest combined failure-density and failure-similarity score. When $w_s = 0$, the rule reduces to selecting the candidate with the lowest GMM density. In either case, this is a relative ranking and does not guarantee that the next attempt will succeed.

Default configuration. We use an action-prefix length of $P = 8$, at most four candidates, $K_{\max} = 2$, $M_{\min} = 2$, fixed variance $\sigma^2 = 0.25$, and 16 EM updates. The default scoring weights are $w_s = 1$ and $w_u = 0.25$; for density-only GMM selection, we set $w_s = 0$. The GMM is cleared at the beginning of each episode and refitted after every failure. Decoder adaptation changes the candidate-generating distribution, while failure-memory-guided selection determines which candidate is executed; neither changes the persistent VLA parameters or its observation interface.

Table 6: Per-suite LIBERO interaction statistics for π_0, with 500 episodes per method and suite. SR is success rate (%); Returns counts interventions and Eval steps include the Return steps.

Suite	Method	SR	Eval steps	Returns	Return steps
Spatial	Return	95.6	61,181	59	1,014
Spatial	Return + FGS	95.2	62,006	64	1,079
Spatial	Return + DA	95.0	61,640	62	1,083
Object	Return	98.4	78,311	153	2,578
Object	Return + FGS	97.8	78,830	157	2,610
Object	Return + DA	98.6	77,600	152	2,499
Goal	Return	91.0	81,356	219	3,740
Goal	Return + FGS	89.6	84,451	238	4,268
Goal	Return + DA	91.2	82,041	221	3,931
Long	Return	81.8	220,998	1,074	19,475
Long	Return + FGS	81.8	221,164	1,082	19,715
Long	Return + DA	83.2	218,313	1,061	19,169

Table 7: Per-episode interaction cost on LIBERO with $\pi_{0.5}$, over 2,000 episodes per method. Steps includes the Return steps, and Return share is the fraction of steps spent on rollback.

Metric	Return	Return + FGS	Return + DA
Steps	185.80	186.21	187.39
Returns	0.53	0.54	0.54
Rollback	9.11	9.18	9.35
Return share (%)	4.90	4.93	4.99

A.4 Protocol for the π-Family Results

The watchdog controller used in the π-family results is distinct from the learned controller described above. Stable gripper transitions delimit operational segments. Recovery is triggered after 150 forward steps since episode initialization or the preceding Return, subject to a nonempty segment buffer and return-count and rollout limits. The most recent half of the segment is reversed, capped at 20 steps, with a neutral gripper command. Recovery neither resets the scene nor restores a simulator state.

In the π-family FGS variant, the failed segment tail updates an episode-local selector, which chooses among four candidates generated with different attention temperatures while keeping the VLA parameters frozen. The DA variant performs test-time adaptation within the policy using the most recently predicted action chunk and its conditioning observation, then replans from the post-rollback observation. These are separate variants; neither alone is a measurement of the complete REVIVE pipeline or a validation of the GR00T decoder-update configuration described above.

B More Experiments Details and Results

B.1 Per-Suite Interaction Statistics for π_0

Table 6 preserves the per-suite π_0 records underlying the aggregate interaction table. Each suite covers 500 episodes per method. Eval steps count all rollout actions, including Return steps, which count rollback actions only; initialization settling steps are excluded. These data are not from the GR00T N1.7 ablation.

For $\pi_{0.5}$, an episode spends about 186 steps, of which Return spends 9.11 on rollback (4.90%); the other variants stay between 4.93% and 4.99%.

Table 8: Component ablations on LIBERO for all three backbones (%). Long denotes LIBERO-10; Avg is the unweighted suite mean. Within each model and column, bold marks the best result and underline the second best, with ties marked alike.

