RePO-VLA: Recovery-Driven Policy Optimization for Vision-Language-Action Models

Weijia Liufu[1,*] **Xiaoyu Guo**[2,3,*,†] **Ruiyi Chen**[1] **Jingzhi Liu**[1] **Kaidong Zhang**[1]
Xiwen Liang[1] **Jianqi Lin**[1] **Dawei Sun**[1] **Yuze Wang**[4] **Rongtao Xu**[5]
Bingqian Lin[1] **Bowen Yang**[6] **Tongtong Cao**[6] **Bowen Peng**[3] **Dongyu Zhang**[1]
Guangrun Wang[1] **Min Wang**[2] **Liang Lin**[1,3,✉] **Xiaodan Liang**[1,3,✉]
[1]Sun Yat-sen University [2]South China University of Technology
[3]Pengcheng Laboratory [4]Harbin Institute of Technology, Shenzhen
[5]Institute of Automation, Chinese Academy of Sciences [6]Huawei Noah's Ark Lab
[*]Equal contribution. [†]Project leader. [✉]Corresponding authors.

Abstract

Vision-Language-Action (VLA) models remain brittle in long-horizon, contact-rich manipulation because success-only imitation provides little supervision for execution drift, while failed rollouts are often discarded. We introduce RePO-VLA, a recovery-driven policy optimization framework that assigns distinct roles to success, recovery, and failure trajectories. RePO-VLA first applies Recovery-Aware Initialization (RAI), slicing each corrective suffix into a new episode so corrective actions depend on the current adverse state rather than the preceding failure. It then learns a Progress-Aware Semantic Value Function (PAS-VF), aligning spatiotemporal trajectory features with instructions and successful references. The resulting labels salvage useful failure prefixes via reliability decay, while low-value labels mark drift and terminal breakdowns, teaching differences among nominal, failed, and corrective actions. The data engine turns adverse states into planner-generated or human-collected corrective rollouts, teaching recovery to the success manifold. Value-Conditioned Refinement (VCR) trains the policy to prefer high-progress actions. At deployment, a fixed high-value token biases actions toward the learned success manifold without online failure detectors or heuristic retries. We introduce the Failure-Recovery Benchmark (FRBench), with standardized error injection and recovery-focused evaluation. Across simulated and real-world bimanual tasks, RePO-VLA improves robustness, raising adversarial success from 20% to 75% on average and up to 80% in scaled real-world trials.

1 Introduction

Vision-Language-Action (VLA) models have advanced general-purpose robot control [Kim et al., 2025, Black et al., 2025b], yet a persistent gap remains between nominal benchmark success and reliable long-horizon execution. The gap is especially pronounced in contact-rich bimanual manipulation, where small deviations in grasp pose, object contact, or inter-arm timing can quickly derail a task. These perturbations rarely make the task impossible; instead, they create recoverable adverse states requiring targeted correction before nominal progress can continue. Resilient execution therefore has a temporal structure: useful approach behavior, drift or stalled progress, and correction back to the success manifold.

Current supervision provides little signal for this behavior, despite growing attention to robotic failures [Liu et al., 2023, Ye et al., 2025, Haddadin et al., 2017]. Human-in-the-loop correction [Shi et al.,

✉Correspondence: linlng@mail.sysu.edu.cn, liangxd9@mail.sysu.edu.cn.

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

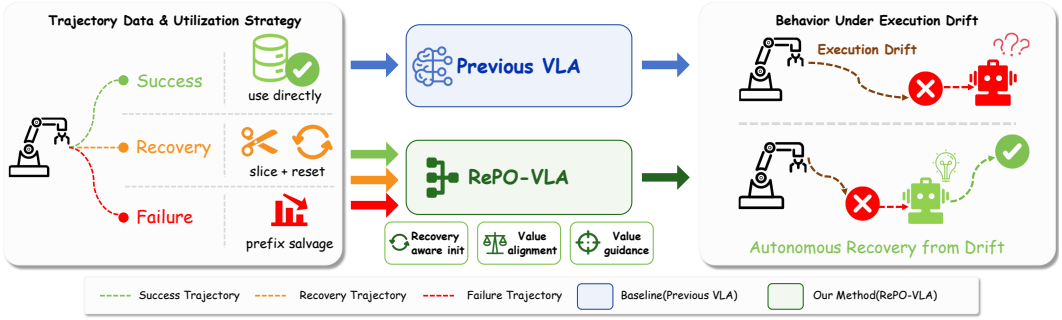


Figure 1: **Recovery-driven trajectory utilization.** Prior VLAs mainly imitate successful demonstrations and often fail under execution drift. RePO-VLA assigns distinct supervision to success, recovery, and failure trajectories, enabling autonomous correction from adverse states.

2024, Wu et al., 2025, Pramanick et al., 2022] and VLM/LLM replanning [Huang et al., 2022, Shinn et al., 2023, Guo et al., 2024] can recover from some high-level mistakes, but they are costly and often disconnected from low-level contact dynamics [Hafner et al., 2023, Ahn et al., 2022]. In practice, most VLA training still relies on supervised fine-tuning (SFT) over successful demonstrations. This creates a failure-utilization blind spot: pure failures are discarded even though their early portions often contain valid approach, contact preparation, or partial-manipulation priors. At the same time, their late portions indicate where behavior drifts away from the success manifold and should be marked as low-value rather than imitated. Recovery demonstrations pose a complementary ambiguity because they contain failed history, adverse states, and corrective actions. Naively imitating the whole sequence conflates the distribution that causes drift with the distribution that repairs it, leading to causal confusion and mode averaging. Sparse binary rewards compound the issue by failing to distinguish useful approach, temporary stagnation, active recovery, and terminal failure.

Figure 1 summarizes the core mismatch: success-only training leaves prior VLAs brittle under drift, while robust recovery requires assigning different roles to nominal-success, recovery, and failure trajectories. Failed rollouts should be decomposed rather than discarded, especially as VLA pipelines converge around large-scale imitation and flow-matching backbones. Portions close to nominal execution can be preserved, later portions show where the same trajectory becomes disadvantageous, and adverse states from injected or policy-induced failures can be turned into generated or collected corrective rollouts.

We propose RePO-VLA, a recovery-driven policy optimization framework that learns from nominal-success, failed, and recovered rollouts under a unified progress signal. It is a recovery-data and supervision framework used at fine-tuning time. It does not modify the pretrained VLA backbone, and it does not claim that recovery representations emerge during large-scale pretraining. A backbone that models failure, recovery, and progress directly is left as future work.

RePO-VLA first performs **Recovery-Aware Initialization (RAI)**. A raw recovery episode mixes the failure prefix that causes an adverse state with the later corrective suffix. Imitating that whole episode can teach the prefix as a trigger for correction. That causal confusion makes the original mistake look necessary, so later errors compound instead of being repaired from the current state. RAI slices out the corrective suffix and serializes it as a new episode, so the corrective skill is conditioned on the adverse state itself. It then learns a **Progress-Aware Semantic Value Function (PAS-VF)** by aligning spatiotemporal video features with language instructions. PAS-VF assigns dense labels to mixed-quality data, isolates error segments from correction segments, and preserves useful early portions of failed trajectories through reliability decay. These labels teach distributional differences among nominal progress, failed drift, and corrective recovery, enabling value-differentiated actions instead of one averaged imitation mode. Finally, **Value-Conditioned Refinement (VCR)** trains the policy to follow high-value behavior. During inference, RePO-VLA conditions on a fixed high-value token and uses the learned success manifold as an attractor, without online failure detectors or hand-coded retry rules.

To evaluate recovery as a first-class capability, we introduce the Failure-Recovery Benchmark (FRBench), with standardized error injection, recovery-focused protocols, and simulated and real-world bimanual tasks. FRBench separates nominal execution from recovery execution, making it possible to measure whether a policy can actively repair execution drift rather than merely succeed

under clean conditions. Its evaluation stages—Nominal, Error, and Recovery—condition recovery on a verified adverse state, preventing improvements in reaching or grasping from being mistaken for true correction. These evaluation stages are separate from the RAI and VCR training stages. Across FRBench-Sim and real-world adversarial perturbations, RePO-VLA improves recovery success over imitation baselines and shows an empirical recovery-data scaling trend as data density increases.

Our contributions are:

- **RePO-VLA**, an RAI and VCR supervision framework rather than a new VLA backbone. It eliminates recovery-data causal confusion and structurally reuses failed rollouts: progress alignment preserves useful failure prefixes, low-value labels expose drift-to-correction boundaries, and the data engine converts adverse failure states into generated or collected corrective rollouts for value-conditioned recovery.
- **FRBench**, a recovery benchmark with structured error injection and progress-sensitive evaluation in simulation and real-world bimanual settings.
- **Empirical validation** showing strong recovery gains over SFT baselines and improved recovery with increased recovery-data density.

2 Related Work

2.1 Vision-Language-Action Models

VLA models connect visual-semantic reasoning with robot control. OpenVLA [Kim et al., 2025] casts actions as language tokens, while recent π-series models [Black et al., 2025b,a] use continuous generative policies for dexterous trajectories. Long-horizon extensions add multimodal reasoning [Wen et al., 2025a], memory [Shi et al., 2026], or RL fine-tuning [Guo et al., 2025, Chen et al., 2025a, Intelligence, Physical et al., 2025, Zhu et al., 2026]. However, most pipelines remain success-demonstration centric, leaving policies poorly supervised for recovery from real perturbations.

2.2 Failure Recovery on Robotic Manipulation

Recovery methods include modular replanning, interactive correction, and learned refinement. VLM-guided systems such as ReplanVLM [Mei et al., 2024], REPLAN [Skreta et al., 2024], RACER [Dai et al., 2025], and RoboFAC [Ye et al., 2025] detect precondition errors [Raman et al., 2024] or generate recovery instructions, while interactive methods such as ThriftyDAgger [Hoque et al., 2021] and TRANSIC [Jiang et al., 2025] reduce distribution shift through intervention; both remain largely external to the learned low-level policy.

Learning-based recovery internalizes this capability. Failure and vulnerability predictors such as FIPER [Römer et al., 2025] and RoboMD [Sagar et al., 2024] estimate risk, while RLFT methods including RLPD [Ball et al., 2023], GRAPE [Zhang et al., 2024], and ConRFT [Chen et al., 2025c] align policies using outcome feedback. These signals are usually sparse or binary [Nakamoto et al., 2023, Mandlekar et al., 2020]. FailSafe [Lin et al., 2025] synthesizes recovery data but still uses SFT, which does not separate erroneous states from corrective actions. RePO-VLA instead uses dense progress labels to train value-conditioned recovery.

