RUNTIME RECOVERY FOR FROZEN GENERATIVE ROBOT POLICIES UNDER LONG-TAILED DISTRIBUTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative behavior cloning can produce diverse action distributions and represent complex, multimodal robot behaviors. However, robot execution is also governed by physical dynamics, including contact, friction, and object motion. Small action errors can be amplified during execution, leading to substantial deviations in the robot state, especially when the robot enters long-tail states that are weakly covered by the training data. A common approach is to sample multiple candidate actions from the original policy and select a more suitable one for execution. However, if the policy itself does not cover successful actions at the current state, a limited increase in the number of samples may still fail to identify an effective action. To address this issue, we propose **TailGuard**, a runtime recovery method for generative policies in long-tail states. TailGuard keeps the original policy frozen and uses previously successful rollouts to characterize a task-conditioned success manifold. When execution deviates from this manifold, an external controller, together with a learned short-horizon dynamics model, corrects the robot behavior and guides it back toward states from which successful execution is feasible. Control is then handed back to the original policy once it can reliably continue the task while maintaining action continuity. Across four robotic manipulation benchmarks, TailGuard requires neither additional demonstrations nor failure data, improving the average closed-loop success rate from 49.1% to 70.4% and outperforming retrieval- and residual-based recovery as well as recent recovery methods for diffusion policies.

1 INTRODUCTION

Behavior cloning has become an important learning paradigm for robotic manipulation. Recent generative policies based on diffusion models, flow matching, and vision-language-action models can capture complex, multimodal action distributions (Ho et al., 2020; Chi et al., 2025; Lipman et al., 2022; Shukor et al., 2025). However, robot execution is also governed by physical dynamics such as contact, friction, and object motion: small action errors can be amplified during execution and gradually drive the system away from successful trajectories (Figure 1).

This problem is especially pronounced in long-tail states that are insufficiently covered by the training data. We use *long-tail* broadly to include low-probability states arising from execution-induced covariate shift, including states outside the empirical training-support boundary, which is the regime our evaluation emphasizes. In such states, the original policy may lack sufficient support for successful actions. Although execution quality can be improved by sampling multiple candidate actions and selecting among them, if successful actions themselves are not adequately covered by the policy, increasing the number of samples within a finite budget still cannot achieve effective recovery. Consequently, the diversity of the action distribution is not equivalent to actual recovery capability. Based on this observation, recovery in long-tail states should not rely solely on searching for more actions from the original policy; instead, it should redirect the system toward state regions in which successful behavior can continue to be executed. To this end, we propose **TailGuard**, a runtime recovery method for generative policies. TailGuard keeps the base policy frozen and uses existing successful rollouts to characterize a task-conditioned success manifold. When execution departs

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

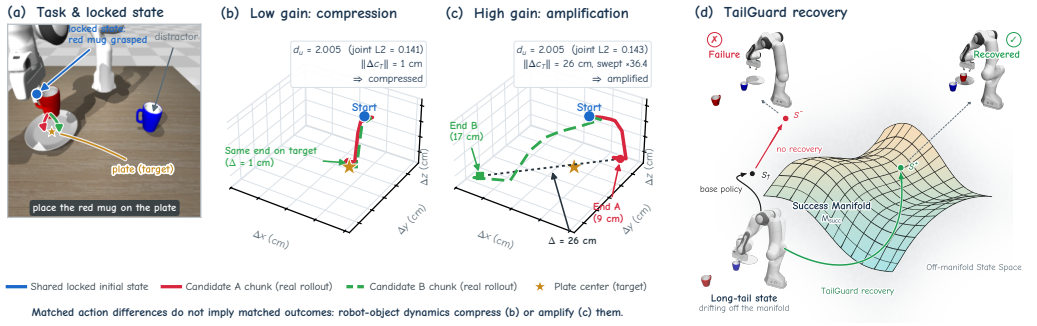

Figure 1: **Motivation.** Robot dynamics reshape action-space diversity into outcome-space diversity: action differences can be compressed into nearly identical outcomes **(b)** or amplified into substantially different futures **(c)** (deterministic MuJoCo rollouts, used only for analysis). **(d)** From a long-tail state that has drifted off the success manifold $\mathcal{M}_{\text{succ}}$, the frozen policy ends in a failed state s^- (red), whereas TailGuard steers execution back onto the manifold to a recoverable state s^+ (green) and completes the task.

from this manifold, an external controller, combined with a learned short-horizon dynamics model, corrects the robot's behavior and guides the system back to a recoverable region.

During recovery, TailGuard also explicitly accounts for the handoff of control. Control is returned to the base policy only when the original policy can reliably continue the task and maintains sufficient action continuity with the recovery controller, thereby avoiding additional disturbances caused by premature switching or discontinuous control. We evaluate TailGuard on four robotic manipulation benchmarks. Without additional demonstrations or failure data, TailGuard raises the mean closed-loop success rate from 49.1% to 70.4% and outperforms retrieval- and residual-based recovery as well as recent recovery methods for diffusion policies.

Our contributions are threefold: (1) we quantify how real dynamics reshape generative action diversity; (2) we characterize the limits of local proxies and multi-candidate sampling for long-horizon control; and (3) we develop a success-manifold-guided recovery and compatibility-gated handoff mechanism for recoverable long-tail failures.

2 Related Work

2.1 Generative Behavior Cloning and Action Diversity

Behavior cloning learns robot policies from expert demonstrations, but sequential execution can amplify prediction errors and induce covariate shift, motivating interactive correction methods such as DAgger (Ross et al., 2011). RoboMimic further shows that imitation performance depends strongly on demonstration quality, observation design, and policy architecture (Mandlekar et al., 2021). Recent policies increase expressiveness through temporal action chunks and multimodal generation: Diffusion Policy models visuomotor control as conditional denoising (Ho et al., 2020; Chi et al., 2025), and flow-matching methods provide continuous-time alternatives (Lipman et al., 2022; Zhang et al., 2025). Large-scale systems such as RT-2, Open X-Embodiment, OpenVLA, and SmolVLA further expand task and embodiment diversity (Brohan et al., 2023; O'Neill et al., 2024; Kim et al., 2024; Shukor et al., 2025). However, diverse sampled actions do not necessarily produce diverse physical outcomes. We therefore distinguish generative action diversity from dynamics-shaped outcome diversity.

2.2 Dynamics, Outcome Structure, and Control Support

Control capability is ultimately determined by the state transitions induced by actions. Sampling-based model predictive control evaluates candidate controls through predicted dynamics (Williams et al., 2017), while generative planners such as Diffuser condition trajectory generation on goals or rewards (Janner et al., 2022). Reachability analysis characterizes whether desired

state sets can be reached (Abate et al., 2008). These issues are especially important in contact-rich manipulation, where dynamics are highly state dependent (Hogan & Rodriguez, 2020). Rather than learning a new planner or solving a global reachability problem, we analyze candidates sampled from a frozen generative policy and measure how the true dynamics reshape their differences.

2.3 Runtime Failure Detection, Recovery, and Policy Handoff

Runtime monitoring methods detect when learned robot policies become unreliable. Sentinel combines temporal action consistency with vision-language monitoring (Agia et al., 2024); FAIL-Detect uses sequential out-of-distribution detection with conformal calibration (Xu et al., 2025); and FIPER combines latent-space out-of-distribution signals with action-chunk uncertainty for generative imitation policies (Römer et al., 2026). These methods mainly address when failure is emerging, rather than how control should be restored. Recovery-oriented methods instead augment policy behavior: RACER adds recovery trajectories and supervisory guidance (Dai et al., 2025), REACH suppresses undesirable diffusion modes through negative prompting (Ke et al., 2026), and FLARE introduces Retry and Reset skills for vision-language-action policies (Zhao et al., 2026). TailGuard differs by keeping the base policy frozen and using an external, dynamics-guided recovery controller that optimizes actions toward a task-conditioned success manifold. It also explicitly separates recovery from policy handoff: control is returned only when the recovered state and action continuity are compatible with continued execution. Accordingly, we evaluate recovery through final closed-loop success rather than latent-space recovery accuracy alone.

3 Method: Dynamics-Shaped Control Support and Guided Recovery

We study a frozen generative behavior-cloning policy under the true robot–environment dynamics and argue in three steps: dynamics reshape action diversity into outcome diversity, finite candidate sampling fails when useful support is scarce, and TailGuard restores execution toward the success manifold of prior successful behavior without retraining the base policy (Figure 2).

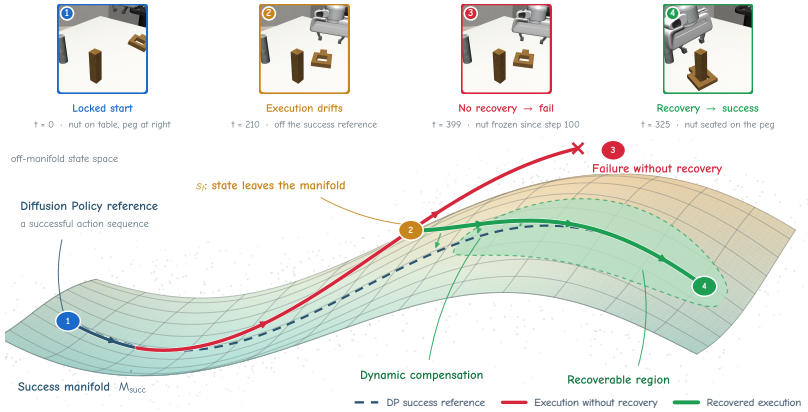

Figure 2: Success-manifold-guided recovery. The frozen policy evolves within the success manifold (1) but may drift away from it (2) and fail (3); TailGuard detects the departure, applies a success-guided correction, and returns control in a compatible recoverable region (4).

3.1 Dynamics-Shaped Outcome Diversity

Let $\pi_\theta(u \mid x)$ denote a frozen generative policy, where x is the current state and u is an action chunk, and let $y = F_H(x, u)$ be the physical outcome after an H-step execution. Around a nominal action u, a small perturbation satisfies

$$\delta y \approx J_H(x, u)\delta u, \qquad J_H(x, u) = \frac{\partial F_H(x, u)}{\partial u}. \tag{1}$$

If the local action distribution has covariance Σ_u, the induced outcome covariance and expected outcome displacement are approximately

$$\Sigma_y \approx J_H \Sigma_u J_H^\top, \qquad \mathbb{E}\big[\|\delta y\|_2^2\big] \approx \operatorname{tr}\big(J_H \Sigma_u J_H^\top\big)\,. \tag{2}$$

Thus, equal action-space differences need not produce equal physical differences: dynamics may amplify some directions while compressing others. Equivalently, the policy induces the physical outcome distribution

$$P_\pi^y(\cdot \mid x) = (F_H(x,\cdot))_{\#}\, p_\pi(\cdot \mid x), \tag{3}$$

where $(\cdot)_{\#}$ denotes the pushforward through the dynamics, so that $P_\pi^y(B \mid x) = \Pr_{u\sim p_\pi(\cdot|x)}[F_H(x,u) \in B]$ for every measurable B. We work with this distribution rather than with a density on outcome space: whenever J_H is rank deficient, $F_H(x,\cdot)$ concentrates the action distribution on a lower-dimensional subset of outcome space and $P_\pi^y(\cdot \mid x)$ admits no Lebesgue density. A multimodal action distribution can therefore correspond to a narrow set of physically distinct outcomes. We use the Jacobian only to characterize short-horizon sensitivity, not long-horizon task value.

3.2 Support Limits of Candidate Sampling

Let $\mathcal{Y}_{\mathrm{succ}}(x)$ denote the measurable set of physical outcomes compatible with successful continuation from state x. The policy mass assigned to this useful support is

$$q(x) = \Pr_{u\sim p_\pi(\cdot|x)}\left[F_H(x,u) \in \mathcal{Y}_{\mathrm{succ}}(x)\right] = P_\pi^y(\mathcal{Y}_{\mathrm{succ}}(x) \mid x)\,. \tag{4}$$

For K conditionally independent candidates, the probability that at least one lies in the useful support, and the budget required to reach probability at least $1-\delta$, are

$$P_K(x) = 1 - (1-q(x))^K\,, \qquad K \geq \frac{\log \delta}{\log(1-q(x))} \approx \frac{\log(1/\delta)}{q(x)}, \tag{5}$$

for small $q(x)$, so the required budget grows inversely with the useful support mass.

This exposes two distinct limitations. A *support failure* occurs when the candidate pool contains no useful continuation. A *utilization failure* occurs when useful candidates exist but are not converted into successful closed-loop execution. Increasing candidate diversity alone cannot resolve the first limitation and does not guarantee resolution of the second. Recovery must therefore act relative to the structure of successful physical behavior rather than relying only on additional sampling.

3.3 Success-Manifold Recovery and Compatibility-Gated Handoff

Success bank and representation. For each task, we construct a success-manifold model from N_{succ} previously successful rollouts. The provenance of these rollouts and their isolation from the evaluation set are specified in Sec. 4.1. Each bank contains only rollouts from the corresponding task. States are sampled at a fixed stride along each rollout, and several action chunks are drawn from the frozen base policy at each sampled state.

Each state–action pair is represented by a visual feature z^{vis} from a frozen pretrained visual encoder and a dynamics feature $z^{\mathrm{dyn}}(x_t,u)$ computed from a learned short-horizon dynamics model f_ϕ, which is conditioned on the projected visual feature and the low-dimensional observation that the base policy itself receives (Appendix B.1). The dynamics feature summarizes predicted end-effector motion, object-pose evolution, and contact information. The two feature blocks are standardized independently using success-bank statistics and separately projected to $d/2$ dimensions each, giving

$$\tilde{z} = \Psi(x_t,u) = [\tilde{z}^{\mathrm{vis}}; \tilde{z}^{\mathrm{dyn}}] \in \mathbb{R}^d. \tag{6}$$

Only the dynamics block depends on the candidate action u. The encoder, the feature dimensions, and the architecture and training details of f_ϕ are provided in Appendix B.

Local manifold estimation. For a query representation $\tilde{z}$, we retrieve its m nearest entries in the task-specific success bank $\mathcal{B}$. Let $\mu(\tilde{z})$ and $\widetilde{\Sigma}(\tilde{z})$ denote the local mean and the Ledoit–Wolf-shrunk covariance of this neighborhood. We define the departure score

$$S(\tilde{z}) = \sqrt{\max\Big[(\tilde{z}-\mu)^\top \widetilde{\Sigma}^\dagger (\tilde{z}-\mu), 0\Big]}, \tag{7}$$

and the corresponding unnormalized success-manifold density

$$\rho_{\text{succ}}(\tilde{z}) \propto \exp\left(-\frac{1}{2}S^2(\tilde{z})\right). \tag{8}$$

Retrieval is conditioned only on task identity and representation similarity; we do not explicitly estimate or condition on task phase.

The deviation threshold is calibrated independently for each benchmark from successful calibration rollouts that are disjoint from both the success bank and the evaluation set:

$$\tau_{\text{dev}} = Q_q\left(\mathcal{C}_{\text{succ}}\right), \qquad \mathcal{C}_{\text{succ}} = \left\{S(\tilde{z}_j^{\text{cal}})\right\}_{j=1}^{N_{\text{probe}}}, \tag{9}$$

where Q_q denotes the empirical q-quantile. The calibration probes are collected from N_{cal} independent successful rollouts; the quantile level and N_{cal} are specified in Appendix B.3.