Model	Method	Spatial	Object	Goal	Long	Avg
π_0	Baseline	93.40	96.40	<u>92.20</u>	81.20	90.80
	Return	<u>95.60</u>	98.40	91.00	81.80	91.70
	Return + FGS	95.20	97.80	89.60	81.80	91.10
	Return + DA	95.00	<u>98.60</u>	91.20	<u>83.20</u>	<u>92.00</u>
	+REVIVE	**99.40**	**100.00**	**98.60**	**93.60**	**97.90**
$\pi_{0.5}$	Baseline	<u>99.20</u>	<u>99.20</u>	97.20	92.80	97.10
	Return	98.20	98.80	97.60	93.80	97.10
	Return + FGS	97.80	99.00	<u>98.00</u>	<u>94.20</u>	<u>97.25</u>
	Return + DA	97.80	98.80	97.60	93.80	97.00
	+REVIVE	**100.00**	**100.00**	**99.60**	**98.40**	**99.50**
GR00T N1.7	Baseline	<u>98.00</u>	97.20	95.80	87.80	94.70
	Return	97.60	**98.20**	<u>96.00</u>	88.20	<u>95.00</u>
	Return + FGS	**98.40**	96.60	**96.60**	88.20	94.95
	Return + DA	97.40	<u>97.80</u>	94.40	**89.60**	94.80
	+REVIVE	97.80	**98.20**	<u>96.00</u>	<u>89.40</u>	**95.35**

B.2 Component Ablations on All Backbones

This appendix reports the remaining abalation results: the full component decomposition on LIBERO for all three backbones. The main text uses π_0 alone because the separation between single components and the complete method is widest there; the other backbones show the same pattern with smaller margins, and are retained here so that no measured result is lost.

B.3 Silent Phase Failure Diagnostic Protocol

The diagnostic injects no artificial perturbation; it uses the benchmark's own perturbation variants directly. The task set is the LIBERO-plus object suite (2,518 variants), of which we screen 724 by perturbation category and difficulty band (654 at difficulty ≥ 3 and 70 at difficulty ≤ 2). The policy is the released GR00T N1.7 LIBERO checkpoint, frozen and run in process, with a budget of 280 environment steps per episode, 10 settle steps after reset, and fixed random seeds. Each variant runs one episode from its first initial state.

All criteria are offline annotations of simulator state, produced by the same annotator for both the per-episode labels and the aggregate table. A *grasp attempt* is the end effector coming within 6 cm of the target object. The *grasp subgoal is met* when the object rises at least 3 cm above its settled height while staying within 12 cm of the end effector for at least 5 consecutive steps. For episodes in which the grasp subgoal is not met, we inspect the final end-effector–object distance. An episode is classified as *Stall* if the end effector remains within the object vicinity, and as *silent phase failure* if it has left that vicinity. We use the 6 cm grasp-attempt radius as the object-vicinity threshold.

Let d_T denote the final end-effector–object distance and let $r_{\text{near}} = 6\,\text{cm}$ denote the object-vicinity threshold. The two terminal categories are complementary:

$$R_{\text{silent}} = \frac{N_{\text{silent}}}{N_{\text{fail}}}, \qquad R_{\text{stall}} = \frac{N_{\text{stall}}}{N_{\text{fail}}}, \qquad R_{\text{silent}} + R_{\text{stall}} = 1, \tag{22}$$

where N_{silent} counts episodes with $d_T > r_{\text{near}}$ and N_{stall} counts episodes with $d_T \leq r_{\text{near}}$, among the N_{fail} grasp-failure episodes. If a category contains no grasp failure, these rates are undefined.

These rates diagnose terminal forms of grasp failure rather than standard benchmark metrics, online controller inputs, or calibrated success probabilities. The complementary split records whether the end effector remains near the object at episode termination; it does not by itself establish the causal source of failure.