Our work is closest to methods that use imperfect or corrective data, but differs by structuring supervision rather than treating each episode uniformly: RePO-VLA assigns low value to error regions, high value to corrective regions, and reliability-decayed progress to useful failure prefixes. This separation makes recovery learning compatible with mixed-quality datasets where success, failure, and repair behaviors coexist; extended discussion is provided in Appendix A.3.

Prior robustness methods share the goal but not the supervision. FailSafe, the closest, synthesizes failure-recovery data for VLA training; RACER and RoboFAC focus on failure reasoning or re-planning; ThriftyDAgger on intervention timing; TRANSIC on a gated residual correction policy; and RLFT-style methods and $\pi^*_{0.6}$-RECAP on broader improvement from rewards or mixed-quality experience. RePO-VLA instead separates failure prefixes, error regions, and corrective suffixes within one low-level policy. We claim no superiority over methods without head-to-head comparison.

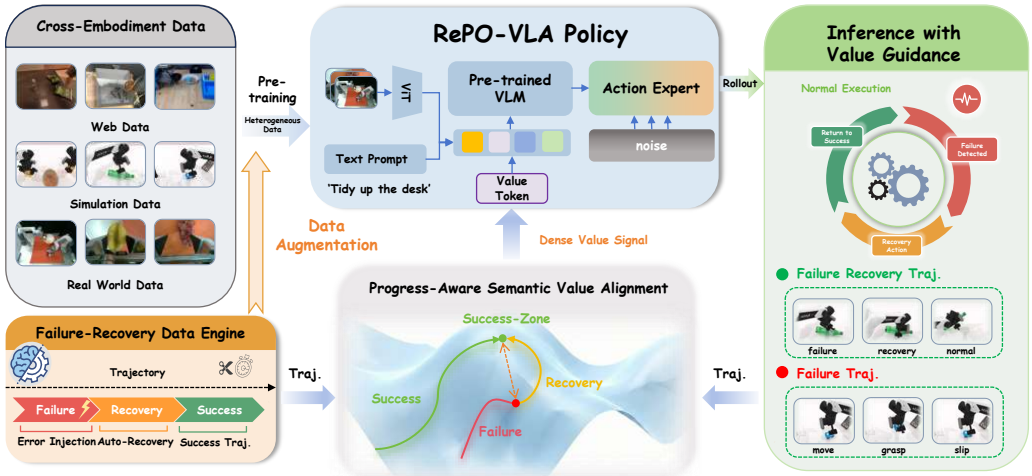


Figure 2: **Overview of RePO-VLA.** The framework builds recovery data, learns a progress-aware semantic value signal, and refines a value-conditioned VLA policy. At inference, constant high-value conditioning ($v = 1.0$) steers execution back toward the success manifold after drift.

3 RePO-VLA

RePO-VLA uses two training stages and does not modify the pretrained backbone. **Recovery-Aware Initialization (RAI)** fine-tunes on expert demonstrations and corrective suffixes serialized as new episodes, so a failure prefix is not part of the same episode as the correction. **Value-Conditioned Refinement (VCR)** then trains on the original mixed trajectories with dense progress labels for nominal-success, failure, and recovery behavior. The same policy handles nominal execution and recovery without switching modules at test time. In the experiments, Phase I names the RAI checkpoint and Full names the VCR checkpoint. Those names are training stages, distinct from the Nominal, Error, and Recovery evaluation stages.

3.1 Recovery-Aware Initialization (RAI)

RAI initializes the policy with expert demonstrations and recovery actions while avoiding causal confusion [de Haan et al., 2019]. A raw recovery trajectory contains two qualitatively different regions: the failure prefix that leads the system away from progress, and the corrective segment that brings it back. The base policy consumes a single-timestamp observation and does not take stacked historical image frames. If the failure prefix and the correction are still stored as one episode, the policy can treat the preceding mistake as the context that triggers correction; see Appendix A.2.1 for details.

We address this with Trajectory Slicing with Episode Reset (TSER). Given a recovery that starts at t_{rec}, we discard the failure prefix and extract $\tau' = \{(o_t, a_t)\}_{t=t_{rec}}^{T}$ as an independent correction episode whose time index restarts at $t = 0$. No historical image frames are attached before this onset. The resulting dataset is denoted $\mathcal{D}_{rec}^{reset}$. These samples expose the policy to adverse states such as an empty gripper, shifted object, or misaligned bimanual pose, but remove the failure prefix from the same episode. Thus RAI learns recovery as a state-conditioned skill rather than a replay of a particular failed rollout. TSER is dataset-level episode bookkeeping, not a reset of an observation-history buffer. Deployment uses the same single-timestamp observation, as clarified in Appendix A.2.2.

RAI mixes expert data $\mathcal{D}_{expert}$ and reset recovery data $\mathcal{D}_{rec}^{reset}$:

$$\mathcal{L}_{SFT} = -\mathbb{E}_{\tau\sim\mathcal{D}_{expert}} \sum_t \log \pi(a_t|o_t) - \lambda\, \mathbb{E}_{\tau'\sim\mathcal{D}_{rec}^{reset}} \sum_t \log \pi(a_t|o_t), \tag{1}$$

where each sum follows the valid horizon of its trajectory, and λ balances nominal imitation and recovery learning.

3.2 Progress-Encoded Semantic Value Learning (PAS-VF)

VCR requires dense labels that preserve useful pre-failure behavior without rewarding terminal breakdowns. A simple zero label for a failed trajectory is too coarse: many failures contain a correct approach, stable pre-contact motion, or partial manipulation before the final deviation. Conversely, assigning a high label to the whole trajectory would reward the terminal error. We therefore learn a self-referential Progress-Aware Semantic Value Function (PAS-VF), denoted $V(\tau)$, that estimates task progress as proximity to successful trajectories in a semantic latent space.

Spatiotemporal Alignment. We map trajectories and language instructions into a shared manifold $\mathcal{Z}$. This is important because bimanual tasks can be visually redundant: different arm assignments or grasp poses may represent the same semantic progress. A frozen V-JEPA spatiotemporal encoder E_v [Assran et al., 2025] extracts trajectory dynamics and a frozen CLIP text encoder E_t [Radford et al., 2021] encodes the instruction, while lightweight adapters produce $z^v = f_\theta(E_v(\tau))$ and $z^l = g_\phi(E_t(l))$. The base encoders remain frozen; only the adapters are optimized, keeping the value model lightweight and focused on progress alignment rather than representation relearning.

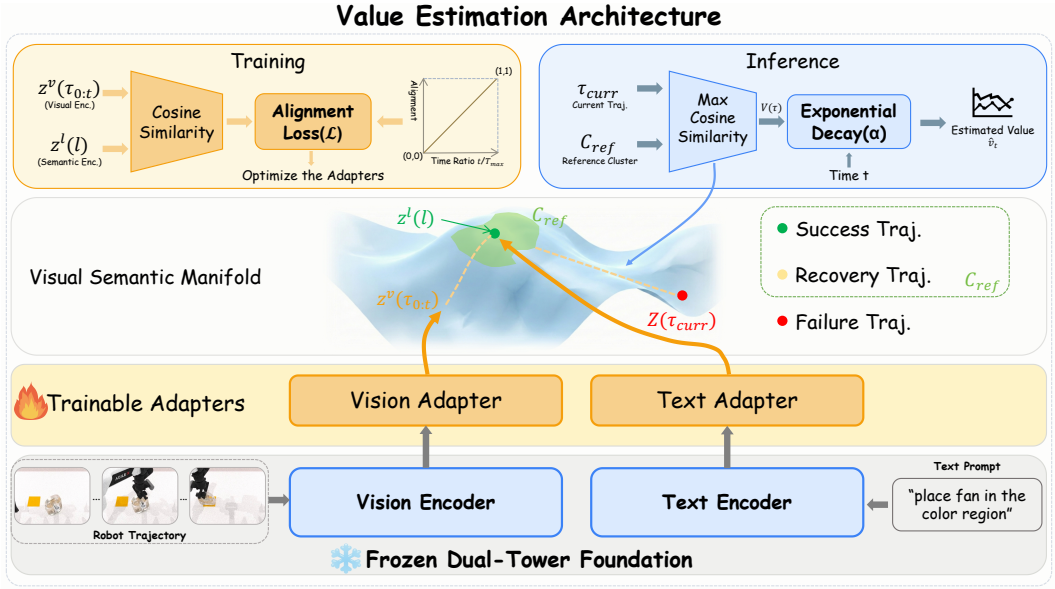


Figure 3: **Semantic value alignment.** Frozen visual and language encoders are projected into $\mathcal{Z}$; cosine similarity provides a dense progress signal.

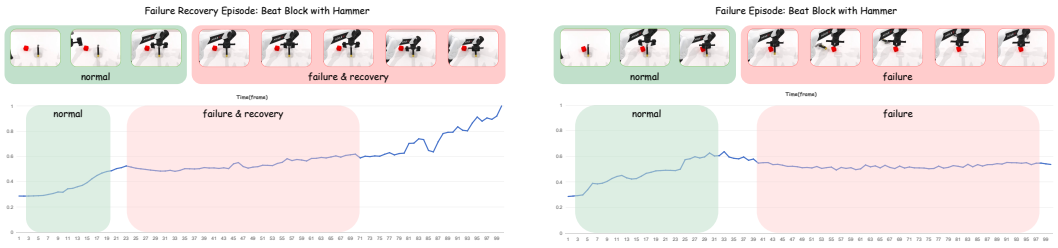
Training: Monotonic Progress Alignment. Adapters are trained on successful trajectories so cosine similarity tracks normalized temporal progress. For each prefix $\tau_{0:t}$, the target progress is t/T_τ, which encourages early prefixes to remain far from the instruction embedding and completed prefixes to align closely with it. With $z_t^v = f_\theta(E_v(\tau_{0:t}))$:

$$\mathcal{L}_{align} = \mathbb{E}_{\tau \in \mathcal{T}_{succ}} \left[\frac{1}{T_\tau} \sum_{t=1}^{T_\tau} \left(\text{CosSim}(z_t^v, z^l) - \frac{t}{T_\tau} \right)^2 \right]. \tag{2}$$

Self-Referential Progress Estimation. After alignment, we use successful executions as an internal reference set. For an unlabeled failure, progress is the nearest similarity to a reference cluster $\mathcal{C}_{ref}$ of successful embeddings:

$$V(\tau_{fail}) = \max_{z \in \mathcal{C}_{ref}} \text{CosSim}(z_{\tau_{fail}}^v, z). \tag{3}$$

Here $z_{\tau_{fail}}^v = f_\theta(E_v(\tau_{fail}))$. This self-referential score avoids manual progress annotation and remains task-relative: a trajectory is valuable when it approaches the successful manifold for its own instruction. Figure 4 visualizes the resulting progress, plateau, and failure regimes.