Detection. At runtime, the frozen policy proposes $u_t^\pi \sim p_\pi(\cdot \mid x_t)$. TailGuard activates when the predicted state–action outcome leaves the calibrated success manifold:

$$S(\Psi(x_t, u_t^\pi)) > \tau_{\text{dev}}. \tag{10}$$

Recovery: ideal target and practical surrogate. Conceptually, recovery biases the frozen policy toward the success manifold, defining the tilted distribution

$$p_{\text{TG}}(u \mid x_t) \propto p_\pi(u \mid x_t)\rho_{\text{succ}}(\Psi(x_t, u))^\lambda. \tag{11}$$

Direct independent sampling from the normalized distribution in Eq. (11) is not available in closed form, while exact MAP optimization would require evaluating the base-policy log density. We therefore use a sample-anchored local surrogate. At recovery step t, the m nearest successful neighbors are retrieved once around the base-policy proposal $\Psi(x_t, u_t^\pi)$. Let μ_t and $\widetilde{\Sigma}_t$ denote the resulting local statistics. They are held fixed during the inner optimization.

The practical recovery objective, minimized over $u \in \mathcal{U}$ to give u_t^{rec}, is

$$\mathcal{L}_{\text{rec}}(u) = \frac{1}{2}\left\|u - u_t^\pi\right\|_2^2 + \frac{\lambda}{2}\left(\Psi(x_t, u) - \mu_t\right)^\top \widetilde{\Sigma}_t^\dagger \left(\Psi(x_t, u) - \mu_t\right). \tag{12}$$

The quadratic term in Eq. (12) acts as an isotropic trust region around the sampled proposal rather than the exact policy log density $-\log p_\pi(u \mid x_t)$ of Eq. (11); TailGuard is thus a *sample-anchored, manifold-guided local correction* rather than exact reweighting (Appendix B.5).

Optimization. Starting from $u^{(0)} = u_t^\pi$, we minimize Eq. (12) using J projected-gradient updates with step size η:

$$u^{(j+1)} = \Pi_{\mathcal{U}}\left[u^{(j)} - \eta\nabla_u\mathcal{L}_{\text{rec}}\left(u^{(j)}\right)\right]. \tag{13}$$

Gradients propagate through the learned dynamics model and the dynamics representation, while μ_t and $\widetilde{\Sigma}_t$ remain fixed. The guidance strength λ, the number of updates J, and the step size η are shared across all benchmarks and listed in Table 6 (Appendix B.8).

Compatibility-gated handoff. Recovery is temporary. At each subsequent control step, the frozen policy again produces u_t^π. Control is returned only when two conditions hold.

First, the base-policy proposal must re-enter the success manifold:

$$S(\Psi(x_t, u_t^\pi)) \le \tau_{\text{dev}}. \tag{14}$$

Second, the switch must satisfy an action-continuity condition. Let Δ_{exec} be the number of low-level actions executed between two consecutive policy queries and let k be the continuity window. For the previously executed recovery chunk u_{t-1}^{rec}, we define

$$C_{\text{action}} = \frac{1}{\sqrt{k}}\left\|u_t^\pi[0{:}k] - u_{t-1}^{\text{rec}}[\Delta_{\text{exec}} : \Delta_{\text{exec}} + k]\right\|_2. \tag{15}$$

Because f_ϕ scores only the segment $u[0{:}H]$ that is actually executed before the next query ($H = \Delta_{\text{exec}}$ on all benchmarks, Table 6), the update in Eq. (13) modifies only that segment, and the tail $u^{\text{rec}}_{t-1}[\Delta_{\text{exec}}{:}]$ coincides with the continuation that the base policy itself planned at the previous step. C_{action} is therefore a receding-horizon consistency test: control is returned only when the base policy, re-queried from the recovered state, proposes what its previous plan intended to do next, i.e., when its behavior has become temporally self-consistent again—the same overlapping-chunk statistic used to detect erratic action chunks in Sentinel (Agia et al., 2024). It is not a smoothness check on the recovery controller's motion; the executed transition itself is kept close to the base proposal by the trust-region term in Eq. (12). Handoff occurs when

$$S(\Psi(x_t, u^\pi_t)) \le \tau_{\text{dev}} \qquad \text{and} \qquad C_{\text{action}} \le \varepsilon_a. \tag{16}$$

The continuity threshold ε_a is calibrated from successful rollouts using the same statistic. If the two conditions are not jointly satisfied within $T_{\max}$ recovery control steps, recovery terminates and control returns to the frozen policy; a recovery-failure flag then blocks re-triggering until the proposal re-enters the success manifold, preventing indefinite switching. Benchmark-specific values and the complete runtime procedure are given in Appendix B, and Figure 4 in Appendix B.7 illustrates the complete mechanism on Franka Red Cup execution frames.

4 Experiments

4.1 Experimental Setup

We evaluate TailGuard across four manipulation benchmarks spanning two robot embodiments, two simulation suites, and two classes of frozen generative policies. Table 1 summarizes the environments, base policies, training data, observation and action spaces, and episode horizons.

Table 1: Experimental setup for the four manipulation benchmarks. PH denotes the proficient-human demonstration set of RoboMimic.

Task	Environment / Policy	Training data	Observation	Action	Horizon
OMY Red Cup (Park, 2025)	MuJoCo / SmolVLA	200 demos 40k frames	2 RGB + 7-D state	7-D chunk 50	240
Franka Red Cup (Park, 2025)	MuJoCo / DP	200 demos 20k frames	2 RGB + 8-D state	8-D 16/8	360
Square (Mandlekar et al., 2021)	RoboSuite / DP	PH 200 demos	23-D state 2 frames	10-D pose 16/8	400
Can (Mandlekar et al., 2021)	RoboSuite / DP	PH 200 demos	23-D state 2 frames	10-D pose 16/8	400

For OMY and Franka Red Cup, simulated in MuJoCo (Todorov et al., 2012), success requires the cup to be placed within 10 cm of the plate center and released above the target. For Square and Can, we use the task success signal provided by RoboSuite (Zhu et al., 2020). OMY and Franka policies are trained for 20k steps, whereas Square and Can use the official pretrained checkpoints.

Table 2: Initial-state distribution protocol across the four benchmarks.

Benchmark	Head	Low-density	Near-Tail	Mid-Tail	Far-Tail
OMY / Franka	High-density states	Low-density in-support states	$0 < v \le 0.25$	$0.25 < v \le 1$	$v > 1$
Square	$(0, 0, 0)$	$(2.5, -2.5)$ cm 0.15 rad	$(5.5, -2)$ cm 0.25 rad	$(8.5, 2.5)$ cm 0.45 rad	$(12, -4)$ cm 0.70 rad
Can	$(0, 0, 0)$	$(2, -2)$ cm 0.15 rad	$(4, -3)$ cm 0.25 rad	$(6, 3)$ cm 0.45 rad	$(9, -5)$ cm 0.70 rad

Table 2 summarizes the initial-state protocol. Head and Low-density denote high- and low-density states inside the empirical training support; Near-, Mid-, and Far-Tail denote states lying progressively farther outside the empirical support boundary, that is, the support-violating states of Sec. 3.2. For OMY and Franka, v is the normalized distance from the training-state bounding box, so $v > 0$

marks a state outside it. For Square and Can, tail severity is controlled by progressively increasing object translation and yaw perturbations from the nominal reset state. Each benchmark is evaluated on 1,000 episodes, 200 per region; all reported success rates and 95% CIs use this protocol.

Success-bank construction and data isolation. For each benchmark, the success bank is constructed exclusively from successful executions generated by the frozen base policy under the nominal in-support distribution. These successful trajectories are collected before evaluation and contain no states from the low-density or tail regions. Their initial conditions are sampled independently of all evaluation initial states. The success bank, calibration set, and evaluation set are mutually disjoint. Retrieval and Residual are provided with exactly the same successful-trajectory data budget as TailGuard, so differences in recovery performance cannot be attributed to additional recovery data or access to low-density or tail-region evaluation trajectories.

4.2 Dynamics-Shaped Outcome Diversity

Eq. (2) predicts that equal-magnitude action perturbations can induce very different physical changes. We evaluate this dynamics-sensitivity hypothesis on four benchmarks: Push-T, Franka Red Cup, Square, and Can. For each locked state, we construct the local Jacobian J_H, extract its high- and low-gain singular directions, and apply norm-matched perturbations along both signs. We then compare Jacobian-predicted gain with the physical gain realized by the environment. Push-T uses a frozen learned dynamics model, and the other three use simulator finite differences.

Table 3: Evaluation of Jacobian-predicted gain across four benchmarks.

Benchmark	Jacobian source	States (clusters)	Gain Spearman	High>low
Push-T (Chi et al., 2025)	learned dynamics	200 (200)	0.660	0.870
Franka Red Cup (Park, 2025)	simulator FD	107 (20)	0.642	0.917
RoboMimic Square (Mandlekar et al., 2021)	simulator FD	152 (38)	0.894	1.000
RoboMimic Can (Mandlekar et al., 2021)	simulator FD	300 (75)	0.908	1.000

Table 3 shows consistent agreement between predicted and realized gain. Spearman correlation ranges from 0.642 to 0.908, while high-gain directions outperform low-gain directions in 87.0–100% of matched comparisons. On Can, action-space distance itself is nearly uncorrelated with outcome-space distance ($\rho = -0.037$), whereas Jacobian-predicted gain correlates strongly with realized gain. The Jacobian therefore provides a reliable local ranking signal; per-direction distributions and component-wise win rates are reported in Appendix C (Figure 5).

Action-space diversity therefore does not faithfully represent usable control support.

4.3 Support Limits of Candidate Sampling

We test whether enlarging the candidate set alone can overcome this limitation. For each benchmark, we compare single-sample execution (Base-K1) with two uses of the same $K = 8$ frozen-policy samples: Uniform-K8 executes one drawn uniformly at random, and Score-K8 uses the same dynamics-conditioned departure score $S(\Psi(x_t, u_i))$, trigger, and success bank as TailGuard to rank them and executes the best-ranked one, but does not optimize actions through gradients (Appendix B.10). We also measure the two limitations of Sec. 3.2: *no support*, the fraction of evaluated states whose pool contains no useful candidate, and *missed support*, the fraction of supported states in which an available useful candidate is not converted into successful continuation. A candidate u is useful if $F_H(x, u) \in \mathcal{Y}_{\text{succ}}(x)$ (Eq. (4)), with membership determined by the empirical support of the success bank; this is an H-step continuation criterion rather than full-episode success.

Both limitations appear on every recovery benchmark (Table 4): 22.4–41.2% of evaluated states remain unsupported, and even when useful candidates exist, 15.3–24.5% of supported states fail to

Table 4: Candidate-sampling support analysis. Small brackets denote 95% CI.

Benchmark	Base-K1 success	Uniform-K8 success	Score-K8 success	No support ↓	Missed support ↓
OMY	42.4% [39.4,45.5]	44.1% [41.0,47.2]	51.8% [48.7,54.9]	38.5% [35.5,41.6]	21.7% [19.2,24.5]
Franka	45.8% [42.7,48.9]	48.3% [45.1,51.4]	56.5% [53.4,59.5]	41.2% [38.2,44.3]	24.5% [21.8,27.4]
Square	42.5% [39.5,45.6]	45.6% [42.5,48.7]	54.8% [51.7,57.9]	35.8% [32.9,38.9]	19.6% [17.1,22.3]
Can	65.8% [62.8,68.7]	68.7% [65.8,71.5]	64.1% [61.1,67.0]	22.4% [19.9,25.1]	15.3% [13.1,17.8]
Mean	49.1%	51.7%	56.8%	34.5%	20.3%

convert them into successful continuation. A random candidate from the larger pool changes success marginally (Uniform-K8: 51.7% vs. 49.1%), and departure-score ranking closes only part of the gap (Score-K8: 56.8%). On Can, ranking does not help (64.1% vs. 65.8%): action distance there barely predicts outcome distance ($\rho = -0.037$, Sec. 4.2), so the dynamics compress differences among candidates, whereas gradient correction follows high-gain directions. Increasing the finite candidate budget alone thus provides limited improvement, suggesting that candidate selection without action correction is insufficient in low-useful-support states; TailGuard's dynamics-guided correction, which can leave the sampled set, reaches 70.4% from a single proposal (Table 5). The same pattern holds on LIBERO for diffusion and flow-matching policies (Appendix D).

4.4 Success-Manifold Recovery and Compatibility-Gated Handoff

We now evaluate whether TailGuard restores closed-loop control by redirecting execution toward the success manifold while keeping the base policy frozen. Table 5 compares it with the base policy, with Retrieval and Residual under the same successful-trajectory budget (Appendix B.10), and with three recent diffusion-policy recovery methods, REACH (Ke et al., 2026), DynaGuide (Du & Song, 2026), and LPB (Sun & Song, 2026), all on the same frozen base policies and protocol.

Table 5: Closed-loop success rates and recovery comparison. Small brackets denote 95% CI; TailGuard's gain over Base is significant on every benchmark ($p < 0.001$). REACH, DynaGuide, and LPB operate on diffusion policies and are not applied to the SmolVLA-based OMY benchmark (N/A); † marks means over the remaining three benchmarks.

Method	OMY	Franka	Square	Can	Mean
Base	42.4% [39.4,45.5]	45.8% [42.7,48.9]	42.5% [39.5,45.6]	65.8% [62.8,68.7]	49.1%
Retrieval	42.5% [39.5,45.6]	45.4% [42.3,48.5]	43.6% [40.6,46.7]	78.2% [75.5,80.6]	52.4%
Residual	47.1% [44.0,50.2]	49.7% [46.6,52.8]	42.4% [39.4,45.5]	65.0% [62.0,67.9]	51.1%
REACH (Ke et al., 2026)	N/A	68.5% [65.6,71.3]	58.4% [55.3,61.4]	79.6% [77.0,82.0]	68.8%†
DynaGuide (Du & Song, 2026)	N/A	66.9% [64.0,69.8]	60.1% [57.0,63.1]	82.7% [80.2,85.0]	69.9%†
LPB (Sun & Song, 2026)	N/A	64.7% [61.7,67.6]	57.2% [54.1,60.3]	81.4% [78.8,83.7]	67.8%†
TailGuard (Ours)	**58.8%** [55.7,61.8]	**75.6%** [72.8,78.2]	**60.2%** [57.1,63.3]	**87.1%** [84.9,89.0]	**70.4%**
Δ vs. Base	+16.4 pp	+29.8 pp	+17.7 pp	+21.3 pp	**+21.3 pp**

TailGuard improves the frozen base policy on all four benchmarks, from 49.1% to 70.4% on average. The largest gain is on Franka Cup (+29.8 pp); OMY Cup improves by 16.4 pp, showing that the mechanism also applies beyond diffusion policies; Can and Square improve by 21.3 pp and 17.7 pp, the varying magnitude reflecting how recoverable the encountered failures are (a breakdown by initial-state region is given in Appendix H). That breakdown also shows where the gain comes from: success in the Head and Low-density regions, which lie inside the empirical training support, is essentially unchanged—the only difference is a 2.5 pp gain on Can Low-density—so the entire improvement is obtained in the Near-, Mid-, and Far-Tail regions outside the support boundary. Among the recovery methods, TailGuard achieves the highest success rate on every benchmark. The strongest competitors, DynaGuide and REACH, average 69.9% and 68.8% over Franka, Square, and Can versus 74.3% for TailGuard; on Franka and Can, TailGuard exceeds the best baseline by 7.1 pp and 4.4 pp, on Square DynaGuide (60.1%) is on par with TailGuard (60.2%), and on OMY,

where the diffusion-specific baselines do not apply, TailGuard is the only method that substantially improves over the base policy (+16.4 pp). The consistent margins indicate that recovery toward the success manifold is most effective when failures remain recoverable but cannot be resolved by the frozen policy alone. Runtime failure detection is compared against Sentinel (Agia et al., 2024) and FIPER (Römer et al., 2026) in Appendix G (AUROC, AUPRC, FPR@95% recall, and failure types).