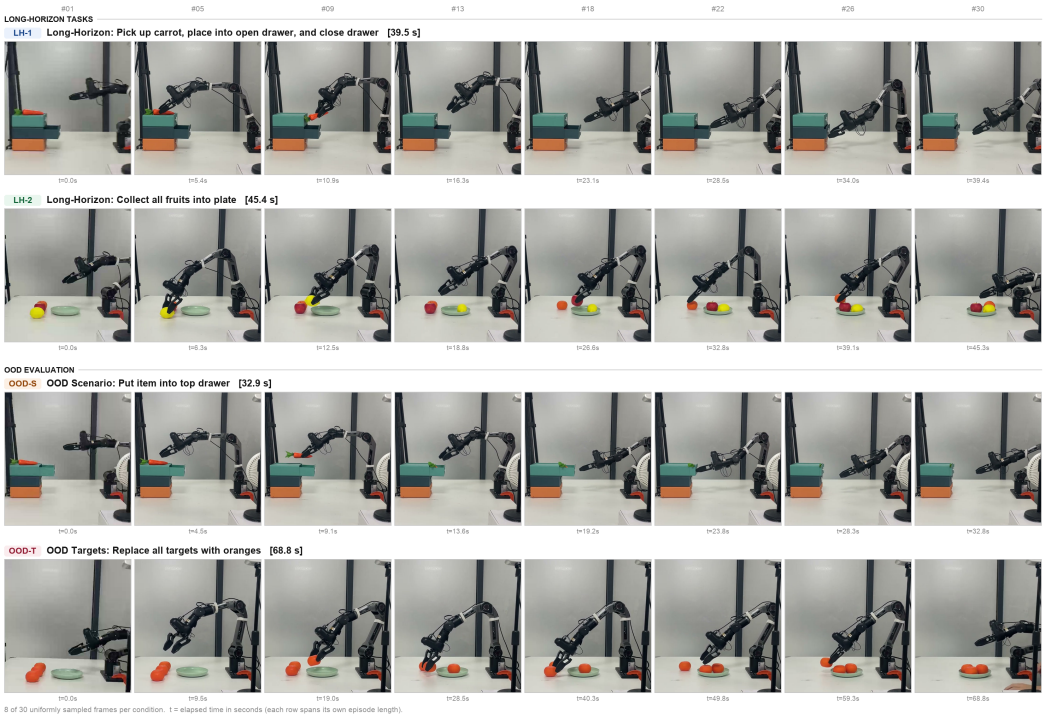

Figure 4: Visualized rollout trajectories for the real-robot evaluation. Rows correspond to different task settings: (1) LH-1: in-distribution long-horizon Carrot task (pick up carrot, place into open drawer, and close drawer); (2) LH-2: in-distribution long-horizon Fruit task (collect all fruits into plate); (3) OOD-S: out-of-distribution scenario with the top drawer; (4) OOD-T: out-of-distribution targets with all fruits replaced by oranges. Those external-camera images are for visualization only and are not fed to the policy.

The variant used for the example figure moves the rendered camera, so the panels are re-rendered from the benchmark's level-0 camera pose by replaying the logged actions; the replay reproduces the logged object poses to 0.006 m, and the panels show the frozen policy only, without a method comparison. A further limitation is that the criteria depend on thresholds (3 cm lift, 12 cm carry radius, 5 cm advance). Re-labelling with a 4 cm carry radius moves 4 of the 32 unreached-grasp labels, so the band-level conclusions are robust to this threshold while the exact counts are not.

C Real-World Demonstration Examples

To provide a visual understanding of the evaluation, Fig. 4 visualizes representative rollout trajectories for the four evaluation conditions. Each row shows 8 frames uniformly sampled from the full 30-frame rollout of a successful execution.

As shown in Fig. 4, the robot executes the complete multi-stage task from initial grasp to final placement/closure. For the in-distribution tasks, LH-1 demonstrates the sequential execution of grasping the carrot, placing it into the drawer, and closing it, while LH-2 shows the sequential collection of the lemon, apple, and orange into the plate. For the OOD settings, OOD-S illustrates the policy handling the modified spatial configuration with the top drawer, and OOD-T shows the execution under the target-replacement shift where all objects are oranges. These visual trajectories complement the quantitative success rates in Table 5 by showing how the policy transitions across phases in the real world.