(a) Failure Recovery Episode value modeling

(b) Failure Episode value modeling

Figure 4: **Visualization of the Progress-Aware Value Landscape.**

3.3 Value-Conditioned Refinement (VCR)

VCR uses $V(\tau)$ to separate high-value correction from low-value failure behavior across the mixed dataset. Unlike RAI, which trains on sliced corrective suffixes in $\mathcal{D}_{rec}^{reset}$, VCR trains on the original mixed trajectories $\mathcal{D}_{rec}^{raw}$. Failure-inducing and corrective actions therefore remain in one episode and are distinguished by value labels rather than by an episode boundary. The observation at each step is still a single timestamp; additional episode-boundary details are in Appendix A.2.2.

3.3.1 Progress-Aware Hindsight Labeling

We construct a continuous value landscape over the raw mixed dataset by assigning each frame a dense label $v_t \in [0, 1]$. The labels are designed to separate behavior quality rather than trajectory identity:

- **Success and recovery:** successful trajectories τ_{succ} and effective recovery suffixes $\tau_{rec}^{+} \subset \mathcal{D}_{rec}^{raw}$ receive $v_t = 1.0$, since these frames contain actions that either maintain or restore progress.
- **Error segments:** deviating recovery prefixes $\tau_{rec}^{-} \subset \mathcal{D}_{rec}^{raw}$ receive $v_t = 0.0$. This explicitly tells the policy that the actions leading into the adverse state should not be averaged with the corrective actions that follow, even though both occur in the same raw episode.
- **Pure failures:** failed rollouts τ_{fail} keep early kinematic priors through reliability decay,

$$v_t = V(\tau_{fail}) \cdot \left(1 - \frac{t}{T}\right)^{\alpha}, \quad \forall t \in \tau_{fail}, \tag{4}$$

with $\alpha = 3.0$. This retains useful early motion while rapidly down-weighting states near irreversible failure; see Appendix A.2.4.

3.3.2 Goal-Conditioned Architecture and Inference

Value-Conditioned Training. We augment the flow-matching $\pi_{0.5}$ policy [Black et al., 2025a] with a value token $\mathbf{e}_{val} = \text{MLP}_{val}(v_t)$. The transformer attends to this token together with visual and language tokens, allowing the same observation to map to different action modes depending on desired progress. In this paper the normalized success anchor is the high-value token $v = 1.0$; that number is the top of our progress labels, not a task-independent physical scale. We fine-tune on $\mathcal{D}_{total} = \mathcal{D}_{succ} \cup \mathcal{D}_{rec}^{raw} \cup \mathcal{D}_{fail}$:

$$\mathcal{L}_{VCR} = -\mathbb{E}_{\tau \sim \mathcal{D}_{total}} \left[\sum_{t=1}^{T_\tau} \log \pi_\theta(a_t | o_t, l, \mathbf{e}_{val}(v_t)) \right]. \tag{5}$$

Goal-Conditioned Autonomous Recovery. During deployment, we do not run an online failure detector, estimate $V(\tau)$, or switch policies. The policy receives the current observation and a fixed high-value token, set to the normalized success anchor $v = 1.0$, at every step. Nominal and corrective actions come from this same interface. Appendix A.2.3 further discusses the detector-free inference setting.

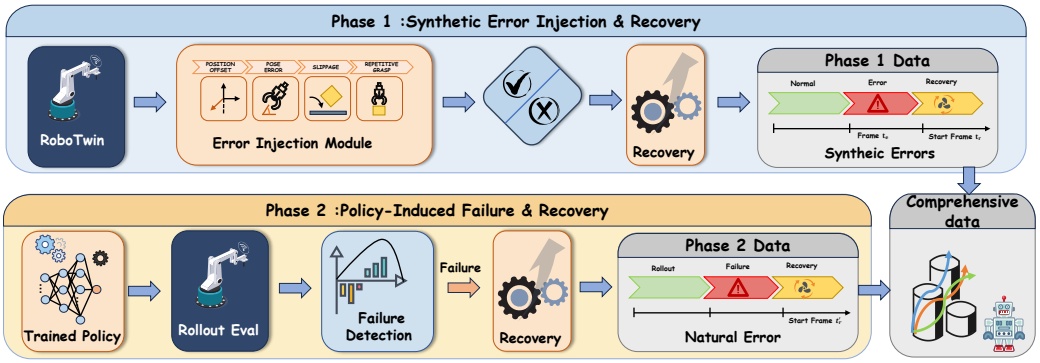


Figure 5: **Failure-Recovery Data Engine.** Recovery data combines control-intercepted error injection with policy-induced rollouts.

Table 1: **FRBench phase-based evaluation protocol.** The protocol isolates nominal task competence from corrective behavior after a controlled physical breakdown.

Phase	State	Control or intervention	Isolated measurement
Nominal	$\phi_t = \textsc{Nominal}$	Policy executes under clean or randomized conditions.	Baseline reaching, grasping, and task progress.
Error	$\phi_t = \textsc{Error}$	Structured E1–E4 perturbation projects the scene into an adverse state.	Reproducible physical breakdown independent of policy scoring.
Recovery	$\phi_t = \textsc{Recovery}$	The policy continues from the adverse state without external retry logic.	Corrective actions that restore task progress.

4 FRBench

FRBench evaluates whether bimanual policies can restore task progress after physical breakdowns. Unlike success-only suites, it separates nominal proficiency from corrective capability through controlled adverse-state projection. FRBench spans RoboTwin simulation [Chen et al., 2025b] and real-world tasks, with 23,453 simulated episodes over 46 tasks; detailed protocol and dataset statistics are provided in Appendix A.4.1 and Appendix A.4.5.

Phase-based protocol. FRBench evaluates recovery as a controlled state-transition problem: each trial passes through nominal execution, error projection, and recovery execution, so recovery success is conditioned on a verified adverse state rather than confounded by approach quality.

4.1 Failure-Recovery Data Engine

The data engine in Fig. 5 supplies recovery supervision through two complementary sources, balancing broad coverage with policy-specific realism:

Interception-Based Synthetic Injection. We hook expert-planner execution nodes and inject targeted errors (E1–E4) during critical segments such as grasping or lifting. A hierarchical planner then restores the task state. The injection sequence and perturbation operators are detailed in Appendix A.4.2 and Appendix A.4.3.

Policy-Induced Rollouts. We also collect closed-loop failures from trained base policies and let the expert planner intervene, yielding in-distribution recovery data for model-specific drift; detection and real-world recovery details appear in Appendix A.4.4.

Both sources are merged into raw recovery trajectories $\mathcal{D}_{rec}^{raw}$ for VCR; RAI derives $\mathcal{D}_{rec}^{reset}$ from them via TSER. The synthetic stream provides controlled coverage of common physical breakdowns, while the policy-induced stream captures the errors that actually arise from learned visual perception and action generation.

Simulation versus real collection. In FRBench-Sim, an interception module injects an error and a privileged hierarchical planner synthesizes the correction, yielding 6,392 recovery episodes without a human. In FRBench-Real, adverse states are induced manually and teleoperated back to success, since friction, contact, fluids, and deformables are hard to script. Neither is used at deployment: one policy with a fixed high-value token emits nominal and corrective actions, with no online value estimate, failure detector, or retry controller.

Table 2: **FRBench-Sim success rate (%).** We report nominal and injected-failure success over ten RoboTwin tasks. Method labels include venue year and citation. Throughout the result tables, ↑ means higher is better, gray marks RePO-VLA, bold/underline mark best/second-best results, and green/red values denote gains/losses. Here Δ reports average gain over the corresponding $\pi_{0.5}$ baseline.

Model	Split	Task Success Rate (%) ↑										Avg. ↑	Δ Avg. ↑
		Blocks RGB	Blocks Size	Hang Mug	Lift Pot	Move Stap.	Open Laptop	Place Bread	Place Can	Place Mouse	Press Stap.		
Standard (nominal)													
RDT (ICLR'25)	Clean	3	0	23	72	2	59	5	19	1	41	22.5	—
[Liu et al., 2025]	Rand.	0	0	16	9	0	32	1	6	0	24	8.8	—
GO-1 (arXiv'25)	Clean	7	2	0	92	3	65	2	29	15	66	28.1	—
[Bu et al., 2025]	Rand.	3	2	0	92	4	60	1	37	10	51	26.0	—
π_0 (RSS'25)	Clean	19	7	11	84	0	85	23	41	7	62	33.9	—
[Black et al., 2025b]	Rand.	5	1	3	36	2	46	1	5	1	29	12.9	—
$\pi_{0.5}$ (CoRL'25)	Clean	30	20	6	28	10	52	26	30	12	60	27.4	—
[Black et al., 2025a]	Rand.	48	22	10	20	16	74	44	20	28	54	<u>33.6</u>	—
w/o Fail	Clean	82	54	4	38	26	62	34	48	38	60	**44.6**	(+17.2)
	Rand.	68	56	8	36	18	54	50	46	48	56	**44.0**	(+10.4)
Injected failure													
$\pi_{0.5}$ (CoRL'25)	Clean	20	2	0	6	0	64	0	0	0	58	<u>15.0</u>	—
[Black et al., 2025a]	Rand.	22	6	0	0	0	68	0	0	4	54	<u>15.4</u>	—
RePO-VLA	Clean	70	60	0	30	0	90	20	10	10	80	**37.0**	(+22.0)
	Rand.	80	40	20	0	30	100	60	10	20	70	**43.0**	(+27.6)

Tasks. Blocks RGB/Size: Blocks Ranking RGB/Size; Move Stap.: Move Stapler Pad; Place Bread/Can/Mouse: Place Bread Skillet/Place Can Basket/Place Mouse Pad; Press Stap.: Press Stapler.