4.5 Real-Robot Demonstration

We also run TailGuard on a physical Franka arm with a diffusion-policy base policy on a cylinder-stacking task (Figure 3): the base policy fails to grasp the cylinder, whereas TailGuard recovers the grasp and completes the stacking. Three further tasks (Pick, Place, and Screw) are shown in Appendix J; all are single-trial qualitative demonstrations rather than a statistical evaluation, and the full clips are provided in the supplementary video.

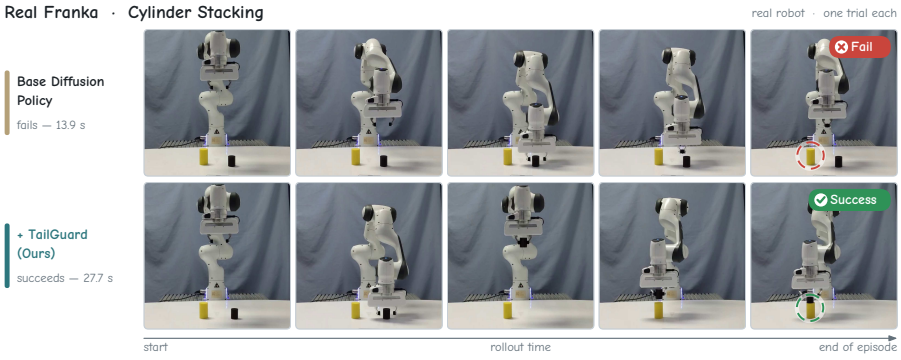

Figure 3: Real-robot cylinder stacking (Franka, Diffusion Policy base). Top row: the base policy fails; bottom row: TailGuard recovers and completes the task. Single trial each.

5 Discussion and Conclusion

Conclusion. Our results show that generative action diversity does not directly imply useful physical control diversity, because robot–environment dynamics can strongly reshape sampled actions. TailGuard addresses this by recovering execution toward the success manifold and returning control only when the base policy becomes compatible again. Experiments on four benchmarks show consistent improvements, obtained entirely on recoverable support-violating tail states rather than on in-support execution, where the frozen policy already succeeds. However, performance depends on the coverage of successful experience and the authority of the recovery controller.

Limitations. TailGuard assumes that failures are physically recoverable; it cannot undo irreversible transitions such as dropped objects or permanent grasp slips, for which the success manifold becomes unreachable and recovery terminates after the budget $T_{\max}$. Moreover, mode switching relies on calibrated thresholds without formal guarantees such as dwell-time bounds or forward invariance; chattering and deadlock are mitigated only by the consistency gate C_{action} on consecutive base proposals, the bounded budget, and the recovery-failure flag (Algorithm 1). Formal switching guarantees and a systematic real-robot evaluation are left for future work.

AI Use Statement

Generative AI tools were used in a limited supporting capacity during the preparation of this work. Their use was restricted to auxiliary tasks that did not contribute to the scientific content, methodology, experimental design, data generation, theoretical analysis, or research claims of the paper. In particular, generative AI was not used to formulate the research questions, develop the proposed method, derive the theoretical results, generate synthetic experimental data, conduct literature discovery, or produce experimental results.

The authors independently designed and implemented all algorithms, experiments, evaluation protocols, and analyses reported in this paper. All numerical results, figures, tables, and scientific conclusions were obtained from the authors' own implementations and experimental pipelines. Any AI-assisted output used during manuscript preparation was manually reviewed and, where applicable, checked against the underlying code, experimental results, and cited literature. No AI-generated scientific claim or reference was included without independent verification.

The authors take full responsibility for the accuracy, originality, and final content of the manuscript, including any material produced with the assistance of generative AI tools.

Reproducibility Statement

We provide the details needed to reproduce our analysis and experiments throughout the main paper and appendices. The formulation of dynamics-shaped outcome diversity and candidate-sampling support is given in Sections 3.1–3.2, while the construction of the success-manifold model, recovery objective, and calibration procedure is described in Section 3.3. Experimental protocols and evaluation settings are reported in Section 4. Detailed implementation, calibration, hyperparameters, ablations, and additional evaluation protocols are provided in Appendices B–J. Code, configurations, and evaluation scripts will be released publicly upon publication.

Ethics Statement

This study primarily focuses on simulation-based robotic manipulation and a closed-loop mechanism for recovery and control handoff. It does not involve human subjects or use private data. Beyond the risks commonly associated with robotics and robot learning research, we do not identify any additional potential harmful applications arising from this work.

References

Alessandro Abate, Maria Prandini, John Lygeros, and Shankar Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734, 2008. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2008.03.027. URL https://www.sciencedirect.com/science/article/pii/S0005109808002677.

Christopher Agia, Rohan Sinha, Jingyun Yang, Zi ang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress, 2024. URL https://arxiv.org/abs/2410.04640.

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

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

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)*, pp. 15657–15664. IEEE, 2025.

Maximilian Du and Shuran Song. Dynaguide: Steering diffusion polices with active dynamic guidance. *Advances in Neural Information Processing Systems*, 38:44192–44221, 2026.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

François Robert Hogan and Alberto Rodriguez. Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics. In *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*, pp. 800–815. Springer, 2020.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Zundong Ke, Junlin Chen, Jiayi Zhu, Kuanhao Xia, Boyi Zhao, and Jiayuan Gu. Reach: Explicit recovery behavior for diffusion policies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 38498–38508, 2026.

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

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

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

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

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

JeongEun Park. LeRobot MuJoCo Tutorial: Minimal examples for collecting demonstration data and training vision-language-action models on custom datasets. GitHub repository, 2025. URL https://github.com/jeongeun980906/lerobot-mujoco-tutorial.

Ralf Römer, Adrian Kobras, Luca Worbis, and Angela Schoellig. Failure prediction at runtime for generative robot policies. *Advances in Neural Information Processing Systems*, 38:7631–7670, 2026.

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

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

Zhanyi Sun and Shuran Song. Latent policy barrier: Learning robust visuomotor policies by staying in-distribution. *Advances in Neural Information Processing Systems*, 38:174280–174305, 2026.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40 (2):344–357, 2017.

Chen Xu, Tony Khuong Nguyen, Emma Dixon, Christopher Rodriguez, Patrick Miller, Robert Lee, Paarth Shah, Rares Ambrus, Haruki Nishimura, and Masha Itkina. Can we detect failures without failure data? uncertainty-aware runtime failure detection for imitation learning policies. *arXiv preprint arXiv:2503.08558*, 2025.

Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. Flowpolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 14754–14762, 2025.

Ganlong Zhao, Zijia Tang, Xingping Chen, Zhanghui Kuang, Ye Tian, and Guanbin Li. Flare: A failure-aware framework for autonomous correction and recovery in visual-language robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22391–22401, 2026.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Abhiram Maddukuri, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

A Detailed Positioning in Related Work

Our work is complementary to recent advances in generative behavior cloning and runtime recovery. Generative policies, including diffusion, flow-matching, and vision-language-action policies, primarily improve policy expressiveness by modeling multimodal action distributions. In contrast, our focus is not on designing a new generative architecture or training objective, but on characterizing the physical control support induced by an already trained policy after its sampled actions are transformed through the robot–environment dynamics. This distinction is important because substantial diversity in action space may still correspond to limited diversity in physically realizable outcomes.

Our analysis is also related to dynamics-aware planning, model-predictive control, and reachability analysis. However, we do not learn a new dynamics-conditioned planner, optimize long-horizon trajectories online, or solve a global reachability problem. Instead, local dynamics are used to diagnose how perturbations of actions sampled from a frozen policy are amplified or compressed in outcome space. The resulting local sensitivity is treated as a short-horizon characterization rather than a surrogate for long-horizon task value.

TailGuard is most closely related to runtime failure detection and recovery methods. Existing monitoring approaches primarily address when a learned policy becomes unreliable, whereas recovery-oriented methods introduce recovery trajectories, modified policy generation, or dedicated retry/reset behaviors. TailGuard differs in three respects. First, the base generative policy remains frozen, and recovery is performed by an external controller. Second, the recovery target is defined by a task-conditioned success manifold estimated from previously successful physical executions, rather than by a predefined reset state. Third, TailGuard explicitly separates recovery from policy handoff: control is returned only when the base-policy proposal re-enters the success manifold and satisfies action continuity. Thus, TailGuard occupies a complementary runtime-control layer between generative policy execution and closed-loop recovery.

B TailGuard Implementation Details

This appendix provides implementation details omitted from Sec. 3.3, including the learned dynamics model, local manifold estimation, the approximations underlying the recovery objective, handoff calibration, hyperparameters, and the complete runtime procedure, as well as the specification of the Retrieval and Residual recovery baselines compared in Sec. 4.4.

B.1 Learned Short-Horizon Dynamics

Observation and visual feature. The runtime observation is $x_t = (I_t^{(1)}, I_t^{(2)}, s_t)$, where $I_t^{(1)}$ and $I_t^{(2)}$ are the two camera views and s_t is the low-dimensional observation vector consumed by the base policy (Table 1): the 7-D and 8-D proprioceptive state on OMY and Franka Red Cup, and the 23-D RoboMimic low-dimensional observation on Square and Can, which in the standard low-dim setting already contains the object pose. Both views are passed through the same frozen DINOv2-B

encoder and their CLS tokens are averaged rather than concatenated,

$$z_t^{\text{vis}} = \frac{1}{2}\left(E_{\text{DINO}}\left(I_t^{(1)}\right) + E_{\text{DINO}}\left(I_t^{(2)}\right)\right) \in \mathbb{R}^{768}, \tag{17}$$

and are then standardized and projected to 32 dimensions,

$$\bar{z}_t^{\text{vis}} = P_{\text{vis}}\left(\text{Std}_{\text{vis}}\left(z_t^{\text{vis}}\right)\right) \in \mathbb{R}^{32}, \tag{18}$$

using statistics fitted once on the raw success-bank rollouts.

Input to the dynamics model. The dynamics model is conditioned on exactly the same 32-dimensional frozen visual feature that forms the visual branch of Ψ:

$$f_\phi\left(\left[\bar{z}_t^{\text{vis}}; s_t; \text{vec}\left(u[0{:}H]\right)\right]\right), \qquad d_{\text{in}} = 32 + \dim(s_t) + H \cdot \dim(a), \tag{19}$$

giving $d_{\text{in}} = 109$ on OMY, 104 on Franka, and 135 on Square and Can (Table 6). Scene information therefore reaches f_ϕ only through $\bar{z}_t^{\text{vis}}$: the MLP never processes raw pixels, and at runtime TailGuard reads no simulator state beyond the observation the base policy itself receives. On Square and Can, whose base policies consume only the 23-D observation, the rendered views entering Eq. (17) are an auxiliary observation that the base policy does not use; the Retrieval and Residual baselines are built on the same representation and therefore receive identical inputs (Appendix B.10).

Architecture, prediction target, and training. We implement f_ϕ as a three-layer MLP with hidden dimension $d_h = 256$ and ReLU activations. For benchmark b it predicts the next H_b steps $\hat{y}_{t+1:t+H_b}$, where each step collects

$$\hat{y}_{t+h} = \left[p_3^{\text{ee}}; r_6^{\text{ee}}; g_1; p_3^{\text{obj}}; r_6^{\text{obj}}; v_3^{\text{obj}}; d_1^{\text{contact}}; c_1^{\text{contact}}\right] \in \mathbb{R}^{24}, \tag{20}$$

namely the end-effector position and 6-D rotation, the gripper state, the object position, 6-D rotation, and linear velocity, and the contact distance and contact indicator, where each subscript gives the dimension of that component. When the policy action chunk is longer than H_b, only its first H_b steps are used to construct the dynamics feature. f_ϕ is trained offline on transition tuples collected from the training demonstrations; the physical quantities in $\hat{y}$ are read from the demonstration logs and are used only as supervision targets, never as inputs at runtime. The DINOv2 encoder, Std_{vis}, and P_{vis} are held fixed while f_ϕ is trained, so the prediction loss updates the MLP weights alone; with the input dimensions above this gives 0.14M trainable parameters per benchmark (Appendix I). After training, f_ϕ is frozen and no simulator rollout is required during TailGuard execution.

Dynamics feature and final representation. The predicted trajectory is flattened into $r_t^{\text{dyn}} = \text{vec}(\hat{y}_{t+1:t+H}) \in \mathbb{R}^{24H}$ and mapped to a benchmark-independent dimension,

$$z^{\text{dyn}}(x_t, u) = P_{192}^{\text{dyn}}\, r_t^{\text{dyn}} \in \mathbb{R}^{192}. \tag{21}$$

For $H = 8$ the flattened prediction is already 192-dimensional and P_{192}^{dyn} is the identity; on OMY ($H = 10$, 240 dimensions) it is a non-trainable PCA projection. The dynamics block is then standardized and projected to 32 dimensions in the same way as the visual block,

$$\bar{z}^{\text{dyn}} = P_{\text{dyn}}\left(\text{Std}_{\text{dyn}}\left(z^{\text{dyn}}\right)\right) \in \mathbb{R}^{32}, \tag{22}$$

and the concatenation $\tilde{z} = [\bar{z}^{\text{vis}}; \bar{z}^{\text{dyn}}]$ gives the 64-dimensional representation used throughout the manifold estimator (Eq. (6)). The three fits are performed in this order and none of them is updated afterwards: the visual transform on the raw success-bank rollouts, then f_ϕ with that transform held fixed, and finally P_{192}^{dyn}, Std_{dyn}, and P_{dyn} on the resulting predictions.

B.2 Local Manifold Estimation

For each benchmark, the reference success bank contains $N_{\text{succ}} = 200$ successful rollouts generated autonomously by the frozen base policy. The rollouts are collected only from the nominal in-support distribution and contain no states from the low-density or tail regions. No evaluation initial condition is used when constructing the bank. The reference bank is also disjoint from the $N_{\text{cal}} = 50$ successful rollouts used for threshold calibration and from all evaluation trajectories. Within each

rollout, states are sampled every five environment steps, and five action chunks are drawn from the frozen base policy at each sampled state, which yields a full reference set of up to roughly 72k state–action entries per task. To bound memory and nearest-neighbor query cost, this set is then uniformly subsampled at random to a fixed capacity of $N_{\text{bank}} = 5{,}000$ entries per task, using a fixed random seed; the subsampling is independent of all evaluation trajectories. Each stored entry is the 64-dimensional representation $\tilde{z}$, so the bank amounts to $5{,}000 \times 64 \approx 0.32$M floating-point values (Appendix I); Appendix F.2 shows that this capacity is sufficient.

