IS ONLINE INTERACTION NECESSARY FOR RECOVERY? A MINIMALIST APPROACH TO ROBUST PLANNING VIA PERTURBATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Behavior cloning (BC) is vulnerable to covariate shift during closed-loop execution, where small prediction or execution errors can drive the robot toward states poorly covered by the demonstration data. We focus on action-sequence planning, where a policy predicts a finite-horizon sequence of actions as a reference trajectory for robot execution. Existing approaches to covariate shift often rely on collecting additional corrective demonstrations, requiring further environment interaction and access to an expert or reference policy. We propose Perturbation-Augmented Recovery Supervision (PARS), a simple training approach for improving recovery using only existing demonstrations. PARS perturbs the robot's proprioceptive state and anchors only the later portion of the predicted trajectory to the original demonstration, leaving the earlier portion free to generate corrective motion and recover toward the demonstrated behavior within the planning horizon. Unlike conventional input-noise augmentation, which preserves the original supervision target over the entire prediction horizon, PARS explicitly provides trajectory-level supervision for recovery from perturbed states. PARS requires neither additional environment interaction nor expert queries and can be instantiated across different action-sequence policy classes with only minor modifications to their BC objectives. Experiments on 51 RLBench manipulation tasks with flow-based, transformer-based, and diffusion-based policies demonstrate that PARS improves robustness to covariate shift during closed-loop execution.

1 INTRODUCTION

Behavior cloning (BC) is a method for training a policy to reproduce desired behaviors from demonstrations, providing a direct way to acquire complex robotic skills without explicitly specifying reward functions (Osa et al., 2018). However, BC from (near-)expert demonstrations is susceptible to covariate shift during closed-loop execution, where small prediction or execution errors can drive the robot toward states poorly covered by the demonstration data and eventually lead to performance degradation. Several approaches have sought to mitigate this issue by improving robustness around the demonstrated distribution through corrective-label synthesis, distributionally robust objectives, stability-aware learning, and trajectory synthesis (Ke et al., 2024; Seo et al., 2024; Agrawal et al., 2025; Mehta et al., 2025; Wang et al., 2026). Beyond single-step action prediction, predicting temporally extended action sequences over a finite horizon has become widely adopted in robotic BC and can further mitigate compounding errors (Lazzati et al., 2026; Zhao et al., 2023; Chi et al., 2025). More recent approaches additionally introduce explicit stabilization or filtering mechanisms to improve robustness against deviations during execution (Jiang et al., 2025; Rosasco et al., 2025).

Another line of work addresses distribution shift more directly by augmenting the training dataset with corrective demonstrations. Ross et al. (2011) iteratively rolls out the learned policy and queries an expert for corrective actions at visited states, thereby expanding the training distribution toward states encountered during deployment. Alternatively, Laskey et al. (2017) inject perturbations into expert actions to induce likely error states and collect recovery behaviors. Subsequent interactive imitation learning methods reduce the burden of expert supervision through selective querying or intervention (Zhang & Cho, 2016; Kelly et al., 2019). However, these approaches require additional

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.

environment interaction and continued access to an expert or reference policy, increasing the cost of applying them to real-world robotic systems.

In this paper, we consider planners that predict finite-horizon sequences of desired robot configurations as reference trajectories, a formulation widely used in robotic systems. We propose **P**erturbation-**A**ugmented **R**ecovery **S**upervision (PARS) to improve recovery from covariate shift without additional environment interaction. During training, PARS perturbs proprioceptive observations from existing demonstrations and anchors only the predicted trajectory suffix to the corresponding demonstration suffix. The preceding prefix receives no direct augmented target, leaving room for corrective motion before rejoining the demonstrated continuation. Unlike conventional input-noise augmentation, which retains the original target over the full horizon and primarily encourages invariance to perturbations (Ke et al., 2021; Ameperosa et al., 2025), PARS specifies where the perturbed trajectory should return. The standard BC objective still trains the full action sequence on unperturbed demonstrations. PARS therefore requires no explicit recovery trajectories and can be incorporated into existing action-sequence policies with only minor modifications to their training objectives.