5 Experiment

5.1 Simulation Evaluation (FRBench-Sim)

We evaluate FRBench-Sim on Dynamic Grasp Failure, a proxy for contact anomalies such as `grasp_slip` and `premature_close`. This perturbation is deliberately low-level: it does not change the task instruction, but it breaks object control and requires the policy to re-enter a valid grasping state. The full perturbation taxonomy is given in Appendix A.4.3. Baselines include RDT [Liu et al., 2025], GO-1 [Bu et al., 2025], π_0 [Black et al., 2025b], $\pi_{0.5}$ [Black et al., 2025a], and the RAI checkpoint, which the tables label Phase I. These imitation baselines are trained only on optimal nominal-success demonstrations. Failed, suboptimal, and recovery trajectories are removed, so a low recovery score is not caused by imitating corrupted actions. It is what success-only imitation does after the state leaves the nominal demonstration distribution.

Protocol. We run 50 rollouts per task in RoboTwin Clean and Random settings. Recovery trials inject a grasp disturbance by holding the gripper open for 30 frames (∼1s) at grasp initiation, forcing the policy to re-establish object control; protocol details and visual examples are in Appendix A.4.1 and Appendix A.6.

Table 2 shows two effects. First, Phase I improves nominal robustness: under randomization, its average success remains stable from 44.6 to 44.0, whereas π_0 drops from 33.9 to 12.9. This indicates that recovery-aware initialization does not trade away base task competence. Second, under injected failures, full RePO-VLA improves over $\pi_{0.5}$ from 15.0/15.4 to 37.0/43.0. The gain is largest in tasks where the perturbation forces a genuine re-grasp or re-alignment, indicating that the combined RAI+VCR training helps the policy return to the success manifold rather than continuing the nominal plan. Baseline models such as RDT and GO-1 show lower success under randomized conditions, highlighting their limited robustness.

5.2 Real-World Evaluation & Recovery Data Scaling

We evaluate on two Dobot X-Trainer arms across *Vegetable Preparation*, *Towel Folding*, *Desk Organization*, and *Liquid Pouring*. These tasks cover precision pouring, deformable-object manipulation,

Table 3: **FRBench-Real success rate (%).** Results are averaged over 10 trials per task under standard and adversarial settings. Method labels include venue year and citation; Δ reports improvement over $\pi_{0.5}$ in the same condition.

Task	Standard (No Perturbation) ↑					Adversarial (With Perturbation) ↑				
	π_0 (RSS'25) [Black et al., 2025b]	$\pi_{0.5}$ (CoRL'25) [Black et al., 2025a]	Phase I	Full (1x)	Δ	π_0 (RSS'25) [Black et al., 2025b]	$\pi_{0.5}$ (CoRL'25) [Black et al., 2025a]	Phase I	Full (1x)	Δ
Pour Water	40	30	60	50	(+20)	20	20	60	30	(+10)
Cook Vegetable	10	10	30	40	(+30)	0	10	30	30	(+20)
Tidy Desk	20	20	40	40	(+20)	10	30	40	30	(+0)
Fold Towel	40	40	40	30	(-10)	20	20	20	30	(+10)
Avg.	27.5	25.0	**42.5**	<u>40.0</u>	(+15.0)	12.5	20.0	**37.5**	<u>30.0</u>	(+10.0)

Table 4: **Recovery-data scaling study (%).** We increase real-world recovery data from 1x to 2x and 4x on two representative tasks. Method labels include venue year and citation; $\Delta_{4x-0.5}$ and Δ_{4x-I} compare Full** (4x) with $\pi_{0.5}$ and Phase I.

Condition	Task	Baselines		RePO-VLA Variants				Improvement ↑	
		π_0 (RSS'25) [Black et al., 2025b]	$\pi_{0.5}$ (CoRL'25) [Black et al., 2025a]	Phase I	Full (1x)	Full* (2x)	Full** (4x)	$\Delta_{4x-0.5}$	Δ_{4x-I}
Standard	Pour Water	40	30	60	50	<u>80</u>	**90**	(+60)	(+30)
	Fold Towel	40	40	40	30	<u>60</u>	**70**	(+30)	(+30)
	Avg.	40	35	50	40	<u>70</u>	**80**	(+45)	(+30)
Adversarial	Pour Water	20	20	60	30	<u>70</u>	**80**	(+60)	(+20)
	Fold Towel	20	20	20	30	<u>60</u>	**70**	(+50)	(+50)
	Avg.	20	20	40	30	<u>65</u>	**75**	(+55)	(+35)

workspace rearrangement, and bimanual coordination. Each task uses 200 expert demonstrations and 50 teleoperated recovery episodes. In the adversarial setting, a human injects one physical perturbation per trial—dynamic displacement, kinematic interference, or forced slippage—at a pre-specified task stage, producing disturbances that are difficult to script exactly but common in physical deployments. Checkpoints are evaluated blind; the protocol is in Appendix A.5.4, and qualitative task settings are shown in Appendix A.7.

Baseline Fragility and Data Scaling. Table 3 shows that imitation baselines degrade under adversarial perturbations. Phase I raises the adversarial average to 37.5%. Full (1x) improves over $\pi_{0.5}$ but does not consistently outperform Phase I: VCR adds the full mixed trajectories, and at 1x the failure and recovery regions are too sparse to model reliably. We do not claim that VCR wins at every data scale.

Table 4 reports the corresponding recovery-data scaling study. Doubling and quadrupling recovery data reduces the 1x bottleneck: Full** (4x) improves average standard and adversarial success over $\pi_{0.5}$ by 45 and 55 points, respectively, and surpasses Phase I under adversarial perturbations. This does not mean value conditioning wins at every scale.

5.3 Ablation Studies

Figure 6(a) shows that directly adding raw failure-recovery trajectories performs substantially worse than Phase I, so imperfect data must be organized rather than appended uniformly. Table 5 gives the matched Fold Towel controls, including SARM+RA-BC [Chen et al., 2026]. Table 6 compares an RLinf [Yu et al., 2026] reproduction of offline RECAP with matched steps, recovery budget, and blind protocol. Details are in Appendix A.5.

Decay Rate. Figure 6(c) identifies $\alpha = 3$ as the best trade-off. A small α insufficiently penalizes near-terminal failure actions, whereas a large α suppresses useful pre-failure behavior.

Table 5: **Fold Towel controls (%).** Δ is versus Success-only BC. "–" was not evaluated. Full-4x without value and RePO-VLA Full-4x share initialization, data, budget, and protocol.

Setting	Data	Value	Standard ↑		Adversarial ↑	
			Rate	Δ	Rate	Δ
Success-only BC	None	No	40	—	20	—
Raw DAgger-style	Raw	No	0	(-40)	–	–
Phase-I DAgger-style	Sliced + reset	No	40	(+0)	20	(+0)
Full-4x, no value	Full 4x	No	20	(-20)	20	(+0)
SARM+RA-BC, Full-4x	Full 4x	External	50	(+10)	20	(+0)
RePO-VLA Full-4x	Full 4x	Yes	**70**	(+30)	**70**	(+50)

Table 6: **Offline RECAP comparison (%).** RECAP-1x (N=50) and RECAP-2x (N=100) use the same $\pi_{0.5}$ backbone. Bold/underline rank RECAP-1x, RECAP-2x, and Full-2x. Δ is Full-2x minus RECAP-2x. Only the offline advantage-conditioned component is compared.

Condition	Task	Baseline and RECAP			RePO-VLA	
		$\pi_{0.5}$ CoRL'25	RECAP-1x N=50	RECAP-2x N=100	Full-2x RePO-VLA	Δ
Standard	Pour Water	30	60	**80**	**80**	(+0)
	Fold Towel	40	50	**70**	60	(-10)
	Average	35	55	**75**	70	(-5)
Adversarial	Pour Water	20	40	**70**	**70**	(+0)
	Fold Towel	20	20	50	**60**	(+10)
	Average	20	30	60	**65**	(+5)

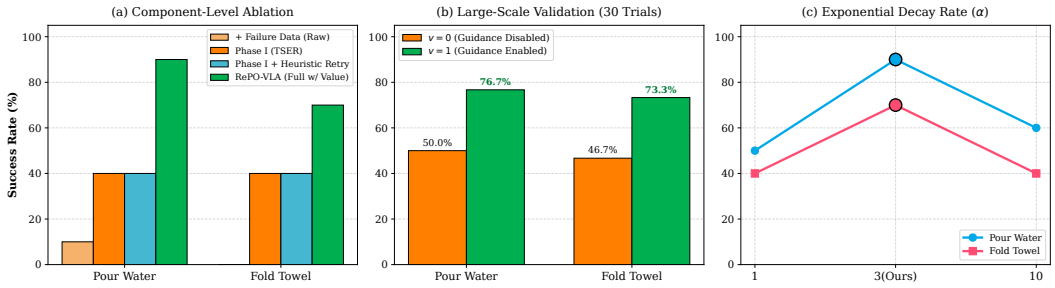


Figure 6: **Real-world ablations.** (a) Raw aggregation is worse than Phase I; see Table 5 for matched controls. (b) On Fold Towel, the high-value token instead of $v = 0$ raises success from 14/30 (46.7%) to 22/30 (73.3%). (c) On Fold Towel, $\alpha = 1, 3, 10$ yield 40%, 70%, and 40%.

6 Conclusion

We presented RePO-VLA, a recovery-driven framework for training VLA policies from nominal-success, failed, and recovered trajectories. By combining episode-reset recovery initialization with progress-aware value conditioning, RePO-VLA converts mixed-quality interaction data into autonomous corrective behavior. The central result is that failures need not be discarded: their useful pre-failure segments can provide kinematic priors, while their terminal regions can be safely down-weighted through semantic progress labels. Results on the Failure-Recovery Benchmark (FRBench) show that explicit recovery supervision and dense value guidance are key ingredients for reliable long-horizon bimanual manipulation. The method supplies this supervision during fine-tuning and does not replace the pretrained backbone.

7 Limitations and Future Work

RePO-VLA is recovery-aware optimization on observed failure families, not open-world zero-shot recovery. FRBench-Sim covers premature closure, grasp slip, position offset, and orientation mismatch. FRBench-Real adds deformable and fluid interaction, dynamic displacement, forced slippage, and bimanual kinematic interference. Severe occlusion, tool misuse, subgoal mismatch, and major perception errors are outside this evaluation. Teleoperated recovery limits scale; the planner and the operator are used only offline. Reducing that annotation, and modeling failure inside pretraining, are left as future work.