For every query $\tilde{z}$, we retrieve the $m = 50$ nearest entries under Euclidean distance in the corresponding task-specific success bank. Let the retrieved samples be $\{\tilde{z}_i\}_{i=1}^{m}$. Their empirical mean is

$$\mu = \frac{1}{m} \sum_{i=1}^{m} \tilde{z}_i. \tag{23}$$

The local covariance is estimated using Ledoit–Wolf shrinkage and is additionally regularized as

$$\widetilde{\Sigma} = \Sigma_{\text{LW}} + 10^{-8} \frac{\operatorname{tr}(\Sigma_{\text{LW}})}{d} I, \qquad d = 64. \tag{24}$$

We use the Moore–Penrose pseudoinverse $\widetilde{\Sigma}^{\dagger}$ when evaluating the Mahalanobis score.

Since $m = 50 < d = 64$, the unregularized empirical covariance is rank deficient. The shrinkage term improves conditioning and ensures a stable local metric. In practice, the regularized matrices are numerically full rank in the evaluated states.

The resulting departure score is

$$S(\tilde{z}) = \sqrt{\max\left[(\tilde{z} - \mu)^{\top} \widetilde{\Sigma}^{\dagger} (\tilde{z} - \mu), 0\right]}. \tag{25}$$

B.3 Calibration Protocol

For each benchmark, the deviation threshold is calibrated using $N_{\text{cal}} = 50$ successful rollouts that are disjoint from both the reference bank and the evaluation set. All sampled probes from these rollouts are pooled into

$$\mathcal{C}_{\text{succ}} = \left\{ S(\tilde{z}_j^{\text{cal}}) \right\}_{j=1}^{N_{\text{probe}}}, \tag{26}$$

and the threshold is

$$\tau_{\text{dev}} = Q_{0.95}\left(\mathcal{C}_{\text{succ}}\right). \tag{27}$$

The action-continuity threshold is calibrated independently for each benchmark using the same statistic as Eq. (15). Specifically,

$$\varepsilon_a = Q_{0.9}\left(\mathcal{C}_{\text{action}}\right), \tag{28}$$

where $\mathcal{C}_{\text{action}}$ contains continuity scores measured on the exact temporally adjacent action-chunk pairs used in the implementation. No evaluation trajectory is used to determine either threshold.

B.4 Frozen-Neighborhood Recovery Approximation

The success-manifold estimator in Sec. 3.3 is query-dependent. For a general query $\tilde{z}$, both the retrieved neighbor set and its local statistics depend on $\tilde{z}$. Consequently,

$$-\lambda \log \rho_{\text{succ}}\left(\Psi(x_t, u)\right) \tag{29}$$

is not globally quadratic in u.

At each recovery step, we instead retrieve the neighborhood once at the base-policy proposal:

$$\mathcal{N}_t = \text{kNN}\left(\Psi(x_t, u_t^{\pi}), \mathcal{B}, m\right), \tag{30}$$

and compute

$$\mu_t = \mu(\mathcal{N}_t), \qquad \widetilde{\Sigma}_t = \widetilde{\Sigma}(\mathcal{N}_t). \tag{31}$$

These quantities are held fixed during all J inner optimization steps. Conditioned on this frozen neighborhood,

$$-\lambda \log \rho_{\text{succ}} \left(\Psi(x_t, u) \mid \mathcal{N}_t\right) = \frac{\lambda}{2} \left(\Psi(x_t, u) - \mu_t\right)^\top \widetilde{\Sigma}_t^\dagger \left(\Psi(x_t, u) - \mu_t\right) + C. \tag{32}$$

Thus, the implemented objective is a local approximation to the query-dependent success manifold. The approximation is exact with respect to the local manifold estimator as long as the relevant nearest-neighbor set remains unchanged over the displacement induced by the inner optimization. We use the frozen-neighborhood construction to avoid discontinuous neighbor changes during gradient-based refinement and to keep recovery local to the original base-policy proposal.

B.5 Relation to the Tilted Distribution

The conceptual target introduced in Eq. (11) is

$$p_{\text{TG}}(u \mid x_t) \propto p_\pi(u \mid x_t) \rho_{\text{succ}} \left(\Psi(x_t, u)\right)^\lambda. \tag{33}$$

If its normalized density were directly available, its MAP solution would minimize

$$\mathcal{L}_{\text{MAP}}(u) = -\log p_\pi(u \mid x_t) - \lambda \log \rho_{\text{succ}} \left(\Psi(x_t, u)\right). \tag{34}$$

Our implemented objective does not optimize Eq. (34). Instead, it replaces the generally unavailable base-policy log-density term with a quadratic penalty around one sampled proposal:

$$-\log p_\pi(u \mid x_t) \;\rightsquigarrow\; \frac{1}{2} \|u - u_t^\pi\|_W^2. \tag{35}$$

The two prior terms would agree up to an additive constant only under a local approximation of the form

$$-\log p_\pi(u \mid x_t) \approx \frac{1}{2} \|u - u_t^\pi\|_W^2 + C, \tag{36}$$

which corresponds to a locally Gaussian approximation when W matches the local precision and u_t^π lies near a local mode.

In our implementation, actions are normalized per dimension to $[-1, 1]$ and we use

$$W = I. \tag{37}$$

We therefore do not attempt to estimate the local curvature of the frozen generative policy. The quadratic term is used only as an isotropic trust region that limits the magnitude of the correction. Accordingly, TailGuard should be interpreted as a sample-anchored local correction mechanism rather than an exact sampler or MAP solver for Eq. (11).

B.6 Gradient Optimization Details

For each recovery step, optimization begins at

$$u^{(0)} = u_t^\pi. \tag{38}$$

We perform $J = 10$ projected-gradient steps with $\eta = 0.01$:

$$u^{(j+1)} = \Pi_{\mathcal{U}} \left[u^{(j)} - \eta \nabla_u \mathcal{L}_{\text{rec}} \left(u^{(j)}\right)\right]. \tag{39}$$

The local statistics μ_t and $\widetilde{\Sigma}_t$ remain fixed throughout the J updates.

The visual component $\bar{z}^{\text{vis}}$ is constant with respect to u during optimization, both as the visual block of Ψ and as a conditioning input of f_ϕ (Eq. (19)). Gradients therefore reach the action only through the dependence of f_ϕ on $u[0{:}H]$ and through the dynamics projection. The full covariance $\widetilde{\Sigma}_t^\dagger$ is nevertheless retained in the objective.

We use

$$\lambda = 0.75, \qquad J = 10, \qquad \eta = 0.01 \tag{40}$$

for all benchmarks. Sensitivity to λ, J, and η is reported in the corresponding ablation experiments.

B.7 Compatibility-Gated Handoff Details

Let Δ_{exec} denote the number of low-level actions executed between two consecutive policy queries. For an action chunk of length L, the continuity window k satisfies

$$\Delta_{\text{exec}} + k \leq L. \tag{41}$$

The continuity statistic is

$$C_{\text{action}} = \frac{1}{\sqrt{k}} \left\| u_t^{\pi}[0{:}k] - u_{t-1}^{\text{rec}}[\Delta_{\text{exec}} : \Delta_{\text{exec}} + k] \right\|_2 . \tag{42}$$

Both Δ_{exec} and k depend on the action-chunk structure of the underlying policy and are therefore specified per benchmark.

Relation to the recovery horizon. The dynamics model scores only the first H actions of a chunk (Appendix B.1), and $H = \Delta_{\text{exec}}$ on every benchmark (Table 6): f_ϕ predicts exactly the segment that will be executed before the policy is queried again. Consequently, the manifold term of Eq. (12) has zero gradient with respect to $u[\Delta_{\text{exec}}{:}L]$, and because the proximal term vanishes at the initialization $u^{(0)} = u_t^{\pi}$, these entries are never updated (the projection $\Pi_{\mathcal{U}}$ leaves them unchanged as well, since the base policy's outputs already lie in $\mathcal{U}$): $u_t^{\text{rec}}[\Delta_{\text{exec}}{:}L] = u_t^{\pi}[\Delta_{\text{exec}}{:}L]$. The window entering C_{action} is thus the continuation that the base policy itself planned at the previous step, and the statistic tests whether the base policy, re-queried from the recovered state, still agrees with that plan. This is deliberate: a handoff is safe once the base policy's consecutive proposals have become consistent again, whereas the executed transition is already kept close to the base proposal by the trust-region term. Extending the horizon to $H \geq \Delta_{\text{exec}} + k$ (e.g., $H = 16$) would make f_ϕ predict actions that are never executed before re-planning, with the accompanying growth of prediction error over the longer horizon, and would require retraining the dynamics model and re-running all TailGuard experiments on these benchmarks; we therefore keep $H = \Delta_{\text{exec}}$.

Handoff occurs only when both

$$S\left(\Psi(x_t, u_t^{\pi})\right) \leq \tau_{\text{dev}} \tag{43}$$

and

$$C_{\text{action}} \leq \varepsilon_a \tag{44}$$

hold simultaneously.

If these conditions are not jointly satisfied within $T_{\max}$ recovery control steps, TailGuard terminates the recovery attempt and returns control to the frozen policy. A control step here is one policy query, so $T_{\max}$ is counted in policy queries rather than environment steps and acts as a loose safety cap: on OMY the 240-step episode horizon spans only 24 control steps, so an OMY episode always ends before the budget is exhausted. A recovery-failure flag then prevents recovery from being re-triggered until the base-policy proposal re-enters the success manifold on its own, so the system cannot alternate indefinitely between the two modes (Algorithm 1). The episode continues normally and is evaluated using the same task-success criterion.

Figure 4 illustrates the complete TailGuard mechanism, including on-manifold execution, manifold departure, and dynamics-guided recovery followed by compatibility-gated handoff. The robot frames are actual execution frames from the Franka Red Cup MuJoCo benchmark.

B.8 TailGuard Hyperparameters

Table 6 lists all TailGuard hyperparameters, separating values shared across benchmarks from benchmark-specific values.

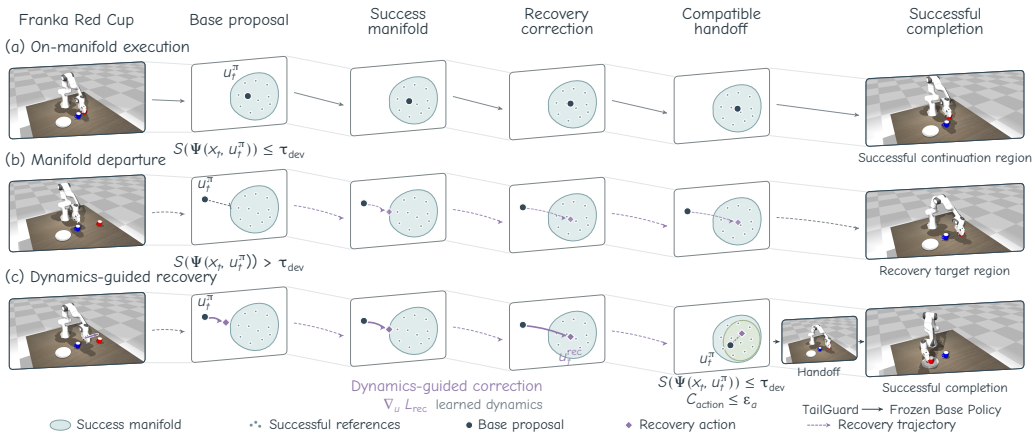

Figure 4: Detailed mechanism of TailGuard. Each row traces one execution from the base proposal through the success manifold, recovery correction, and compatible handoff: **(a)** on-manifold execution, **(b)** manifold departure, and **(c)** dynamics-guided recovery followed by compatibility-gated handoff. Robot images are actual execution frames from the Franka Red Cup MuJoCo benchmark.

Table 6: TailGuard hyperparameters. Values above the horizontal rule are shared across benchmarks; values below depend on the benchmark or the base policy's action-chunk structure.

Shared		Value			
Neighborhood size	m	50			
Success-bank size	N_{succ}	200 rollouts			
Bank capacity	N_{bank}	5,000 entries (uniform subsampling)			
Calibration rollouts	N_{cal}	50			
Representation dimension	d	64 (32 visual + 32 dynamics)			
Dynamics MLP width	d_h	256			
Guidance strength	λ	0.75			
Gradient steps	J	10			
Step size	η	0.01			
Recovery budget	$T_{\max}$	40 control steps			
Deviation quantile	—	$Q_{0.95}$			
Continuity quantile	—	$Q_{0.9}$			
Per benchmark		**OMY**	**Franka**	**Square**	**Can**
Dynamics horizon	H	10	8	8	8
Execution stride	Δ_{exec}	10	8	8	8
Dynamics input dimension	d_{in}	109	104	135	135
Continuity window	k	10	8	8	8
Continuity threshold	ε_a	0.0420	0.0415	0.0578	0.0575
Deviation threshold	τ_{dev}	12.8682	10.8138	14.8926	9.5682

B.9 Runtime Procedure

Algorithm 1 summarizes the complete runtime procedure.

Algorithm 1 TailGuard runtime loop

Require: frozen policy π, dynamics model f_ϕ, success bank $\mathcal{B}$, threshold τ_{dev}, continuity threshold ε_a, recovery budget $T_{\max}$

MODE $\leftarrow$ BASE
$c \leftarrow 0$
HASPREVREC $\leftarrow$ FALSE
RECFAILED $\leftarrow$ FALSE
for $t = 0, 1, \ldots$ **do**
  $u_t^\pi \sim p_\pi(\cdot \mid x_t)$
  $\tilde{z}_t^\pi \leftarrow \Psi(x_t, u_t^\pi)$
  $(\mu_t, \widetilde{\Sigma}_t) \leftarrow \text{LOCALSTATS}(\mathcal{B}, \tilde{z}_t^\pi, m)$
  $S_t \leftarrow S(\tilde{z}_t^\pi)$
  if MODE $=$ BASE **then**
    if $S_t \le \tau_{\text{dev}}$ **then**
      RECFAILED $\leftarrow$ FALSE
      execute u_t^π
      continue
    else if RECFAILED **then**
      execute u_t^π ▷ no re-trigger after a failed attempt
      continue
    else
      MODE $\leftarrow$ RECOVER
      $c \leftarrow 0$
      HASPREVREC $\leftarrow$ FALSE
    end if
  end if
  if HASPREVREC **then**
    $C_t \leftarrow C_{\text{action}}(u_t^\pi, u_{t-1}^{\text{rec}})$
    if $S_t \le \tau_{\text{dev}}$ **and** $C_t \le \varepsilon_a$ **then**
      MODE $\leftarrow$ BASE
      execute u_t^π
      continue
    end if
  end if
  if $c \ge T_{\max}$ **then**
    MODE $\leftarrow$ BASE
    RECFAILED $\leftarrow$ TRUE
    execute u_t^π
    continue
  end if
  $u^{(0)} \leftarrow u_t^\pi$
  for $j = 0, \ldots, J-1$ **do**
    $u^{(j+1)} \leftarrow \Pi_{\mathcal{U}}\left[u^{(j)} - \eta \nabla_u \mathcal{L}_{\text{rec}}(u^{(j)})\right]$ ▷ $\mu_t, \widetilde{\Sigma}_t$ fixed
  end for
  $u_t^{\text{rec}} \leftarrow u^{(J)}$
  execute u_t^{rec}
  HASPREVREC $\leftarrow$ TRUE
  $c \leftarrow c + 1$
end for

B.10 RECOVERY BASELINES

This section specifies the recovery baselines: Retrieval and Residual, compared in Table 5, and Score-K8, reported with the candidate-sampling analysis in Table 4. All three are drop-in replacements for the recovery step of TailGuard: they share the representation, the departure-score detector, the calibrated trigger threshold, and the successful-rollout data of TailGuard, and differ only in how the executed chunk is produced after a trigger and in how control is returned to the frozen policy.