Our contributions are summarized as follows:

- We propose PARS, a simple supervision scheme for improving recovery during closed-loop execution using only existing demonstrations, without additional environment interaction or expert queries during training.
- We provide a theoretical analysis of PARS, characterizing how suffix anchoring affects local sensitivity to proprioceptive perturbations across policy classes and deriving sufficient conditions for the free recovery prefix to remain close to demonstrated motion and satisfy joint-space speed and acceleration limits.
- Across 51 RLBench manipulation tasks and multiple action-sequence policy classes, we show that PARS improves mean closed-loop success and promotes recovery toward demonstrated behavior or mitigates further deviation.

2 Related Works

2.1 Robustness in Single-Step Behavior Cloning

BC commonly trains a policy to predict a single action conditioned on the current observation, making it susceptible to compounding errors under covariate shift. Several approaches have sought to improve the robustness of policies that predict individual actions. CCIL (Ke et al., 2024) synthesizes corrective labels using learned local dynamics, DrilDICE (Seo et al., 2024) applies stationary-distribution correction, and other methods exploit structural or stability properties of demonstration trajectories (Agrawal et al., 2025; Mehta et al., 2025). More recently, KATS (Wang et al., 2026) generates dynamically consistent trajectories in a learned Koopman latent space to augment the training distribution. As shown in Appendix A, these methods perform substantially worse than action-sequence policies in our setting; we therefore focus our main evaluation on action-sequence methods.

2.2 Robustness through Action-Sequence Modeling

An alternative direction is to move beyond single-step action prediction and generate temporally extended action sequences. Action-sequence modeling has become widely adopted in robotic BC, with recent analysis attributing its benefits to reduced compounding errors, increased non-Markovian expressivity, and implicit ensembling across temporal predictions (Lazzati et al., 2026). Action Chunking with Transformers (ACT) (Zhao et al., 2023) predicts action sequences using a Transformer-based conditional generative model, while Diffusion Policy (DP) (Chi et al., 2025) represents distributions over action sequences through a denoising diffusion process. More recent methods explicitly target robustness to deviations during execution. Streaming Flow Policy (SFP) (Jiang et al., 2025) represents action trajectories as flow trajectories and introduces a stabilizing flow field around demonstrated behaviors, whereas KDPE (Rosasco et al., 2025) generates multiple candidate action sequences from a diffusion policy and filters potentially harmful trajectories using kernel density estimation.

2.3 Interactive Corrective Data Collection

A direct way to address covariate shift is to expand the training distribution with states outside the original demonstration distribution and provide corrective supervision at those states. DAgger (Ross et al., 2011) established this paradigm by iteratively rolling out the learned policy, querying an expert at visited states, and aggregating the newly labeled samples into the training dataset. Subsequent methods reduce the supervision burden through selective querying or intervention, as in SafeDAgger (Zhang & Cho, 2016) and HG-DAgger (Kelly et al., 2019). Rather than rolling out the learner, DART (Laskey et al., 2017) perturbs expert actions during demonstration collection to induce deviations from the expert trajectory and collect corresponding recovery behaviors. More recently, Zhang et al. (2026) analyzed an exploratory data-collection scheme that perturbs expert actions during data collection while retaining expert supervision at the resulting states, thereby collecting data around expert trajectories without rolling out the learned policy. Despite their different collection strategies, these approaches require additional environment interaction and expert or reference supervision to obtain corrective data. Our work instead investigates whether recovery supervision can be constructed directly from existing demonstrations without collecting additional interactive trajectories.

3 Preliminaries

Problem Formulation. We consider a sequential decision-making problem in which the robot receives an observation $o_t = (o_t^{\text{prop}}, o_t^{\text{env}})$, where o_t^{prop} contains the robot's proprioceptive state, including its current configuration and gripper state, and o_t^{env} describes the surrounding scene and other task-relevant information. In this work, we consider a robotic manipulator equipped with a gripper. Each action is represented as $a_t = (q_t^{\text{des}}, g_t)$, where $q_t^{\text{des}} \in \mathbb{R}^{d_q}$ specifies the desired robot configuration, d_q is the number of robot degrees of freedom, and $g_t \in \mathbb{R}$ specifies the gripper command. The desired configuration is subsequently tracked by a low-level controller, so that the policy specifies high-level reference commands rather than low-level control inputs. In our experiments, the policy parameterizes these configuration commands as joint displacements, which are accumulated from the configuration at the beginning of each action sequence to obtain the desired configurations.