Acknowledgments and Disclosure of Funding

This work was supported by the National Key Research and Development Program of China (Grant No. 2024YFE0203100); the Scientific Research Innovation Capability Support Project for Young Faculty (Grant No. ZYGXQNJSKYCXNLZCXM-I28); the National Natural Science Foundation of China (NSFC) (Grant Nos. 62476293, 62636010, and U24A20329); the Shenzhen Science and Technology Program (Grant No. QNXMA20250701093544048); the Guangdong Basic and Applied Basic Research Foundation (Grant No. 2024B1515120024); the General Embodied AI Center of Sun Yat-sen University; and the Major Key Project of Pengcheng Laboratory (Grant No. PCL2025A17).

Competing interests. The authors declare no financial relationships outside the submitted work during the 36 months prior to submission that could be perceived as influencing this work.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 1577–1594. PMLR, 23–29 Jul 2023.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40. PMLR, 27–30 Sep 2025a.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025b.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, et al. π_{RL}: Online rl fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 2025a.

Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Fred Shentu, and Philipp Wu. Sarm: Stage-aware reward modeling for long horizon robot manipulation. In *International Conference on Learning Representations*, volume 2026, pages 17091–17116, 2026.

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

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. Conrft: A reinforced fine-tuning method for vla models via consistency policy. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025c.

Yinpei Dai, Jayjun Lee, Nima Fazeli, and Joyce Chai. Racer: Rich language-guided failure recovery policies for imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15657–15664. IEEE, 2025.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Yanjiang Guo, Yen-Jen Wang, Lihan Zha, and Jianyu Chen. Doremi: Grounding language model by detecting and recovering from plan-execution misalignment. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12124–12131. IEEE, 2024.

Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. Improving vision-language-action model with online reinforcement learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15665–15672. IEEE, 2025.

Sami Haddadin, Alessandro De Luca, and Alin Albu-Schäffer. Robot collisions: A survey on detection, isolation, and identification. *IEEE Transactions on Robotics*, 33(6):1292–1312, 2017.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S Brown, and Ken Goldberg. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning. *arXiv preprint arXiv:2109.08273*, 2021.

Huang Huang, Fangchen Liu, Letian Fu, Tingfan Wu, Mustafa Mukadam, Jitendra Malik, Ken Goldberg, and Pieter Abbeel. Otter: A vision-language-action model with text-aware visual feature extraction. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 25566–25580. PMLR, 13–19 Jul 2025.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

Intelligence, Physical, 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*, 2025.

Yunfan Jiang, Chen Wang, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Transic: Sim-to-real policy transfer by learning from online correction. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 1691–1729. PMLR, 06–09 Nov 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 06–09 Nov 2025.

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. *arXiv preprint arXiv:2510.01642*, 2025.

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 *International Conference on Learning Representations*, 2025.

Zeyi Liu, Arpit Bahety, and Shuran Song. Reflect: Summarizing robot experiences for failure explanation and correction. *arXiv preprint arXiv:2306.15724*, 2023.

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.

Aoran Mei, Guo-Niu Zhu, Huaxiang Zhang, and Zhongxue Gan. Replanvlm: Replanning robotic tasks with visual language models. *IEEE Robotics and Automation Letters*, 9(11):10201–10208, 2024.

Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, 2023.

Pradip Pramanick, Chayan Sarkar, Snehasis Banerjee, and Brojeshwar Bhowmick. Talk-to-resolve: Combining scene understanding and spatial dialogue to resolve granular task ambiguity for a collocated robot. *Robotics and Autonomous Systems*, 155:104183, 2022.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

Shreyas Sundara Raman, Vanya Cohen, Ifrah Idrees, Eric Rosen, Raymond Mooney, Stefanie Tellex, and David Paulius. Cape: Corrective actions from precondition errors using large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14070–14077. IEEE, 2024.

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

Som Sagar, Jiafei Duan, Sreevishakh Vasudevan, Yifan Zhou, Heni Ben Amor, Dieter Fox, and Ransalu Senanayake. Robomd: Uncovering robot vulnerabilities through semantic potential fields. *arXiv preprint arXiv:2412.02818*, 2024.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *International Conference on Learning Representations*, 2026.

Lucy Xiaoyang Shi, Zheyuan Hu, Tony Z Zhao, Archit Sharma, Karl Pertsch, Jianlan Luo, Sergey Levine, and Chelsea Finn. Yell at your robot: Improving on-the-fly from language corrections. *arXiv preprint arXiv:2403.12910*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Marta Skreta, Zihan Zhou, Jia Lin Yuan, Kourosh Darvish, Alán Aspuru-Guzik, and Animesh Garg. Replan: Robotic replanning with perception and language models. *arXiv preprint arXiv:2401.04157*, 2024.

Junjie Wen, Minjie Zhu, Jiaming Liu, Zhiyuan Liu, Yicun Yang, Linfeng Zhang, Shanghang Zhang, Yichen Zhu, and Yi Xu. dvla: Diffusion vision-language-action model with multimodal chain-of-thought. *arXiv preprint arXiv:2509.25681*, 2025a.

Junjie Wen, Yichen Zhu, Minjie Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Xiaoyu Liu, Chaomin Shen, Yaxin Peng, and Feifei Feng. Diffusionvla: Scaling robot foundation models via unified diffusion and autoregression. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 66558–66574. PMLR, 13–19 Jul 2025b.

Philipp Wu, Yide Shentu, Qiayuan Liao, Ding Jin, Menglong Guo, Koushil Sreenath, Xingyu Lin, and Pieter Abbeel. Robocopilot: Human-in-the-loop interactive imitation learning for robot manipulation. *arXiv preprint arXiv:2503.07771*, 2025.

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. *arXiv preprint arXiv:2505.12224*, 2025.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, et al. Rlinf: Flexible and efficient {Large-Scale} reinforcement learning via {Macro-to-Micro} flow transformation. In *20th USENIX Symposium on Operating Systems Design and Implementation (OSDI 26)*, pages 829–846, 2026.

Hongyin Zhang, Zifeng Zhuang, Han Zhao, Pengxiang Ding, Hongchao Lu, and Donglin Wang. Reinbot: Amplifying robot visual-language manipulation with reinforcement learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 77254–77271. PMLR, 13–19 Jul 2025.

Zijian Zhang, Kaiyuan Zheng, Zhaorun Chen, Joel Jang, Yi Li, Siwei Han, Chaoqi Wang, Mingyu Ding, Dieter Fox, and Huaxiu Yao. Grape: Generalizing robot policy via preference alignment. *arXiv preprint arXiv:2411.19309*, 2024.

Fangqi Zhu, Zhengyang Yan, Zicong Hong, Quanxin Shou, Xiao Ma, and Song Guo. Wmpo: World model-based policy optimization for vision-language-action models. In *International Conference on Learning Representations*, 2026.

A Appendix

A.1 Roadmap and Acronyms

The appendix is organized as follows; each entry links to the corresponding section.

- Appendix A.2: Additional Details on RePO-VLA covers episode reset, the RAI/VCR data split, detector-free inference, reliability decay, and compute resources.
- Appendix A.3: More Related Work expands the compact discussion in the main paper.
- Appendix A.4: Additional Details on FRBench specifies the protocol, perturbation taxonomy, detection logic, and dataset composition.
- Appendix A.5: Added Real-World Controls explains the DAgger-style procedure, the SARM progress diagnostic, the RECAP data distinction, and the blind adversarial protocol. The success rates are in Section 5.3.
- Appendix A.6: Visualization of Recovery Data shows simulated failure-recovery episodes.
- Appendix A.7: Qualitative Real-World Evaluation shows the real-world task settings.

Table 7 summarizes recurring abbreviations and notation.

Table 7: **Acronym and notation reference.**

Term	Full name	Meaning in this paper
VLA	Vision-Language-Action	Robot policy family that maps visual observations and language instructions to actions.
RePO-VLA	Recovery-driven Policy Optimization for VLA	Our training framework for autonomous recovery from physical execution drift.
FRBench	Failure-Recovery Benchmark	Benchmark and data engine for evaluating recovery after controlled adverse-state projection.
RAI	Recovery-Aware Initialization	First training stage. It injects recovery priors from expert demonstrations and sliced corrective episodes.
TSER	Trajectory Slicing with Episode Reset	RAI preprocessing that drops the failure prefix and restarts the corrective suffix as a new episode at $t = 0$.
PAS-VF	Progress-Aware Semantic Value Function	Dense progress estimator based on semantic alignment to successful trajectory features.
VCR	Value-Conditioned Refinement	Second training stage. It fine-tunes the policy on mixed trajectories with dense value labels.
SFT	Supervised Fine-Tuning	Imitation-style fine-tuning on action-labeled demonstrations.
HITL / MITL	Human-/Model-in-the-Loop	External correction or replanning mechanisms used by prior recovery systems.
RLFT	Reinforcement Fine-Tuning	Policy refinement using reward or preference feedback.
V-JEPA	Video Joint-Embedding Predictive Architecture	Frozen spatiotemporal encoder used by PAS-VF.
E1–E4	Error types 1–4	FRBench perturbation taxonomy: premature close, grasp slip, position offset, and orientation mismatch.

A.2 Additional Details on RePO-VLA

A.2.1 Why Episode Reset Matters

Recovery demonstrations naturally contain a failure prefix followed by corrective actions. The base policy reads a single-timestamp observation and does not stack historical image frames, so "reset" does not clear an observation buffer. The risk is at the episode level: if the prefix and the suffix stay in one training episode, the policy can associate corrective actions with the preceding mistake. TSER removes that shortcut during RAI by discarding the failure prefix and writing the corrective suffix as a new episode that begins at $t = 0$. The model therefore infers the correction from the present observation, such as an empty gripper or a displaced object.

A.2.2 Episode Boundaries Across RAI, VCR, and Inference

RAI and VCR use different episode boundaries, not different observation-history buffers. RAI trains on $\mathcal{D}_{rec}^{reset}$, where each corrective suffix is its own episode, so the failure prefix cannot act as a trigger. VCR returns to $\mathcal{D}_{rec}^{raw}$ and keeps the prefix and the suffix in one episode, using low value on the failure-inducing actions and high value on the corrective actions. At deployment there is no episode reset and no online value estimate: the policy sees the current observation and the fixed high-value token $v = 1.0$.

A.2.3 Inference Without Online Failure Detection

RePO-VLA decouples value estimation during training from deployment-time recovery. At test time, we do not estimate $V(\tau)$ online or trigger a recovery module. The policy is always conditioned on the high-value token $v = 1.0$, the normalized success anchor of our labels. This makes high-progress behavior the default target; when drift occurs, the same policy selects actions that move the state back toward the learned success manifold.