Retrieval tests whether reusing a stored successful chunk, without adapting it to the current state, is sufficient for recovery; Residual tests a learned, state-conditioned correction that is applied without dynamics guidance, magnitude control, or gating; Score-K8 tests whether selecting among the frozen policy's own samples by the departure score is sufficient. None of the baselines uses the dynamics-guided optimization of Eq. (12), the compatibility gate of Eq. (16), the recovery budget $T_{\max}$, or the recovery-failure flag of Algorithm 1. Table 7 summarizes Retrieval, Residual, and TailGuard side by side and Table 8 lists the baseline hyperparameters.

Shared representation, trigger, and data. At every policy query, the frozen policy proposes u_t^π, the representation $\tilde{z}_t^\pi = \Psi(x_t, u_t^\pi) \in \mathbb{R}^{64}$ is computed as in Appendix B.1 from the same inputs, including the auxiliary rendered views on Square and Can, and the departure score $S(\tilde{z}_t^\pi)$ is evaluated against the task-specific success bank $\mathcal{B}$ exactly as in Algorithm 1. A baseline is triggered by the detection condition of Eq. (10), $S(\tilde{z}_t^\pi) > \tau_{\text{dev}}$, using the per-benchmark thresholds of Table 6; otherwise u_t^π is executed unchanged. All baselines use the same $N_{\text{succ}} = 200$ successful base-policy rollouts as the TailGuard bank (Appendix B.2), collected under the nominal in-support distribution, and none uses low-density or tail-region trajectories for construction, training, or hyperparameter selection.

Retrieval. *Representation and distance.* Retrieval operates directly in the 64-dimensional representation space of Eq. (6). Each entry i of the task-specific bank $\mathcal{B}$ stores, in addition to its representation $\tilde{z}_i$, the executed segment of the base-policy chunk u_i from which $\tilde{z}_i$ was computed. The bank is identical to the TailGuard bank: the same sampled states, the same base-policy chunks, and the same uniformly subsampled $N_{\text{bank}} = 5{,}000$ entries per task. When triggered, the baseline performs a single Euclidean nearest-neighbor query,

$$i^\star = \arg\min_{i \in \mathcal{B}} \|\tilde{z}_t^\pi - \tilde{z}_i\|_2 , \tag{45}$$

and selects the stored chunk $u_{i^\star}$. No local statistics are estimated, no optimization is performed, and the retrieved chunk is not adapted to the current state.

Execution. The retrieved chunk is executed in its entirety and open loop, i.e., over its full chunk length: 50 joint-space commands on OMY, 8 joint-space commands on Franka, and 8 absolute OSC end-effector pose targets on Square and Can. The frozen policy is not queried and the departure score is not evaluated while the chunk is being executed.

Handoff. After the last command of $u_{i^\star}$ has been executed, control is returned to the frozen policy unconditionally: neither the manifold condition of Eq. (14) nor the action-continuity condition of Eq. (15) is checked, and the baseline does not verify that the departure score has decreased. The detector is simply evaluated again at the next policy query.

Data and hyperparameters. Retrieval has no learned components and no tunable hyperparameters; its only inputs are the shared bank, the shared representation, and the shared detector threshold.

Residual. *Representation and network.* Residual adds a learned, state-conditioned correction to the base-policy chunk. Let $u_t^\pi[0] \in \mathbb{R}^A$ denote the first low-level action of the proposal, where A is the per-step action dimension ($A = 7$ on OMY, 8 on Franka, and 10 on Square and Can), and let $d = 64$ be the representation dimension. The correction network $g : \mathbb{R}^d \times \mathbb{R}^A \to \mathbb{R}^A$ is a two-layer ReLU MLP with 256 hidden units per layer applied to the concatenation $[\tilde{z};\, u[0]]$:

$$g = \text{Linear}(256, A) \circ \text{ReLU} \circ \text{Linear}(256, 256) \circ \text{ReLU} \circ \text{Linear}(d{+}A, 256). \tag{46}$$

No gradient is propagated through the dynamics model f_ϕ; the only dynamics information available to g is the dynamics block of $\tilde{z}$.

Training objective. Training pairs are constructed from the same sampled states and base-policy chunks as the bank. For a sampled state with representation $\tilde{z}_i$ and sampled proposal u_i, the regression target is the difference $u_i^\star[0] - u_i[0]$ between the first action of the chunk that was actually executed at that state in the successful rollout, $u_i^\star[0]$, and the first action of the proposal. The network is trained by mean-squared-error regression,

$$\min_g \frac{1}{|\mathcal{D}|} \sum_{(\tilde{z}_i, u_i, u_i^\star) \in \mathcal{D}} \|g(\tilde{z}_i, u_i[0]) - (u_i^\star[0] - u_i[0])\|_2^2 , \tag{47}$$

with AdamW (learning rate 10^{-3}, weight decay 10^{-4}, batch size 64) for 400 epochs and a 20% holdout split. All training hyperparameters are identical for the four benchmarks and were fixed before any closed-loop evaluation; no per-benchmark tuning is performed. On the holdout split, g explains 0.96, 0.73, 0.93, and 0.98 of the variance of the regression target on OMY, Franka, Square, and Can, respectively (Table 8), indicating that the modest closed-loop gains of Residual in Table 5 do not stem from underfitting on in-support data.

Execution. When triggered, the same residual is broadcast to every step of the current chunk,

$$u_t^{\mathrm{res}}[i] = u_t^{\pi}[i] + g(\tilde{z}_t^{\pi}, u_t^{\pi}[0]), \qquad i = 0, \ldots, L-1, \tag{48}$$

where L is the chunk length and $\tilde{z}_t^{\pi}$ is the representation that triggered recovery. The correction is applied as is, without norm clipping, scaling, or gating, and the corrected chunk is executed over its full chunk length L (50 on OMY and 8 on Franka, Square, and Can).

Handoff. As for Retrieval, control is returned to the frozen policy unconditionally once the corrected chunk has been executed; no compatibility check, recovery budget, or re-trigger lockout is applied, and the detector is evaluated again at the next policy query.

Table 7: Side-by-side specification of the recovery baselines and TailGuard. Rows above the rule are shared; per-benchmark values are ordered OMY / Franka / Square / Can.

	Retrieval	Residual	TailGuard
Representation	$\tilde{z} = \Psi(x_t, u_t^{\pi}) \in \mathbb{R}^{64}$ (Eq. (6))		
Trigger	$S(\tilde{z}_t^{\pi}) > \tau_{\mathrm{dev}}$ with the calibrated thresholds of Table 6		
Successful data	same $N_{\mathrm{succ}} = 200$ nominal rollouts; no low-density or tail trajectories		
Recovery action	chunk stored at the Euclidean 1-NN bank entry (Eq. (45))	$u_t^{\pi}[i] + g(\tilde{z}_t^{\pi}, u_t^{\pi}[0])$ for every step i (Eq. (48))	J projected-gradient steps on $\mathcal{L}_{\mathrm{rec}}$ through f_ϕ (Eq. (13))
Learned model	none	MLP g: $d{+}A \to 256 \to 256 \to A$	none beyond the shared f_ϕ
Training objective	—	MSE on $u^\star[0] - u[0]$; AdamW, 400 epochs	—
Local statistics	not used	not used	$\mu_t, \widetilde{\Sigma}_t$ (frozen neighborhood)
Steps per trigger	50/8/8/8, open loop	50/8/8/8	$\Delta_{\mathrm{exec}} = 10/8/8/8$, closed loop
Magnitude control	—	none (no clipping, no gating)	isotropic trust region and $\Pi_{\mathcal{U}}$
Handoff	unconditional after the chunk	unconditional after the chunk	$S \le \tau_{\mathrm{dev}}$ and $C_{\mathrm{action}} \le \varepsilon_a$; budget $T_{\max}$
Tunable parameters	none	optimizer settings only (shared)	λ, J, η, $T_{\max}$ (shared)

Table 8: Recovery baseline hyperparameters (Retrieval has no tunable parameters). Values above the rule are shared across benchmarks.

Shared		Value			
Retrieval: query	—	Euclidean 1-NN on $\tilde{z} \in \mathbb{R}^{64}$ over the 5,000-entry TailGuard bank			
Residual: network	g	2×256 ReLU MLP on $[\tilde{z};\, u[0]]$ (Eq. (46))			
Residual: optimizer	—	AdamW, learning rate 10^{-3}, weight decay 10^{-4}, batch size 64			
Residual: schedule	—	400 epochs, 20% holdout			
Score-K8: candidates	K	8 frozen-policy samples; execute $\arg\min_i S(\Psi(x_t, u_i))$ (Eq. (49))			
Per benchmark		**OMY**	**Franka**	**Square**	**Can**
Action space (dimension)	A	joint (7)	joint (8)	abs. OSC pose (10)	abs. OSC pose (10)
Executed chunk length	—	50	8	8	8
Residual holdout R^2	—	0.96	0.73	0.93	0.98

Score-K8. *Candidates and selection.* When triggered, Score-K8 draws $K = 8$ action chunks $u_1, \ldots, u_8 \sim p_\pi(\cdot \mid x_t)$ from the frozen policy, the same eight candidates used by Uniform-K8 in Table 4, evaluates each representation $\Psi(x_t, u_i)$ and departure score against $\mathcal{B}$ as above, and selects

$$i^\star = \arg\min_{i \in \{1,\ldots,8\}} S(\Psi(x_t, u_i)) \,. \tag{49}$$

Score-K8 thus uses the same dynamics-conditioned departure score as TailGuard for ranking, but does not optimize actions through gradients: f_ϕ enters only through the forward pass that produces the dynamics block of Ψ, and the selected chunk is not modified, so Score-K8 can only execute actions that the frozen policy itself proposed.

Execution. The selected chunk $u_{i^\star}$ is executed for Δ_{exec} low-level steps before the next policy query, the same stride as Base-K1 and Uniform-K8 (Table 6). Uniform-K8 differs only in executing a candidate drawn uniformly at random from the same eight instead of $u_{i^\star}$.

Handoff. As for Retrieval and Residual, control is returned to the frozen policy unconditionally once the chunk has been executed; no compatibility check, recovery budget, or re-trigger lockout is applied, and the detector is evaluated again at the next policy query.

Data and hyperparameters. Score-K8 has no learned components beyond the shared representation; its only parameter is $K = 8$, fixed to match Uniform-K8. It is evaluated with the same 1,000 episodes per benchmark, initial states, random seeds, and frozen base policy as all other methods.

C Additional Dynamics Ablation Results

We provide additional dynamics ablation results to analyze how local action perturbations are transformed into realized environmental changes. The analysis includes per-direction evidence across the four benchmarks, strategy-level correlation evaluation, physical component-wise ablation, and the relationship between action diversity and outcome diversity.

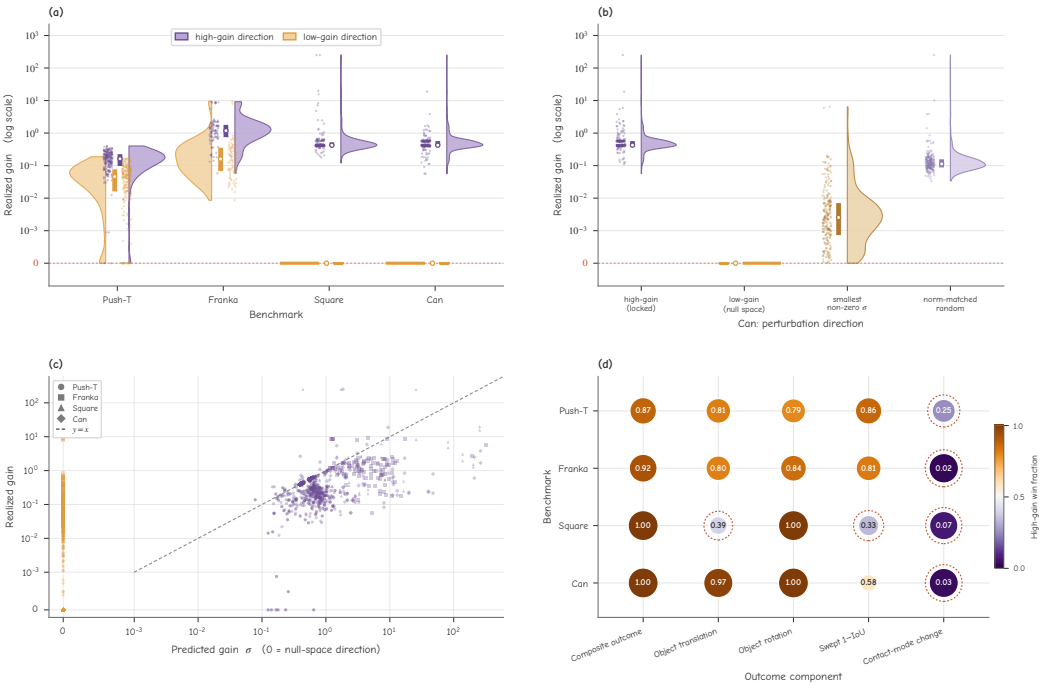

Figure 5: Evidence and theory across four contact-rich benchmarks, based on per-direction records. Purple and orange indicate high- and low-gain directions, respectively. **(a)** Distribution of realized gain. **(b)** Low-gain structure in Can. **(c)** Correlation between predicted and realized gain. **(d)** Pairwise win rates across physical components.

Figure 5 complements Table 3 in Sec. 4.2 with the underlying per-direction records. Panel (a) shows the distribution of realized gain for high- and low-gain directions on each benchmark: high-gain directions realize larger physical changes on all four benchmarks, and on Square and Can the low-gain (null-space) directions produce almost no realized change. Panel (b) isolates this low-gain structure on Can by comparing locked high-gain directions, null-space directions, the smallest non-zero singular direction, and norm-matched random directions, which explains why action-space distance is nearly uncorrelated with outcome-space distance on this benchmark ($\rho = -0.037$). Panel (c) plots Jacobian-predicted gain against realized gain: the ranking is preserved, whereas the magnitude prediction becomes less accurate near contact. Panel (d) decomposes the high-gain win fraction by physical outcome component; the composite-outcome column coincides with the High>low column of Table 3, and the win fraction is consistently high for object rotation and lowest for contact-mode change.