Action-Sequence Planning. Rather than predicting a single action, we consider a planning policy that predicts an H-step sequence of future actions,

$$\mathbf{a}_t = (a_t, a_{t+1}, \ldots, a_{t+H-1}) \,. \tag{1}$$

The sequence of desired configurations forms a reference trajectory for the robot over the planning horizon. We represent the planning policy as $\mathbf{a}_t \sim \pi_\theta(\cdot \mid o_t)$ and use $f_\theta(o_t)$ to denote one H-step action sequence sampled from this policy.

Behavior Cloning. Let $\mathcal{D} = \{\tau_i\}_{i=1}^N$ denote a dataset of N demonstration trajectories, where $\tau_i = \{(o_t, a_t)\}_{t=0}^{T_i-1}$. From each trajectory, we construct valid training pairs consisting of an observation o_t and its corresponding H-step future action sequence $\mathbf{a}_t$, and write $(o_t, \mathbf{a}_t) \sim \mathcal{D}$ to denote sampling such a pair from the demonstration dataset. BC trains the policy to reproduce the demonstrated action sequences by minimizing

$$\mathcal{L}_{\text{BC}}(\theta) = \mathbb{E}_{(o_t, \mathbf{a}_t)\sim\mathcal{D}} \left[\ell \left(\pi_\theta(\cdot \mid o_t), \mathbf{a}_t \right) \right], \tag{2}$$

where ℓ denotes the training objective associated with the chosen policy model.

Covariate Shift in Closed-Loop Execution. Let $d_E(o)$ denote the observation distribution represented by the demonstrations and $d_{\pi_\theta}(o)$ the observation distribution induced by executing the learned policy. BC minimizes its training objective under d_E, whereas the policy is evaluated under d_{π_θ} during closed-loop execution. To distinguish the predicted action sequence from the trajectory resulting from its execution, let $\boldsymbol{\tau}_\theta(o) = \Phi\big(o, f_\theta(o)\big)$, where Φ denotes the execution mapping from the current observation and predicted action sequence to the resulting robot trajectory. Prediction or execution errors can cause the realized trajectory to deviate from the behaviors represented in the demonstrations, thereby altering subsequent observations. As a result, in general, $d_{\pi_\theta} \neq d_E$. As

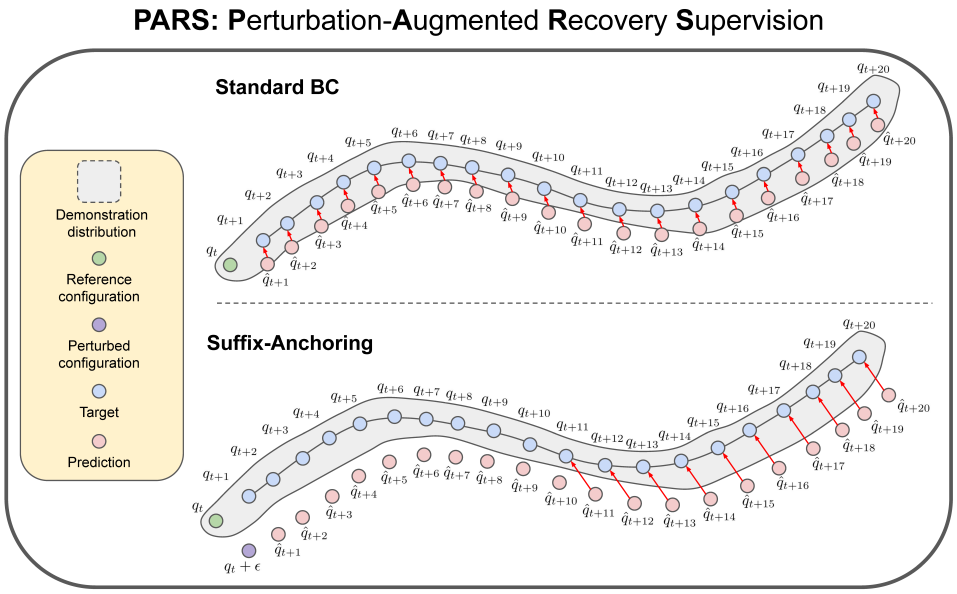

Figure 1: Standard BC supervises the full predicted trajectory at an unperturbed observation (top). For a perturbed proprioceptive observation, PARS anchors only the predicted suffix to the corresponding demonstrated suffix (bottom). The prefix receives no direct supervision from the perturbed sample, leaving room for corrective motion.