A.2.4 Reliability Decay for Pure Failures

Pure failures often contain useful approach behavior before the terminal breakdown. Assigning zero to every frame discards these kinematic priors, while assigning high value near failure rewards harmful actions. The decay label $v_t = V(\tau_{fail})(1 - t/T)^{\alpha}$ keeps early useful behavior and rapidly downweights late failure states. We use $\alpha = 3.0$ in all main experiments.

A.2.5 Compute Resources

All training and data-generation jobs were run on an internal server with two GPUs, each with 80 GB of GPU memory, 128 CPU cores, and 512 GB of host RAM. RAI and VCR used both GPUs and required approximately 30 hours per full training run, including value-label preparation and PAS-VF adapter training. PAS-VF trains only lightweight adapters on top of frozen encoders; the exact optimizer, batch size, learning rate, and step count follow the released training configuration. FRBench-Sim data generation in RoboTwin used up to 128 CPU workers on the same server, with GPU resources used for policy/planner inference and visual feature extraction. The real-world evaluation used two Dobot X-Trainer arms and did not require online policy optimization; deployment only required policy inference on a single GPU.

A.3 More Related Work

This section expands the compact related work discussion in the main paper while keeping the central distinction unchanged: RePO-VLA is not another VLA backbone or external retry module, but a way to reuse success, failure, and recovery data under progress-aligned value supervision.

VLA backbones and action modeling. Recent VLA models have rapidly improved the base policy interface. OpenVLA [Kim et al., 2025] casts actions as language-like tokens, RDT-1B [Liu et al., 2025] scales diffusion modeling for bimanual manipulation, and the π series [Black et al., 2025b,a] uses flow matching to support continuous robot control. DiffusionVLA [Wen et al., 2025b] and OTTER [Huang et al., 2025] further study unified diffusion/autoregressive training and text-aware

visual extraction. These works strengthen nominal policy capacity, but they do not specify how failed rollouts should be decomposed into useful prefixes, low-value drift, and corrective recovery.

Failure recovery and runtime correction. Recovery-oriented systems often add an external reasoning or correction layer. ReplanVLM [Mei et al., 2024], RACER [Dai et al., 2025], and RoboFAC [Ye et al., 2025] use visual-language reasoning or failure analysis to produce recovery plans, while FailSafe [Lin et al., 2025] synthesizes failure-recovery data for VLA training. RoboMD [Sagar et al., 2024] studies policy vulnerabilities and motivates explicit failure-aware evaluation. RePO-VLA is complementary to these efforts, but differs by internalizing recovery into the low-level policy through dense value labels rather than relying on a separate detector, planner, or uniformly positive recovery imitation.

Learning from mixed-quality robotic data. Offline-to-online RL and preference-style refinement provide useful tools for reusing non-ideal data. RLPD [Ball et al., 2023] improves online RL with offline demonstrations, ConRFT [Chen et al., 2025c] aligns VLA policies through consistency-based reinforced fine-tuning, ReinboT [Zhang et al., 2025] leverages dense returns for visual-language manipulation, and WMPO [Zhu et al., 2026] optimizes VLA behavior with learned world-model feedback. RePO-VLA targets a more specific failure-recovery structure: it gives different semantic roles to successful segments, failure prefixes, terminal drift, and corrective actions, allowing the policy to learn distributional boundaries between harmful and reparative behavior.

A.4 Additional Details on FRBench

This section details the FRBench failure-recovery data engine, including the phase-based protocol, synthetic error injection mechanism, autonomous detection logic, and dataset scale.

A.4.1 Phase-Based Evaluation Protocol Details

FRBench logs a phase variable $\phi_t \in \{\text{NOMINAL}, \text{ERROR}, \text{RECOVERY}\}$ for every evaluation trajectory. A rollout begins in NOMINAL, where the policy is evaluated under the same task specification as a standard success benchmark. At a task-specific critical node t_{crit}, the benchmark enters ERROR and applies a deterministic perturbation operator from the E1–E4 taxonomy. Recovery evaluation begins only after the adverse state is verified, so the recovery score is conditional on a physical breakdown having occurred rather than on the policy merely reaching the perturbation point. During RECOVERY, the policy continues from the perturbed state and must satisfy the task's recovery condition without an external retry controller. This phase separation prevents nominal improvements, such as better reaching or grasp timing, from being mistaken for genuine corrective behavior.

A.4.2 The 5-Step Interception Sequence

The synthetic error injection follows a strictly controlled execution flow that intercepts the native planner without altering the base pipeline. To autonomously generate paired failure-recovery episodes at scale, the generation process follows a strict five-step sequence:

Step 1: Nominal Reference Generation: The expert oracle computes a complete, successful trajectory τ_{nom} for a given task specification.

Step 2: Execution Hooking: We continuously monitor the task progression and intercept the critical execution node, such as the exact moment of a grasp or lift motion (t_{crit}).

Step 3: Perturbation Injection: At t_{crit}, we inject a targeted error type (E1–E4) by overriding the oracle's standard control output a_t with predefined perturbation operators.

Step 4: Adverse State Projection: The robot executes the perturbed trajectory, deliberately inducing a natural physical breakdown. This reliably projects the system into a deterministic "adverse state."

Step 5: Hierarchical Recovery: Once the failure is registered, the hierarchical recovery planner is activated to dynamically replan and synthesize the corrective actions needed to restore the task state.

A.4.3 Mathematical Formulation of Perturbation Taxonomy (E1–E4)

A core contribution of FRBench is the parameterization of physical breakdowns. Let the nominal base action at timestep t be defined as $a_t = [p_t, R_t, g_t]$, where $p_t \in \mathbb{R}^3$ is the end-effector position, $R_t \in SO(3)$ is the orientation, and g_t represents the continuous gripper state. To guarantee reproducible execution drift without relying on stochastic physics engine artifacts, the four error types are implemented as direct control overrides:

- **E1:** `premature_close`**.** The gripper closes early during the approach stage of a grasp primitive, often resulting in an empty grasp or partial contact. Mathematically, we override the gripper command to a closed state prior to reaching the target: $g'_t = g_{closed} \quad \text{for } t \in [t_{approach}, t_{approach} + N_{hold}]$.
- **E2:** `grasp_slip`**.** The object is released during the lift phase to simulate insufficient grasp force, causing a drop or loss of grasp. We model this dynamic slippage proxy by forcefully opening the gripper for a short, deterministic window to abruptly break contact forces: $g'_t = g_{open} \quad \text{for } t \in [t_{lift}, t_{lift} + 30]$.
- **E3:** `grasp_position_offset`**.** A cm-level translational offset is applied to the grasp target pose, modeling visual misalignment and localization bias. The perturbed position is defined as $p'_t = p_t + \delta p_{trans}$, where $\delta p_{trans} \sim \mathcal{U}(-d, d)$, forcing the gripper to close at an offset.
- **E4:** `grasp_orientation_mismatch`**.** A large orientation mismatch, coupled with a stabilizing lateral shift, is applied to induce a realistic misgrasp without trivial physical collisions. The rotational override is formulated as $R'_t = R_t \cdot \Delta R_{err}$, accompanied by an adjustment $p'_t = p_t + \delta p_{lat}$.

The E1–E4 operators define controlled breakdowns in FRBench-Sim. Real-world adversarial trials do not replay these scripted gripper overrides. Each real trial applies one physical perturbation—dynamic displacement, forced slippage, or bimanual kinematic interference—at a pre-specified task stage, as detailed in Appendix A.5.4.

A.4.4 Detailed Detection and Recovery Logic

Autonomous Failure Detection: In simulation, relying solely on collision or object-drop heuristics is insufficient for complex tasks. Therefore, failure events $\mathcal{F}_t$ are triggered when the execution horizon exceeds a task-specific threshold defined by the maximum nominal success duration:

$$\mathcal{F}_t = \mathbb{I}(t > T_{max}), \quad \text{where } T_{max} = \max_{i \in \{1,\dots,n\}} (T^i_{succ}) \tag{6}$$

This temporal heuristic allows the data engine to autonomously capture insidious failure modes—such as execution loops or progress stagnation—that do not trigger immediate physical collisions but definitively signify task failure.

Real-World Recovery Protocol: Unlike the automated simulation planner, real-world recovery episodes are curated via human-in-the-loop (HITL) teleoperation. Operators intentionally drive the robot into "Adverse States" (e.g., an object partially slipped, or inter-arm kinematic misalignment) and provide a continuous corrective demonstration back to the nominal success manifold. This ensures that the high-value recovery behaviors (τ^+_{rec}) are strictly grounded in real-world friction and contact dynamics, which are often unmodeled in simulation environments.

A.4.5 Dataset Statistics and Composition

The FRBench-Sim dataset comprises a large-scale collection of bimanual manipulation trajectories. It is generated via our automated pipeline within the RoboTwin simulation platform, utilizing the ALOHA-Agilex embodiment. In total, the dataset contains **23,453 episodes**: **17,061** nominal-success trajectories with no injected error, and **6,392** verified failure-recovery trajectories. Nominal episodes are unperturbed. Each retained failure-recovery episode contains one injected error. The E1–E4 counts below are injection attempts recorded before this filter, not a second inventory of the 23,453 episodes. Those attempts sum to 16,912, of which 6,392 were kept as verified recovery episodes.

Table 8: **FRBench-Sim dataset statistics.** Counts summarize the generated trajectory corpus and the aggregate error-injection coverage.

Category	Count	Description
Total episodes	23,453	All generated bimanual manipulation episodes
Nominal success	17,061	Unperturbed successful trajectories, with no injected error
Failure-recovery	6,392	Verified recovery episodes retained after filtering; each has one injected error
Task diversity	46	Distinct bimanual tasks across daily and industrial settings
Environment modes	2	Clean and randomized task instantiations
E1: Premature Close	8,022	Early gripper closure during approach; pre-filter attempts
E2: Grasp Slip	3,516	Forced gripper opening during lift; pre-filter attempts
E3: Position Offset	4,686	Translational grasp-target offset; pre-filter attempts
E4: Orientation Mismatch	688	Rotational mismatch with a lateral shift; pre-filter attempts

Note. The four error-type rows sum to 16,912 pre-filter injection attempts. They are not added on top of the 23,453 episodes. Of those attempts, 6,392 were retained as verified failure-recovery episodes. The other 17,061 episodes are nominal successes and were not injected.