Table 9: Strategy- and suite-level evaluation of dynamics sensitivity prediction.

Strategy	Suite	Spearman ρ	Pearson r	High-low win rate	Action-outcome ρ
DP	All Tasks	0.986	0.978	1.000	0.189
DP	Target	0.976	0.887	0.990	0.169
DP	Object	0.994	0.995	1.000	0.194
DP	Spatial	0.993	0.987	1.000	0.170
FM	All Tasks	0.994	0.992	1.000	0.179
FM	Target	0.950	0.834	0.990	0.170
FM	Object	0.994	0.995	1.000	0.186
FM	Spatial	0.969	0.854	0.990	0.179

Table 9 evaluates whether local dynamics sensitivity can consistently predict realized environmental changes across different generative policy families and task suites. Both Diffusion Policy (DP) and Flow Matching (FM) achieve strong Spearman and Pearson correlations across all evaluated suites, with high-low gain selection accuracy close to 100%. These results demonstrate that the proposed local dynamics analysis is robust across generators and tasks, rather than being specific to a particular policy architecture.

Table 10: Full component-wise ablation of local dynamics sensitivity.

Component	Valid seeds	Mean ρ	High-low win rate
Joint velocity	160/160	0.984	0.999
Full outcome vector	160/160	0.982	0.996
End-effector pose	160/160	0.981	0.999
Gripper state	160/160	0.978	0.995
Joint position	160/160	0.978	0.999
End-effector velocity	160/160	0.977	0.998
Object relative position	160/160	0.977	0.999
End-effector position	160/160	0.971	0.999
Object relative pose	160/160	0.952	0.996
Object position	47/160	0.576	0.880
Object velocity	103/160	0.279	0.687
Object angular velocity	160/160	0.184	0.492
Contact distance	25/160	0.986	0.860
Contact force	25/160	0.983	0.930
Contact existence	2/160	0.522	0.250

Table 10 provides a component-wise analysis of local dynamics sensitivity to identify which physical variables contribute most to predicting realized environmental changes. Robot kinematics, end-effector states, and object-relative states consistently achieve high correlations and near-perfect high-low gain selection accuracy. In contrast, object motion derivatives and sparse contact variables show weaker or unstable correlations due to limited observability. These results indicate that local dynamics effects are selectively distributed across different physical components.

Table 11: Action diversity versus outcome diversity correlation.

Setting	Action-outcome diversity ρ
DP (Diffusion Policy)	0.181
FM (Flow Matching)	0.179
Average across generators	0.180
State-level average	0.190
Positive correlation states	786/800 (98.25%)

Table 11 investigates the relationship between action-space diversity and realized outcome diversity after physical execution. Although generated actions can exhibit substantial variation, their resulting environmental changes show only weak correlations, with average correlation coefficients around 0.18 across different generators. These results indicate that action diversity alone is not a reliable proxy for outcome diversity, as environment dynamics reshape and filter the effects of diverse actions.

D Additional Candidate Sampling Analysis on LIBERO

Sec. 4.3 evaluates the support limits of candidate sampling on the four recovery benchmarks. Here we report the same analysis on the LIBERO suite (Liu et al., 2023) for both diffusion-policy (DP) and flow-matching (FM) base policies, which provides a larger and more diverse set of tasks than the recovery benchmarks.

Table 12: Candidate-sampling limitations across LIBERO (Liu et al., 2023). "No support" denotes states with no useful candidate among $K = 8$ samples. "Missed support" denotes supported states where an available useful candidate is not converted into successful continuation.

Scope	Base-K1	Uniform-K8	No support	Missed support
ALL	59.9%	61.5%	25.0%	18.0%
DP-spatial	78.0%	77.0%	15.0%	9.4%
DP-object	7.0%	13.0%	81.0%	31.6%
DP-goal	8.0%	10.0%	85.0%	33.3%
DP-10	73.0%	77.0%	8.0%	16.3%
FM-spatial	88.0%	87.0%	5.0%	8.4%
FM-object	64.0%	69.0%	5.0%	27.4%
FM-goal	83.0%	85.0%	0.0%	15.0%
FM-10	78.0%	74.0%	1.0%	25.3%

Table 12 reveals the same two limitations. First, *support failure* is substantial: 25.0% of all evaluated states contain no useful candidate within the eight-candidate pool, rising to 81.0% on DP-object and 85.0% on DP-goal. Second, candidate availability does not guarantee successful utilization. Among states containing at least one useful candidate, 18.0% still fail to convert that support into successful continuation. Consistently, increasing the candidate budget from Base-K1 to Uniform-K8 changes overall success only from 59.9% to 61.5%. Thus, larger candidate sets are limited both by missing support and by ineffective utilization of available support, on LIBERO as well as on the recovery benchmarks in Sec. 4.3.

E Success-Manifold Geometry in Latent Space

Figure 6 visualizes the representation geometry underlying TailGuard's success-manifold model. The success bank and held-out successful probes occupy a structured, concentrated region, while failure probes extend beyond this region along several distinct branches. Among 2,088 failure probes, 1,341 (64.2%) exceed the calibrated deviation threshold τ_{dev}, whereas approximately 5% of held-out successful probes exceed it by construction. Failure probes with $S < \tau_{\text{dev}}$ remain closer to the success-manifold region, while those with $S \geq \tau_{\text{dev}}$ predominantly populate peripheral or displaced structures. The complementary 2D and 3D views show the same qualitative organization

from different projections. This agreement between the embedding geometry and the local Mahalanobis score supports interpreting the score as a measure of departure from successful behavior. Importantly, UMAP is used only for visualization. All TailGuard manifold estimation and recovery triggering are performed in the standardized PCA-64 space using the local Mahalanobis metric; distances in the UMAP embeddings are not used by the controller.

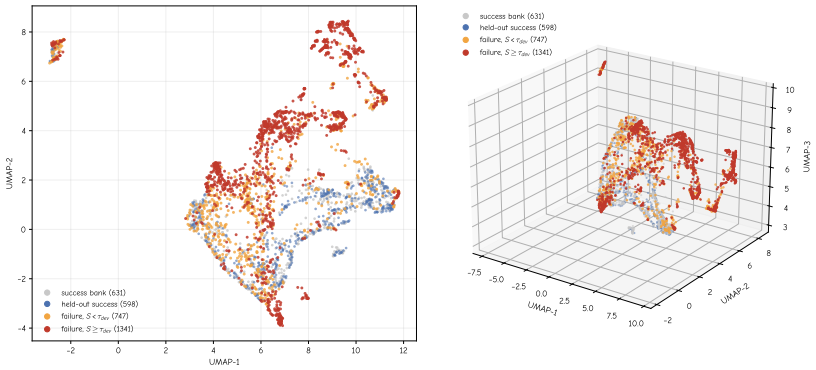

Figure 6: Latent success-manifold geometry of TailGuard under 2D and 3D UMAP projections (visualization only). Gray: success bank; blue: held-out successful probes; orange/red: failure probes below/above the calibrated deviation threshold τ_{dev}.

E.1 Metric Ablation for Success-Manifold Scoring

We compare four metrics for measuring departure from the success manifold: Euclidean kNN, global Mahalanobis distance, local Mahalanobis distance, and a graph-geodesic approximation. Table 13 reports AUROC, AUPRC, and FPR@TPR90, where higher AUROC and AUPRC and lower FPR@TPR90 indicate better separation between successful and failure probes. Local Mahalanobis achieves the strongest performance across all three criteria, with an AUROC of 0.934, an AUPRC of 0.897, and an FPR@TPR90 of 0.139. The improvement over Euclidean kNN and global Mahalanobis suggests that successful representations exhibit locally anisotropic geometry that cannot be adequately captured by isotropic distance or a single global covariance model. The graph-geodesic metric performs substantially worse, indicating sensitivity to neighborhood connectivity and local noise. Together with the latent-space visualization in Figure 6, these results motivate the local Mahalanobis metric used for TailGuard's success-manifold scoring.

Table 13: Ablation of success-manifold distance metrics.

Metric	AUROC ↑	AUPRC ↑	FPR@TPR90 ↓
Euclidean kNN	0.787	0.683	0.500
Global Mahalanobis	0.855	0.772	0.306
Local Mahalanobis	**0.934**	**0.897**	**0.139**
Graph-geodesic	0.707	0.534	0.528

E.2 Representation Ablation for Success-Manifold Modeling

Table 14 further ablates the composition of the success-manifold representation to quantify the contribution of the two representation branches and to examine the effect of dimensional allocation. The ablation compares which branches enter the manifold representation, not whether visual information is available to TailGuard at all: the dynamics branch is itself conditioned on the visual feature (Eq. (19)), so scene context reaches the dynamics-only variant as well. The distinction between the branches is what they encode, namely the current visual scene context for $\bar{z}^{\mathrm{vis}}$ and the action-conditioned predicted physical evolution for $\bar{z}^{\mathrm{dyn}}$. Using only the visual branch yields an AUROC of 0.875 but does not improve closed-loop success over the base policy, resulting in a mean success rate of 49.1%. In contrast, using only the dynamics branch achieves an AUROC of 0.842

Table 14: Ablation of success-manifold representations and visual–dynamics dimensional allocation.

Representation	d_{vis}	d_{dyn}	**AUROC** ↑	**AUPRC** ↑	**FPR@TPR90** ↓	**Mean success** ↑
Representation ablation						
Visual only	64	—	0.875	0.808	0.245	49.1%
Dynamics only	—	64	0.842	0.761	0.287	63.1%
Joint PCA-64	joint projection		0.908	0.854	0.186	68.0%
Block-PCA dimensional allocation						
Block PCA	16	48	0.921	0.881	0.158	69.2%
Block PCA	**32**	**32**	**0.934**	**0.897**	**0.139**	**70.4%**
Block PCA	48	16	0.927	0.888	0.151	69.8%

and a mean success rate of 63.1%. Since the visual branch is action-independent during recovery optimization, the visual-only representation provides no gradient for action correction; its closed-loop success therefore remains at the base-policy level. Jointly projecting the concatenated features to 64 dimensions improves performance to 0.908 AUROC and 68.0% mean success, indicating that the two branches provide complementary information. However, the proposed block-wise projection performs better. Allocating 16 visual and 48 dynamics dimensions gives a 69.2% mean success rate, while the reverse 48+16 allocation reaches 69.8%. The balanced 32+32 allocation achieves the strongest overall performance, with an AUROC of 0.934, an AUPRC of 0.897, an FPR@TPR90 of 0.139, and a mean success rate of 70.4%. These results show that both feature blocks contribute to success-manifold estimation, while balanced block-wise compression prevents either branch from dominating the latent geometry. The improvement over joint PCA further supports separating the two branches before constructing the final success-manifold representation.

F TailGuard Hyperparameter Sensitivity

We evaluate the sensitivity of TailGuard to three key hyperparameters while keeping all other settings fixed: the neighborhood size m, the success-bank size N_{succ}, and the guidance strength λ. Figure 7 reports closed-loop success rates on all four benchmarks. For neighborhood size, performance improves as the local estimate becomes more stable and remains nearly flat over a broad intermediate range, with $m = 50$ lying inside this stable region. For success-bank size, increasing the number of successful rollouts substantially improves performance at small data scales, while the gains become marginal around $N_{\text{succ}} = 200$, indicating saturation of empirical success-manifold coverage. Increasing the bank further produces only small task-dependent fluctuations rather than systematic gains. Guidance strength exhibits a broad inverted-U trend: weak guidance provides insufficient attraction toward the success manifold, whereas overly strong guidance can overrule the frozen policy prior and reduce performance. The default value $\lambda = 0.75$ lies near the center of the high-performing region. Overall, the results show that TailGuard is not critically dependent on a narrowly tuned hyperparameter choice. The dashed vertical lines indicate the default configuration used in the main experiments, and the dotted horizontal lines show the corresponding base-policy success rates for each benchmark. This supports robustness of the recovery mechanism across settings.

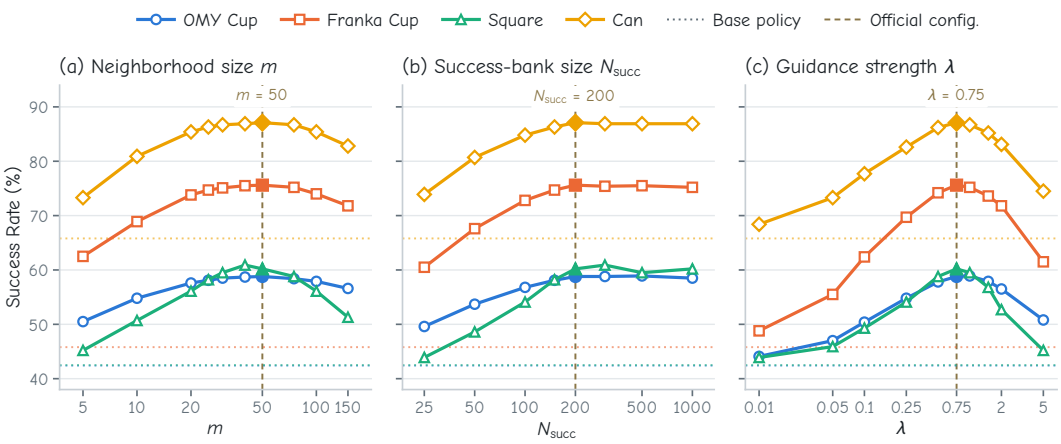

Figure 7: Hyperparameter sensitivity of TailGuard on four benchmarks, varying (a) neighborhood size m, (b) success-bank size N_{succ}, and (c) guidance strength λ with all other settings fixed. Dashed vertical lines mark the default configuration; dotted horizontal lines mark the base-policy success rates.

F.1 Gradient Optimization Hyperparameter Sensitivity

We additionally evaluate the sensitivity of the gradient-based recovery optimization to the number of projected gradient steps J and the step size η (Table 15). All other TailGuard hyperparameters are kept fixed.

Table 15: Sensitivity of TailGuard to the number of projected gradient steps J and step size η.

J	η	Mean success (%)
5	0.01	69.1
10	0.005	69.9
10	0.01	**70.4**
10	0.02	70.2
20	0.01	70.3

Performance remains stable over moderate changes in both optimization hyperparameters. We therefore use $J = 10$ and $\eta = 0.01$ in all experiments, which achieves the highest mean success rate while remaining close to neighboring settings. Both hyperparameters are selected on held-out validation states and fixed across all benchmarks.

F.2 Reference-Bank Capacity

We further vary the capacity N_{bank} of the uniformly subsampled reference bank (Appendix B.2), from 0.5k entries up to the full reference set of 72k entries, with all other settings fixed. Table 16 reports the mean closed-loop success over the four benchmarks together with the number of stored values ($N_{\text{bank}} \times 64$).

Table 16: Sensitivity of TailGuard to the reference-bank capacity N_{bank}. Small brackets denote 95% CI; Δ is relative to the default $N_{\text{bank}} = 5$k.