execution progresses, these deviations can accumulate and drive the robot toward observations that are poorly covered by the demonstrations, where the learned policy may produce increasingly unreliable predictions (Ross et al., 2011). This mismatch between the training and execution observation distributions is commonly referred to as *covariate shift*.

4 Method

4.1 Perturbation-Augmented Recovery Supervision

We propose Perturbation-Augmented Recovery Supervision (PARS), a simple supervision scheme for improving recovery from covariate shift in action-sequence planning. Under covariate shift, where the robot's proprioceptive state deviates from those represented in the demonstrations, the policy may no longer reliably reproduce the demonstrated motion. The key intuition behind PARS is to encourage the planner to generate a trajectory that approaches the demonstrated trajectory within the planning horizon, bringing the robot toward states where it can more reliably reproduce demonstrated behavior. To induce such recovery behavior, we perturb the proprioceptive observation and apply direct supervision only to the suffix of the predicted action sequence, encouraging it to match the corresponding suffix of the original demonstration, as illustrated in fig. 1. Formally, we define

$$\begin{aligned} \epsilon &\sim \mathcal{U}\left(-\delta_e, \delta_e\right), \\ \tilde{o}_t &= \left(o_t^{\text{prop}} + \epsilon, o_t^{\text{env}}\right), \\ [\mathbf{x}]_{H_r:H} &= \left(x_{H_r}, \ldots, x_{H-1}\right), \end{aligned} \tag{3}$$

where $\mathcal{U}(-\delta_e, \delta_e)$ denotes a uniform distribution over the task-dependent perturbation range $[-\delta_e, \delta_e]$, and H_r denotes the number of initial planning steps left free from direct augmented supervision. The perturbation ϵ has dimension $d_e = \dim(o_t^{\text{prop}})$ and is applied to both the robot configuration and gripper state. We write $[\,\cdot\,]_{H_r:H}$ for the suffix of any length-H sequence, so that $[\mathbf{a}_t]_{H_r:H}$ is the demonstrated suffix and $[f_\theta(\tilde{o}_t)]_{H_r:H}$ the corresponding predicted one.

Based on this formulation, for policies that directly predict an action sequence, we augment the standard BC objective with a suffix-anchoring loss:

$$\mathcal{L}_{\mathrm{PARS}}(\theta) = \mathbb{E}_{(o_t,\mathbf{a}_t)\sim\mathcal{D}}\Big[\underbrace{\ell\left(f_\theta(o_t), \mathbf{a}_t\right)}_{\mathcal{L}_{\mathrm{BC}}} + \lambda_{\mathrm{aug}} \underbrace{\ell\left(\left[f_\theta(\tilde{o}_t)\right]_{H_r:H}, \left[\mathbf{a}_t\right]_{H_r:H}\right)}_{\mathcal{L}_{\mathrm{aug}}}\Big], \tag{4}$$

Here λ_{aug} represents the contribution of the augmented suffix-anchoring loss. In our implementation, augmented supervision is applied with a task-specific perturbation probability p; its resulting weight in the training objective is detailed in Appendix D.2. We first analyze the squared-error case. For policy classes whose training objectives are not defined directly on predicted action sequences, such as diffusion- and flow-based policies, we instantiate the same suffix-supervision principle in their respective training objectives, as described in Section 4.3.