A.5 Added Real-World Controls

A.5.1 DAgger-Style Aggregation

Table 5 reports the Fold Towel controls in the main paper. The DAgger-style setting rolls out the policy to collect adverse states, obtains corrective trajectories from an expert planner or a human operator, aggregates them with the existing dataset, and then performs supervised fine-tuning. It is called DAgger-style rather than canonical DAgger because the expert provides a corrective trajectory suffix after an adverse state, instead of relabeling every visited state with a single expert action. Raw DAgger-style aggregation keeps the complete failure-recovery trajectory, including the failure-inducing prefix and the corrective suffix. Phase-I DAgger-style keeps only the expert corrective suffix and treats it as an independent episode with a reset boundary. The failure-inducing prefix should not be treated as expert supervision. Recovery data provides coverage of adverse states and corrective behaviors, while value-conditioned refinement helps the policy use this mixed-quality data. This is consistent with the scaling result in Table 4, where RePO-VLA on Fold Towel improves from 30% at 1x to 60% at 2x and 70% at 4x.

A.5.2 SARM+RA-BC

SARM+RA-BC is an alternative learned-progress baseline from one stage-aware reward model [Chen et al., 2026]. It does not use the PAS-VF labels or the temporal reliability-decay formulation. SARM estimates stage-aware progress. RA-BC is the behavior-cloning rule in that same model: it uses the progress difference between adjacent action chunks to filter or reweight the imitation loss. On Fold Towel, uniform Full-4x fine-tuning without value reaches 20% standard and 20% adversarial success. SARM+RA-BC reaches 50% and 20%. RePO-VLA Full-4x reaches 70% and 70%. The strict matched pair is uniform Full-4x without value versus RePO-VLA Full-4x. SARM+RA-BC is the external value-model baseline.

Learned progress is useful for mixed-quality data under the standard setting, but in this experiment it does not add robustness under forced perturbations. After a physical failure, the scene may still appear partially complete, so a progress estimator can output a moderate or plateaued value. Using that value directly would assign positive utility to actions near the failed endpoint. Reliability decay converts estimated progress into training utility: it preserves useful early motion while reducing the weight of actions approaching terminal failure.

On held-out Dobot failure-recovery rollouts, SARM generally captures the overall task order, but it occasionally shows stage mismatch, oscillation during re-grasping or backtracking, and spurious progress increases after a physical failure. Those errors can cause RA-BC to retain failure-inducing chunks or suppress corrective chunks. This observation is a qualitative diagnostic. The 50%/20% standard and adversarial results above are the policy-level evidence.

A.5.3 Offline RECAP Comparison

Table 6 in the main paper reports the offline comparison. Official RECAP code and checkpoints were not available at submission. During the discussion period, the open-source RLinf [Yu et al., 2026] reproduction was ported to the Dobot platform. The comparison uses the same $\pi_{0.5}$ backbone,

matched training steps, equal recovery-data budgets, and the same blind evaluation protocol. Because the complete RECAP system includes an autonomous online improvement loop, the experiment compares its advantage-conditioned learning component under a fixed offline budget. RECAP-1x is labeled $N = 50$ and RECAP-2x is labeled $N = 100$. RECAP collects experience from $\pi_{0.5}$ rollouts, where severe failures occur less frequently than in the targeted recovery-data collection, so its dataset contains a larger proportion of nominal and near-nominal experience. RePO-VLA deliberately induces adverse states and collects corrections from them, producing denser coverage of forced slips, displacement, and recovery states. The two results are complementary. Neither is evidence that one method dominates the other.

A.5.4 Blind Adversarial Protocol

The real-world evaluation is blind. Each checkpoint is served through a separate `ip:port`, and the port-to-method mapping is held by an author who does not participate in the trials. The operator selects only a port and does not know the method identity. All methods use the same scene, initial poses, control frequency, success criteria, and 10 trials per task and condition.

Each adversarial trial receives exactly one physical perturbation. Its type, nominal magnitude, and task-phase trigger are fixed by a pre-specified schedule, and the operator does not intervene again. Triggering by semantic phase rather than wall-clock time places the perturbation at the same task stage for policies with different execution speeds. All checkpoints were selected with the same pre-specified validation criterion and frozen before adversarial testing. No adversarial-test outcome was used for checkpoint selection.

A.6 Visualization of Recovery Data

This appendix visualizes representative data samples generated by our automated pipeline within the RoboTwin simulation platform, utilizing the ALOHA-Agilex embodiment. The purpose is to intuitively illustrate the complete workflow of our system, specifically the physics-driven error injection and the subsequent autonomous recovery process.

For each of the four core error types (E1–E4), we provide six task examples across both Clean and Random (domain-randomized) environments to ensure visual and kinematic diversity. Each task sequence consists of 9 keyframes sampled from a single episode:

- `Frames 1-3 (Error):` Visualize the "Error Attempt" phase, where physics-driven perturbations (rather than simple Gaussian noise) are introduced to cause a natural failure.
- `Frames 4-8 (Recovery):` Display the "Recovery" phase, featuring smooth, planner-based trajectories that include re-perception and corrective actions.
- `Frame 9 (Normal):` Illustrate the robot successfully resuming the "Normal" execution phase from a corrected state.

A.7 Qualitative Results: Real-World Evaluation

In this section, we present qualitative visualizations of our real-world settings to complement the quantitative results in Section 5.2. Figure 15 illustrates the four challenging bimanual manipulation tasks used in our evaluation: *Pour Water*, *Cook Vegetable*, *Tidy Desk*, and *Fold Towel*. These tasks are selected to cover a wide range of manipulation primitives, including precision pouring, insertion, deformable object manipulation, and long-horizon coordination.

Comparing the execution traces under adversarial perturbation, RePO-VLA responds to forced deviations (e.g., external pushes or object slippage) by synthesizing valid recovery trajectories that resume task progress. In contrast, baseline methods often fail to react to these deviations, leading to task failures.

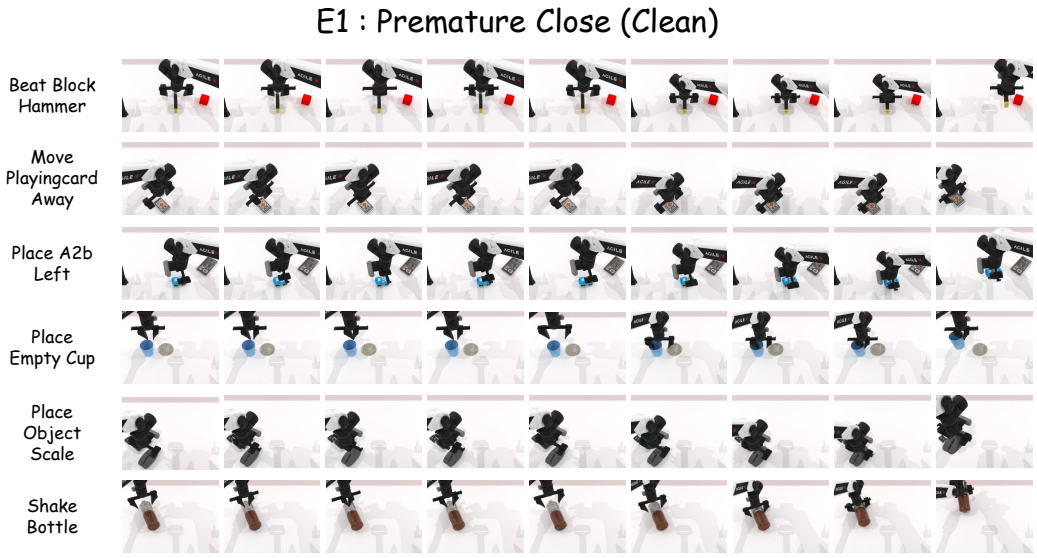


Figure 7: Premature Close (Clean)



Figure 8: Premature Close (Random)

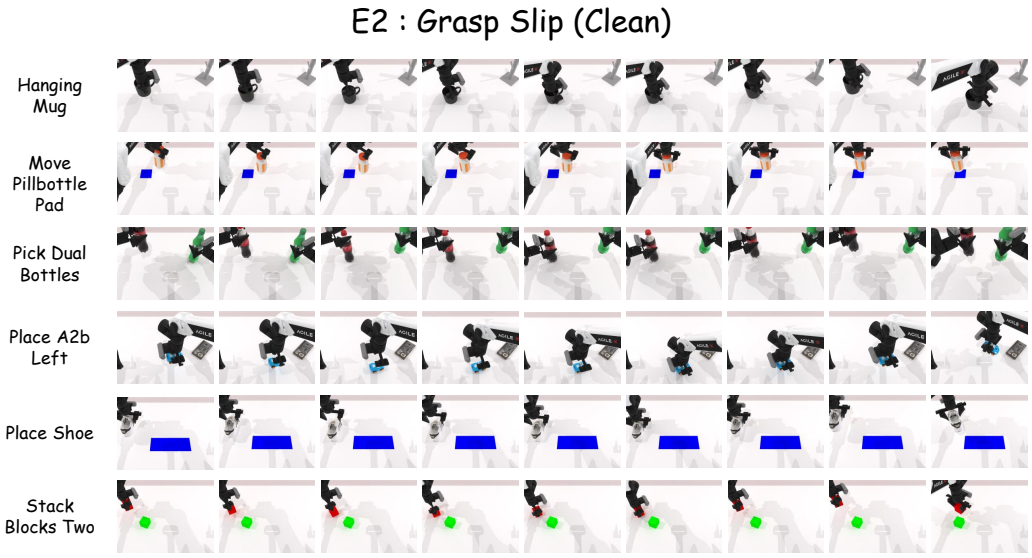


Figure 9: Grasp Slip (Clean)