N_{bank}	Mean success (%)	Δ	Stored values
0.5k	61.7 [60.2, 63.2]	−8.7 pp	0.032M
1k	65.5 [64.0, 67.0]	−4.9 pp	0.064M
2.5k	68.8 [67.4, 70.2]	−1.6 pp	0.160M
5k	**70.4** [69.0, 71.8]	—	**0.320M**
7.5k	70.6 [69.2, 72.0]	+0.2 pp	0.480M
10k	70.5 [69.1, 71.9]	+0.1 pp	0.640M
20k	70.8 [69.4, 72.2]	+0.4 pp	1.280M
50k	70.7 [69.3, 72.1]	+0.3 pp	3.200M
72k (full)	70.9 [69.5, 72.3]	+0.5 pp	4.608M

Performance improves substantially as the reference-bank capacity increases up to $N_{\text{bank}} = 5$k, while further increasing the bank to the full 72k entries yields at most a 0.5 pp gain with overlapping confidence intervals. We therefore use 5k entries as a favorable performance–storage trade-off.

Table 17: Ablation study of the TailGuard handoff mechanism. Small brackets denote 95% confidence intervals.

Benchmark	**Base**	**w/o handoff**	**w/o action-continuity**	**Full**
OMY Cup	42.4% [39.4,45.5]	50.8% [47.7,53.9]	55.9% [52.8,58.9]	**58.8%** [55.7,61.8]
Franka Cup	45.8% [42.7,48.9]	63.5% [60.5,66.4]	70.8% [67.9,73.5]	**75.6%** [72.8,78.2]
Square	42.5% [39.5,45.6]	58.3% [55.3,61.4]	59.4% [56.3,62.5]	**60.2%** [57.1,63.3]
Can	65.8% [62.8,68.7]	80.9% [78.0,83.6]	84.8% [82.0,87.4]	**87.1%** [84.9,89.0]
Mean	49.1%	63.4%	67.7%	**70.4%**

Table 17 isolates the contribution of the compatibility-gated handoff. Recovery without an explicit handoff gate already improves over the frozen base policy, showing that redirecting execution toward the success manifold is beneficial by itself. Requiring re-entry into the success manifold before returning control further improves performance, while adding the action-continuity criterion C_{action} provides an additional gain by withholding the handoff while the base policy's consecutive proposals are still inconsistent (Appendix B.7). The complete handoff mechanism achieves the highest mean success rate, indicating that recovery and handoff play complementary roles.

G Runtime Failure Detection Analysis

We evaluate TailGuard's departure score as a runtime failure detector independently from the recovery controller. Following prior runtime monitoring works, we report AUROC, AUPRC, and FPR at 95% recall. We compare against Sentinel (Agia et al., 2024), which monitors temporal action consistency and task progress, and FIPER (Römer et al., 2026), which combines latent-space out-of-distribution signals with action-chunk uncertainty. All monitors are evaluated on the same evaluation rollouts of each benchmark. A rollout is labeled as failure when the final task success criterion is not satisfied, and detection metrics are computed from intermediate timestep scores before task termination. For Sentinel and FIPER, we use the official implementations with their default configurations. The TailGuard detector is the departure score $S(\Psi(x_t, u_t^\pi))$ of Eq. (7) with the threshold τ_{dev} calibrated as in Appendix B.3; no evaluation trajectory is used for calibration. Table 19 reports detection rates (%) across different failure categories at the calibrated operating point of each monitor.

Table 18: Runtime failure detection performance compared with existing monitors. Higher AUROC and AUPRC and lower FPR at 95% recall are better; the best value per benchmark is in bold.

Benchmark	Detector	AUROC ↑	AUPRC ↑	FPR@95% recall ↓
OMY	Sentinel	0.82	0.79	0.31
	FIPER	0.85	0.82	0.25
	TailGuard detector	**0.89**	**0.87**	**0.18**
Franka	Sentinel	0.79	0.75	0.36
	FIPER	0.83	0.80	0.29
	TailGuard detector	**0.91**	**0.89**	**0.15**
Square	Sentinel	0.76	0.71	0.40
	FIPER	0.81	0.77	0.33
	TailGuard detector	**0.87**	**0.84**	**0.22**
Can	Sentinel	0.84	0.81	0.28
	FIPER	0.88	0.85	0.21
	TailGuard detector	**0.93**	**0.91**	**0.12**
Mean	Sentinel	0.80	0.77	0.34
	FIPER	0.84	0.81	0.27
	TailGuard detector	**0.90**	**0.88**	**0.17**

Table 18 shows that the departure score is a competitive runtime monitor on its own. It attains the highest AUROC and AUPRC and the lowest FPR at 95% recall on all four benchmarks, averaging 0.90 AUROC, 0.88 AUPRC, and 0.17 FPR, compared with 0.84/0.81/0.27 for FIPER and 0.80/0.77/0.34 for Sentinel. The margin is largest on Square and Franka, where failures are dominated by departures from the success manifold that the local Mahalanobis score captures directly. The low false-positive rate at high recall is important for recovery, since spurious triggers would interrupt otherwise successful executions.

Table 19: Detection rate (%) across different failure categories at the calibrated operating point. Higher is better; the best value per category is in bold.

Failure type	Sentinel	FIPER	TailGuard
State deviation	82	87	**93**
Action inconsistency	**91**	88	89
Task progression failure	**86**	84	78
Mean	86	86	**87**

Table 19 breaks detection down by failure category. Detection rate is computed as the fraction of failure instances correctly identified by each monitor under its calibrated threshold. The TailGuard detector is strongest on state-deviation failures (93% versus 87% for FIPER and 82% for Sentinel), which is expected because the departure score directly measures distance from the success manifold in the state–action representation. On action-inconsistency failures it is comparable to the action-consistency monitor of Sentinel (89% versus 91%), since inconsistent action chunks also move the predicted outcome away from successful references. Its weakest category is task-progression failure (78% versus 86% for Sentinel), where the policy remains on the success manifold locally but fails to make progress; detecting such failures requires progress- or phase-aware monitoring, which TailGuard deliberately does not model (Sec. 3.3). The mean detection rates are similar across monitors (86–87%), indicating that the three detectors are complementary rather than strictly ordered, with the departure score offering the best trade-off for the recoverable state-deviation failures that TailGuard targets.

H Closed-Loop Success by Initial-State Region

Table 20 breaks down the closed-loop success rates of Table 5 by the initial-state regions defined in Table 2. Each benchmark is evaluated on 1,000 episodes, 200 per region, so region-level success rates are reported at a resolution of 0.5%, and the Overall row is the mean over the five regions.

Table 20: Closed-loop success rates by initial-state region for the frozen base policy (Base) and TailGuard (Ours). Each region contains 200 evaluation episodes per benchmark. Overall is the mean over the five regions and matches Table 5. Small brackets denote 95% CI.

Benchmark	Region	Base	Ours
OMY	Head	96.5% [93.0,98.3]	96.5% [93.0,98.3]
	Low-density	70.5% [63.8,76.4]	70.5% [63.8,76.4]
	Near-Tail	37.5% [31.1,44.4]	53.5% [46.6,60.3]
	Mid-Tail	7.5% [4.6,12.0]	44.5% [37.8,51.4]
	Far-Tail	0.0% [0.0,1.9]	29.0% [23.2,35.6]
	Overall	42.4% [39.4,45.5]	**58.8%** [55.7,61.8]
Franka	Head	98.0% [95.0,99.2]	98.0% [95.0,99.2]
	Low-density	78.5% [72.3,83.6]	78.5% [72.3,83.6]
	Near-Tail	42.0% [35.4,48.9]	88.0% [82.8,91.8]
	Mid-Tail	10.5% [7.0,15.5]	69.5% [62.8,75.5]
	Far-Tail	0.0% [0.0,1.9]	44.0% [37.3,50.9]
	Overall	45.8% [42.7,48.9]	**75.6%** [72.8,78.2]
Square	Head	94.5% [90.4,96.9]	94.5% [90.4,96.9]
	Low-density	72.5% [65.9,78.2]	72.5% [65.9,78.2]
	Near-Tail	37.5% [31.1,44.4]	61.5% [54.6,68.0]
	Mid-Tail	8.0% [5.0,12.6]	47.5% [40.7,54.4]
	Far-Tail	0.0% [0.0,1.9]	25.0% [19.5,31.4]
	Overall	42.5% [39.5,45.6]	**60.2%** [57.1,63.3]
Can	Head	99.0% [96.4,99.7]	99.0% [96.4,99.7]
	Low-density	89.5% [84.5,93.0]	92.0% [87.4,95.0]
	Near-Tail	71.5% [64.9,77.3]	90.0% [85.1,93.4]
	Mid-Tail	50.0% [43.1,56.9]	84.0% [78.3,88.4]
	Far-Tail	19.0% [14.2,25.0]	70.5% [63.8,76.4]
	Overall	65.8% [62.8,68.7]	**87.1%** [84.9,89.0]

Success rates in the Head and Low-density regions are essentially unchanged (the only difference is a small gain on the Can Low-density region), so the overall improvement is obtained entirely from the Near-, Mid-, and Far-Tail regions. The absolute improvement is largest in the Mid-Tail region on OMY, Franka, and Square and in the Far-Tail region on Can; in particular, on OMY, Franka, and Square the base policy never succeeds from Far-Tail initial states, whereas TailGuard recovers 29.0%, 44.0%, and 25.0% of these episodes, respectively.

I Model Size and Runtime Overhead

We further report the model size and runtime overhead of TailGuard (Tables 21 and 22). TailGuard introduces only 0.14M trainable parameters together with approximately 0.32M stored reference values for success-manifold estimation ($N_{\text{bank}} = 5{,}000$ uniformly subsampled 64-dimensional entries per task; Appendix B.2). This overhead is small relative to the frozen base policies, whose parameter counts range from 65.9M to 450.0M.

As Table 22 shows, the computational overhead of the TailGuard module itself is below one millisecond per control step. The small increase in episode-level wall-clock time mainly reflects additional closed-loop interaction introduced by recovery and handoff, rather than expensive TailGuard inference. Overall, TailGuard adds only modest runtime overhead while substantially improving closed-loop success.

Table 21: Parameter and storage overhead of TailGuard relative to the frozen base policies.

Benchmark	Base policy	Base params.	TailGuard overhead
OMY Cup	SmolVLA	450.0M	0.14M trainable + 0.32M stored
Franka Cup	Diffusion Policy	266.7M	
Square	Diffusion Policy	65.9M	
Can	Diffusion Policy	65.9M	

Table 22: Wall-clock evaluation cost on the same hardware. Per-step time includes policy inference and TailGuard execution.

Benchmark	Base	TailGuard	Difference
OMY	0.036 s/step	0.037 s/step	+2.8%
Franka	0.215 s/step	0.223 s/step	+3.7%
Square	39 s/episode	40 s/episode	+2.6%
Can	30 s/episode	31 s/episode	+3.3%

J Additional Qualitative Rollouts

Figure 8 provides additional qualitative rollout examples across four manipulation benchmarks: Franka Red Cup, OMY Red Cup, RoboMimic Square, and RoboMimic Can.

These examples complement the quantitative success-rate evaluation in the main paper.

The visualizations show representative failure cases of the frozen base policies and the corresponding recovery behaviors of TailGuard. In challenging states, the base policies may enter regions where the generated actions no longer lead to successful task completion. TailGuard detects these off-manifold states and applies an external recovery mechanism to return the system to a controllable region, after which the original policy resumes execution.

LIBERO. Figure 9 shows the same paired visualization on the four LIBERO suites (Liu et al., 2023) (Spatial, Object, Goal, and LIBERO-10) with a diffusion-policy base policy. Each fixed evaluation case places the target object outside the bounding box of the demonstrated object positions; the corner note gives the case id and its tail bin (Near-Tail: 0.5–3 cm, Mid-Tail: 3–6 cm, Far-Tail: 6–10 cm outside the box). These are single-episode illustrations of the runtime on LIBERO, not an aggregate evaluation.

Real robot. Figure 10 shows the same paired visualization on a physical Franka arm for three further tabletop tasks (Pick, Place, and Screw) with a diffusion-policy base policy; the cylinder-stacking trial is shown in Figure 3 of the main text. For each task we show one base-policy trial that fails and one TailGuard trial that completes the task. These are qualitative single-trial demonstrations, not a statistical evaluation; the full clips are provided in the supplementary video.

Additional embodiments. Figure 11 shows the same paired visualization on three further embodiment–policy pairs: **(a)** the public SmolVLA checkpoint for the OpenArm Cube-in-Box task, **(b)** a Diffusion Policy checkpoint for OpenArm Red-Block pick-and-place, and **(c)** the public SmolVLA checkpoint for the G1 dual-arm cup task. We use the same high-level TailGuard recovery pipeline across all three examples, while adapting the embodiment-specific low-level controller to each robot. Panel (a) retains the detailed OpenArm Cube-in-Box implementation, including the trigger, the finite-horizon simulator evaluation, and the DLS-IK recovery servo. It is executed in the OpenArm v2 MuJoCo scene with the TailGuard runtime behind an OpenArm adapter, and two elements of that port differ from the main benchmarks: no detector is calibrated for this embodiment, so recovery is admitted at $t = 0$ whenever the cube lies outside the bounding box of the demonstrated cube positions, and candidate ranking at handoff uses the simulator as the finite-horizon dynamics model. The recovery servo is a damped-least-squares IK servo on the joint-position actuators, since this arm has no operational-space controller. Fixed cases place the cube 0.5–3, 3–6, or 6–10 cm outside the demonstrated support box (Near-, Mid-, and Far-Tail), two per bin. In the Mid-Tail case of panel (a), the base policy never reaches the cube within the 150-step horizon, whereas TailGuard

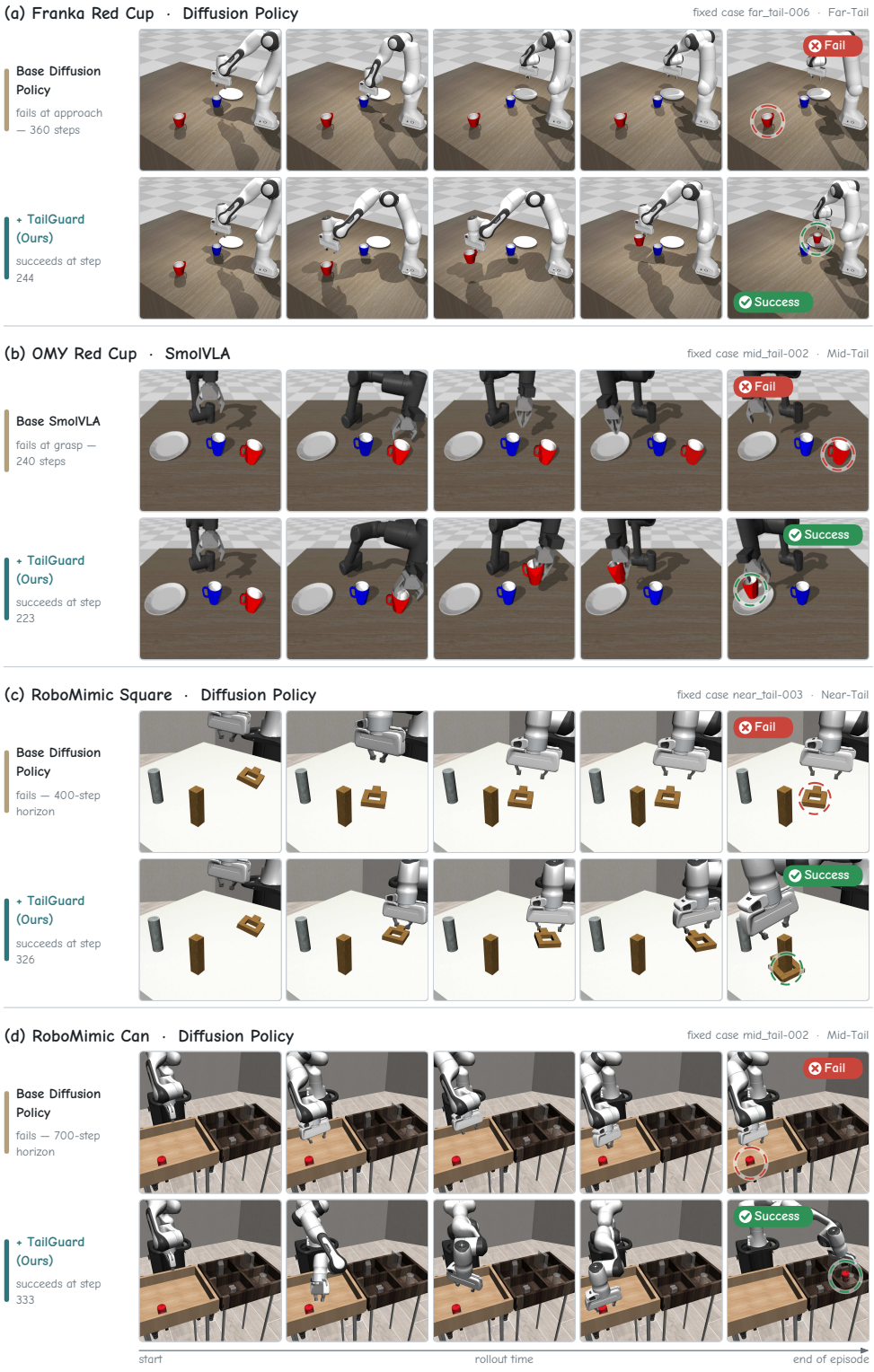

Figure 8: Additional qualitative rollout examples across four manipulation benchmarks. The base policies fail in challenging cases, while TailGuard recovers the execution and completes the task.