4.2 Theoretical Justification

The suffix-anchoring loss acts as a selective invariance regularizer. Following the slicing notation defined above, $[f_\theta(o)]_{H_r:H}$ denotes the suffix of the predicted action sequence. For a training sample $(o_t, \mathbf{a}_t)$, we consider the augmented loss in squared-error form for the following analysis:

$$\mathcal{L}_{\mathrm{aug}}(\epsilon) = \frac{1}{2}\left\|\left[f_\theta(\tilde{o}_t)\right]_{H_r:H} - \left[\mathbf{a}_t\right]_{H_r:H}\right\|^2, \tag{5}$$

where $\tilde{o}_t$ is the perturbed observation defined in eq. (3). This objective compares only the suffix of the action sequence predicted from the perturbed observation with the corresponding suffix of the demonstrated action sequence.

For small perturbations and a small nominal suffix-prediction residual, the expansion in Appendix B.1 approximates the expected augmented loss as

$$\mathbb{E}_\epsilon\left[\mathcal{L}_{\mathrm{aug}}(\epsilon)\right] \approx \underbrace{\frac{1}{2}\left\|\left[f_\theta(o_t)\right]_{H_r:H} - \left[\mathbf{a}_t\right]_{H_r:H}\right\|^2}_{\text{unperturbed suffix loss}=\mathcal{L}_{\mathrm{aug}}(0)} + \underbrace{\frac{1}{2}\operatorname{tr}\left(J_{\mathrm{suf}}\Sigma_\epsilon J_{\mathrm{suf}}^\top\right)}_{\text{suffix sensitivity regularizer}}, \tag{6}$$

where

$$J_{\mathrm{suf}} = \left.\frac{\partial[f_\theta(o)]_{H_r:H}}{\partial o^{\mathrm{prop}}}\right|_{o=o_t}, \tag{7}$$

and $\Sigma_\epsilon = \mathrm{Cov}(\epsilon)$. The second term in eq. (6) penalizes the sensitivity of the predicted suffix to proprioceptive perturbations, while no such direct penalty is imposed on the prefix. Thus, the augmented suffix-anchoring loss acts as a selective invariance regularizer over the later portion of the predicted action sequence. The detailed derivation is provided in Appendix B.1.

Local feasibility and consistency of the free prefix. The augmented loss does not directly supervise the prefix predicted from a perturbed observation. We therefore ask whether this prefix stays close to the configuration trajectories in the demonstrations and whether its joint-space speed and acceleration remain within the limits used to generate those demonstrations. We derive sufficient conditions for both properties.

Let $\mathbf{q}_\theta^{\mathrm{pre}}(o)$ denote the desired-configuration prefix generated by the policy, and let $\mathcal{S}_{\mathrm{data}}^{\mathrm{pre}}$ denote the set of configuration prefixes represented in the demonstrations. We define the trajectory-level distance between two prefixes as $d_{\mathrm{traj}}(\mathbf{q}, \mathbf{q}') = \max_{0\le k<H_r}\|q_k - q'_k\|_2$ and use $\mathrm{dist}(\cdot, \mathcal{S}_{\mathrm{data}}^{\mathrm{pre}})$ for the distance induced by this metric to the demonstration-prefix set. We assume that the nominal prefix lies within distance η of this set and that the predicted prefix is locally Lipschitz with respect to the proprioceptive perturbation:

$$\mathrm{dist}\left(\mathbf{q}_\theta^{\mathrm{pre}}(o_t), \mathcal{S}_{\mathrm{data}}^{\mathrm{pre}}\right) \le \eta, \qquad d_{\mathrm{traj}}\left(\mathbf{q}_\theta^{\mathrm{pre}}(\tilde{o}_t), \mathbf{q}_\theta^{\mathrm{pre}}(o_t)\right) \le L_q\|\epsilon\|_2. \tag{8}$$

Since $\|\epsilon\|_2 \le \sqrt{d_e}\,\delta_e$, these assumptions imply

$$\mathrm{dist}\left(\mathbf{q}_\theta^{\mathrm{pre}}(\tilde{o}_t), \mathcal{S}_{\mathrm{data}}^{\mathrm{pre}}\right) \le \eta + L_q\sqrt{d_e}\,\delta_e. \tag{9}$$

Thus, a locally perturbed prefix may deviate from the nominal trajectory while remaining close to demonstrated motion. We additionally assume that the nominal prefix satisfies the joint-space speed

and acceleration limits used for trajectory planning with positive margins m_v and m_a. Under the same local-sensitivity assumption, the perturbed prefix remains within these limits whenever

$$\frac{2L_q\sqrt{d_e}\,\delta_e}{\Delta t} \le m_v, \qquad \frac{4L_q\sqrt{d_e}\,\delta_e}{\Delta t^2} \le m_a. \tag{10}$$

Together, eqs. (9) and (10) show that, under sufficiently small local perturbations, the free prefix can retain flexibility for corrective motion while remaining locally consistent with demonstrated motion and within the joint-space speed and acceleration limits used for trajectory planning. Detailed derivations are provided in Appendix C.