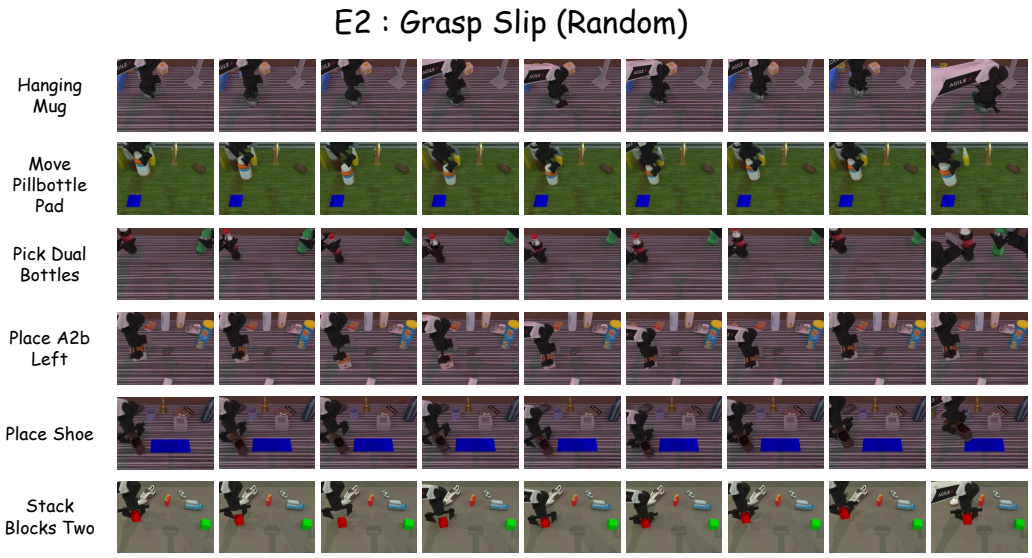


Figure 10: Grasp Slip (Random)

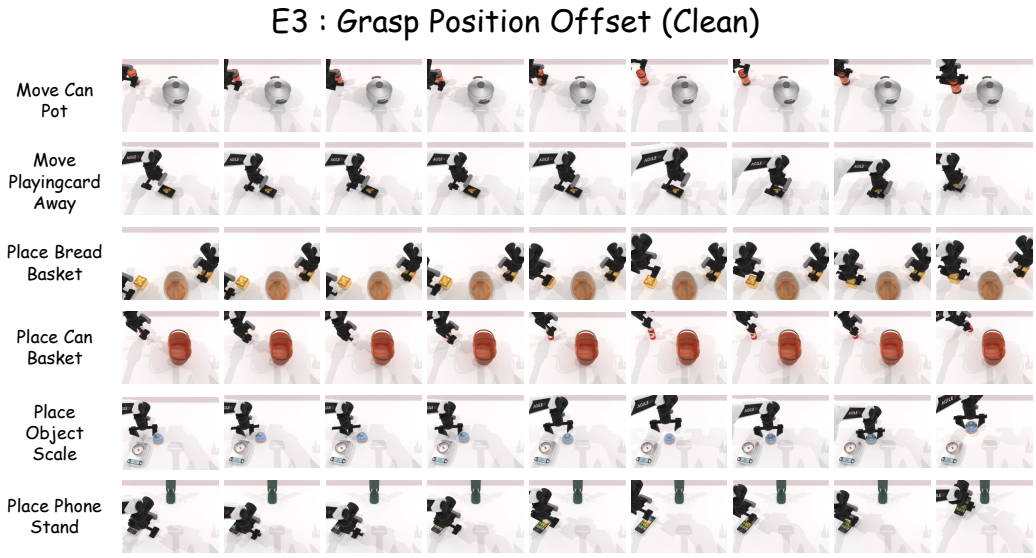


Figure 11: Grasp Position Offset (Clean)



Figure 12: Grasp Position Offset (Random)

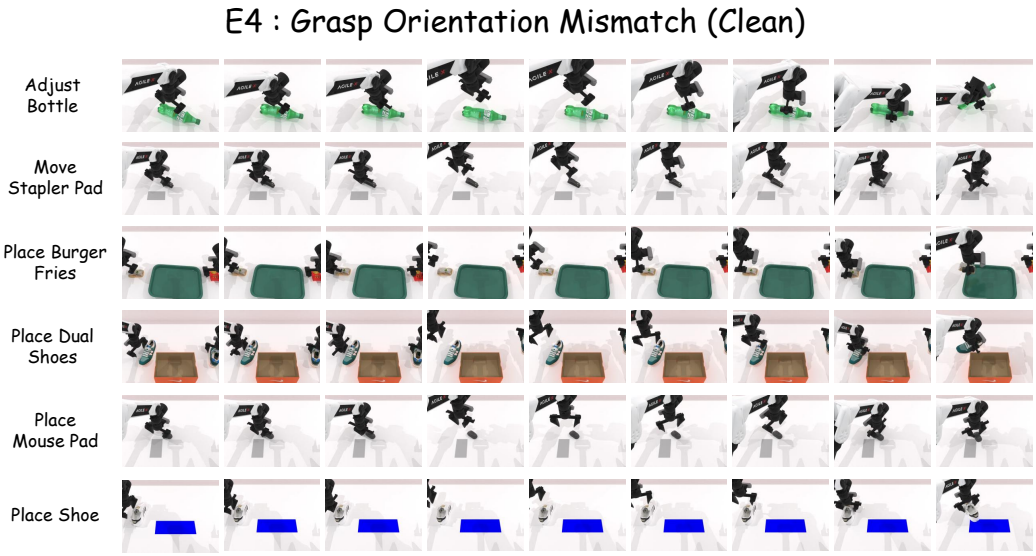


Figure 13: Grasp Orientation Mismatch (Clean)

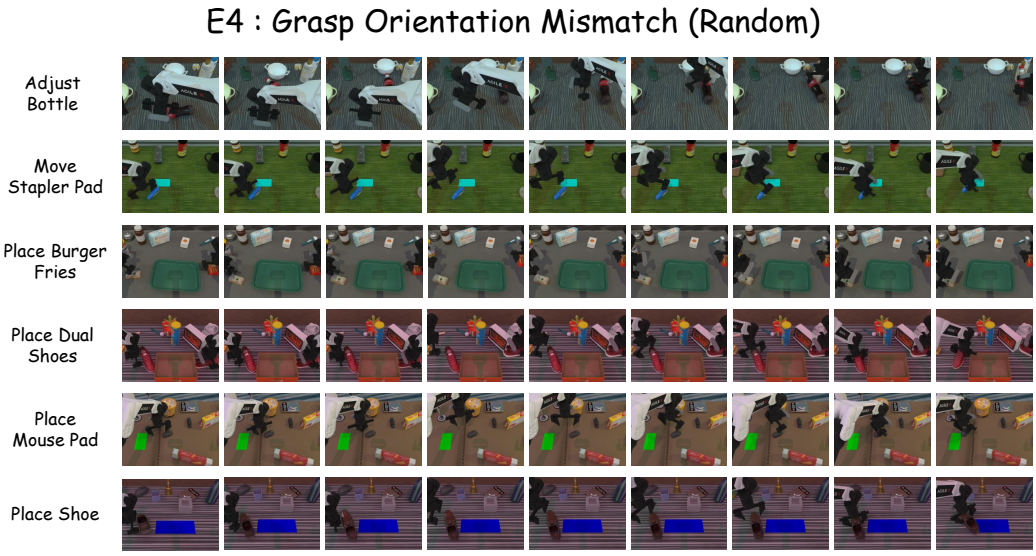


Figure 14: Grasp Orientation Mismatch (Random)



Figure 15: Real-World Evaluation Tasks. We evaluate our method on four contact-rich bimanual tasks. The figure showcases the environmental setup and key interaction phases for each task.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly describe RePO-VLA's main contributions, including its recovery-driven training framework, progress-aware value function, and robust performance under execution drift. These claims match the experimental validation presented in Section 5, and the scope reflects both simulated and real-world tasks.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The limitations are discussed in Section 7: RePO-VLA targets observed failure families rather than open-world zero-shot recovery; severe occlusion, tool misuse, subgoal mismatch, and major perception errors are outside the evaluation; and teleoperated real-world recovery data limits scale. Reducing this annotation cost and modeling failure during pretraining are left as future work.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

   Justification: This paper does not include formal theoretical results, theorems, or proofs. The focus is on empirical evaluation and framework development.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Section 5 and the appendix describe the method components, data composition, task setups, perturbation protocols, evaluation procedures, and hardware used for the main results. Further implementation details and reproduction instructions will be released with the code and data.

   Guidelines:

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

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

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

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

   Answer: [Yes]

   Justification: The code, data, and related resources will be progressively open-sourced, together with instructions for reproducing the main results.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 5 specifies the task setups, perturbation settings, and evaluation protocols needed to understand the results (50 rollouts per task in RoboTwin; 10 blind trials per task and condition in the real world), with further protocol details in Appendix A.4 and compute resources in Appendix A.2.5. As permitted by the guidelines, the complete training configurations will be provided with the code in our public repository.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer:[Yes]

Justification: All success rates are computed over a fixed number of trials (50 rollouts per task in RoboTwin; 10 blind trials per task and condition in the real world), as stated in Section 5. For the value-guidance ablation in Figure 6(b), each condition uses 30 trials; the 95% Wilson confidence intervals are [30%, 64%] for $v = 0$ vs. [56%, 86%] for $v = 1$ on Fold Towel, and [33%, 67%] vs. [59%, 88%] on Pour Water. Error bars are not shown for the main tables because repeated real-robot evaluation is costly.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendix A.2.5 reports the number of 80 GB GPUs, CPU workers, 512 GB host RAM, approximate training time, simulation data-generation resources, and real-world inference hardware.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research follows ethical guidelines; no human subjects are involved in the core experiments, and all simulation data is synthetic or robot-collected.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: RePO-VLA aims to improve the reliability of manipulation policies under recoverable execution drift, which may benefit assistive, laboratory, and industrial robot deployments. However, recovery capability is not a safety guarantee: out-of-taxonomy failures or incorrect value conditioning may still produce unsafe actions, so practical deployment should include task-specific validation, physical safeguards, and human override mechanisms. Section 7 discusses the scope of failures covered by our evaluation.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The released models are specific to bimanual robot tasks and pose minimal risk; no high-risk generative models or large-scale scraping are used.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We use RoboTwin 2.0 [Chen et al., 2025b] (MIT License) for simulation tasks and assets; the openpi code base (Apache-2.0) for π_0 [Black et al., 2025b] and $\pi_{0.5}$ [Black et al., 2025a], whose released weights are additionally subject to the Gemma Terms of Use; V-JEPA 2 [Assran et al., 2025] (MIT License) as the frozen visual encoder; and CLIP [Radford et al., 2021] (MIT License) as the frozen text encoder. All assets are cited and used in accordance with their terms.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: FRBench and its failure-recovery data engine are described in Section 4 and Appendix A.4; documentation and usage instructions will be released alongside the assets.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: No human subjects were involved in the experiments.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The method does not rely on LLMs for core functionality; LLMs were not used in experiments or policy optimization.

    Guidelines:

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