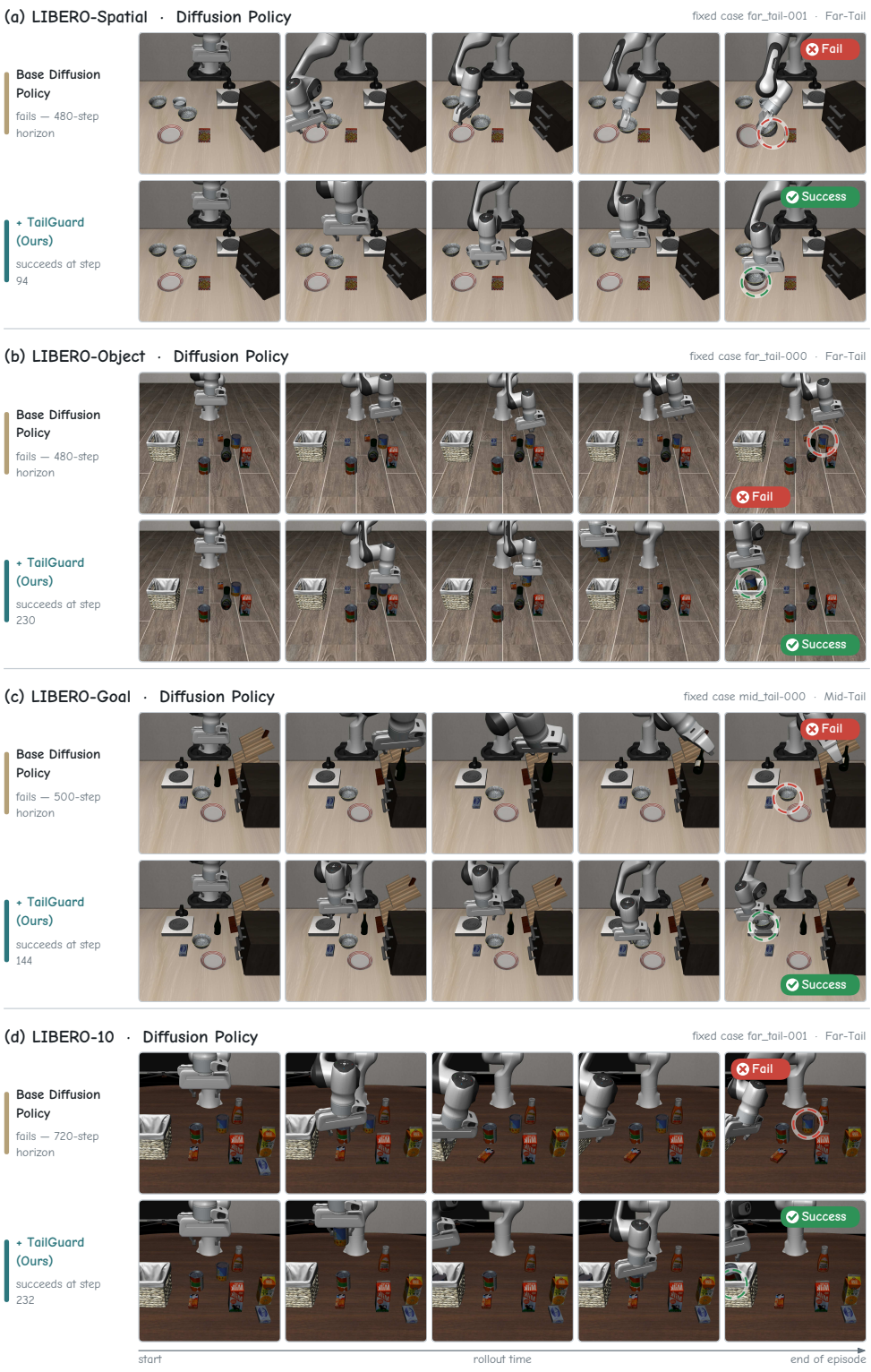

Figure 9: Qualitative rollout examples on the four LIBERO suites. In each panel the diffusion-policy base policy fails on a fixed tail case, while TailGuard recovers the execution and completes the task.

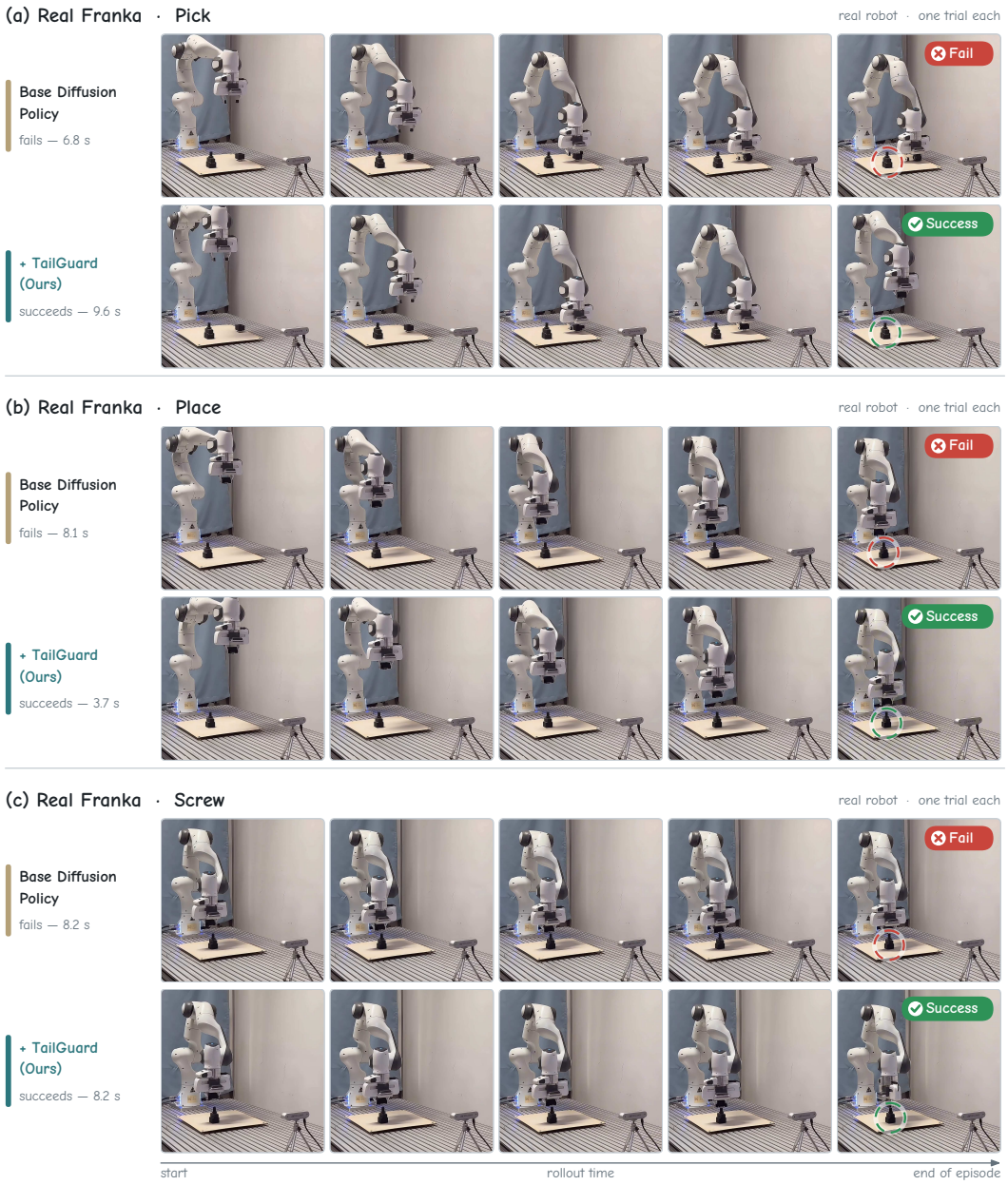

Figure 10: Real-robot qualitative demonstration on three further Franka tabletop tasks. In each panel the base Diffusion Policy fails (top row) while TailGuard completes the task (bottom row); one trial each.

completes the grasp, hands control back at step 22, and the cube rests in the tray at step 42; this is a single-episode illustration on a six-case development split (base 0/6, TailGuard 1/6), not an aggregate evaluation. Panels (b) and (c) use the same high-level recovery logic but employ low-level controllers adapted to the corresponding OpenArm and G1 embodiments, and are included as qualitative transfer examples rather than additional quantitative benchmarks: the base policy fails within the 900-step and 700-step horizons, whereas TailGuard succeeds at step 231 on the Near-Tail Red-Block case and at step 504 on the Mid-Tail cup case.

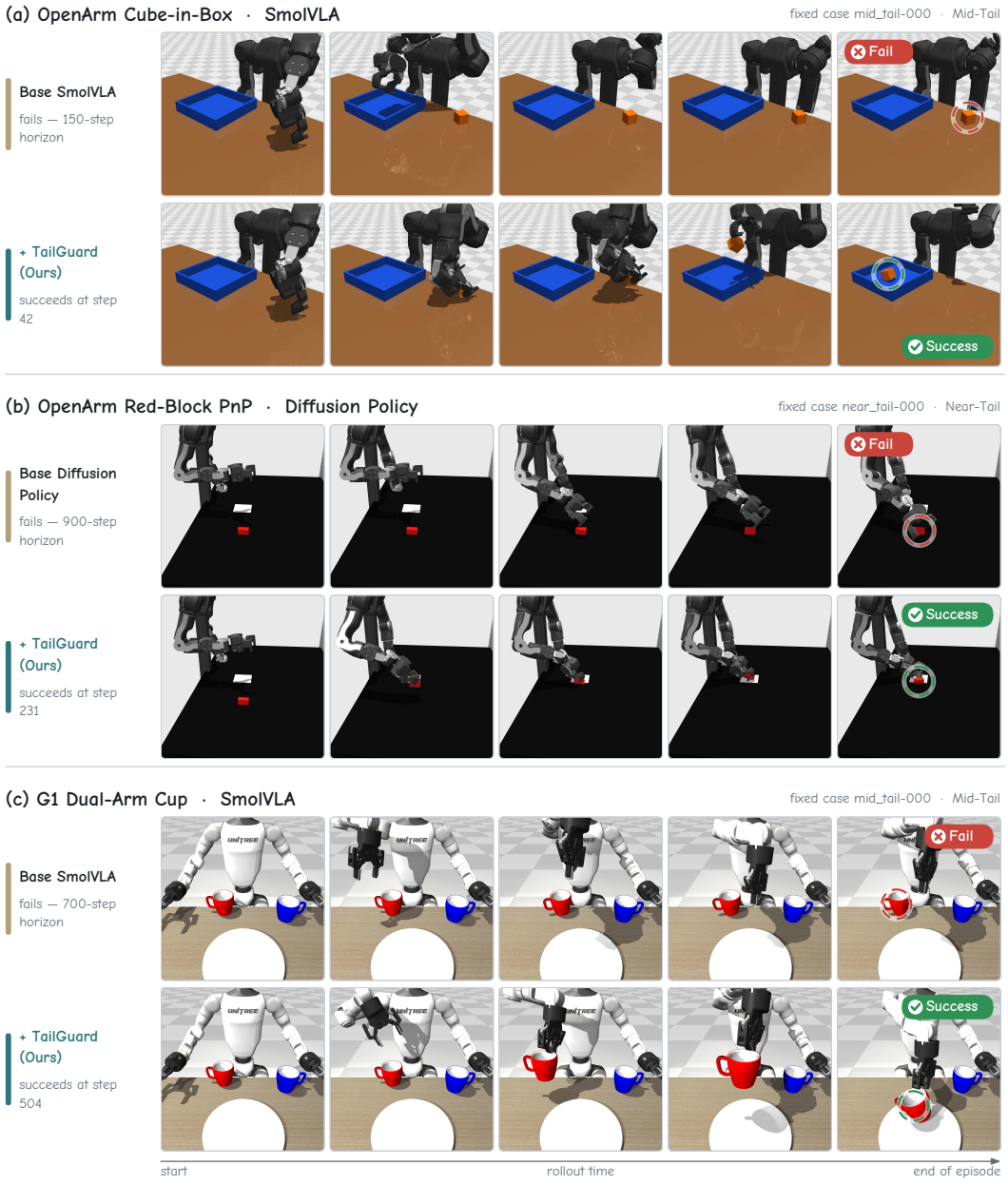

Figure 11: Qualitative rollouts on three further embodiment–policy pairs. **(a)** OpenArm Cube-in-Box with a public SmolVLA base policy, Mid-Tail case: the base policy fails within the 150-step horizon while TailGuard succeeds at step 42. **(b)** OpenArm Red-Block pick-and-place with a Diffusion Policy base policy, Near-Tail case: the base policy fails within the 900-step horizon while TailGuard succeeds at step 231. **(c)** G1 dual-arm cup with a public SmolVLA base policy, Mid-Tail case: the base policy fails within the 700-step horizon while TailGuard succeeds at step 504. In each panel the top row is the base policy and the bottom row is TailGuard. Single episodes; see text.