4.3 Instantiating PARS across policy classes

PARS requires the augmented objective to supervise only the suffix of the predicted action sequence. For policies that directly apply a loss to individual steps of the predicted action sequence, this can be implemented by simply restricting the augmented loss to the suffix. However, other policy classes, such as flow-matching and diffusion policies, supervise intermediate generative quantities rather than the final predicted action sequence, requiring different implementations of the same suffix-only supervision. We describe below how PARS is instantiated across the policy classes used in our experiments. The corresponding sensitivity analyses are provided in Appendix B.

Case 1: direct chunk-prediction policies. For policies that directly predict an entire action sequence and compute the loss separately at each step of the sequence, PARS can be directly applied by computing the augmented loss only over the trajectory suffix. The per-step loss is computed on accumulated joint positions, preserving the corrective displacement in the supervised suffix. The transformer policy follows this setting and therefore requires no further modification beyond restricting the augmented loss to the suffix. ACT (Zhao et al., 2023) likewise predicts an entire action chunk, but is trained as a conditional VAE with an L_1 reconstruction term and a KL regularization term. For ACT, we apply the suffix restriction to the reconstruction term in the same accumulated trajectory space while leaving the KL term unchanged.

Case 2: streaming flow policies. Streaming Flow Policy (SFP) (Jiang et al., 2025) treats the action trajectory itself as a flow trajectory: integrating its learned velocity field advances the predicted action along sequence time $\tau \in [0, 1]$. For PARS, we apply the augmented velocity-regression loss only after $\tau_r := H_r/H$. Supervising velocities over this suffix does not fix the action reached at τ_r, which depends on the freely generated prefix. We therefore integrate the policy's velocity field up to τ_r for each perturbed observation and use the resulting action as the starting point for suffix supervision. The remaining sequence is then trained toward the demonstrated continuation, while the prefix receives no direct augmented supervision.

Case 3: joint generative policies. Conventional flow-matching and diffusion policies do not directly regress the target action sequence. Instead, they learn a velocity or noise field from complete target sequences, making suffix-only supervision less straightforward. Under a perturbed observation $\tilde{o}_t$, the corresponding recovery trajectory is unavailable in the demonstration dataset: the desired recovery prefix is unknown, while the available supervision specifies only that the later portion of the trajectory should remain consistent with the demonstrated suffix. For a perturbed observation, we generate an action sequence with the current FP or DP policy and refine it by reducing the suffix-anchoring loss on accumulated joint configurations. The refined sequence then serves as a surrogate target for the policy's original flow-matching or diffusion training objective. This adapts the sample-refinement principle of Park et al. (2026) to suffix supervision; the refinement settings are reported in Appendix D.2.

5 Experiments

5.1 Experimental Setup

We evaluate PARS on 51 RLBench manipulation tasks (James et al., 2020) using a Franka Panda 7-DoF robotic arm. We use low-dimensional state observations rather than images. Each observation contains the robot's joint positions, joint velocities, and gripper state, together with task-specific

Table 1: Performance of PARS across different action-sequence methods on 51 RLBench tasks. Mean success rate (%) over 50 evaluation episodes; for PARS, we report the best per-task augmentation setting. The last column counts the number of tasks on which PARS matches or exceeds the corresponding baseline. Full results for all 51 tasks are provided in Table 8.

	Method	PARS Comparison		Noise Augmentation	Tasks w/ PARS $\geq$ w/o PARS
		w/o	w/		
Flow-based	FP	58.0	**66.9**	60.0	48/51
	SFP	49.8	**61.6**	46.7	44/51
Transformer-based	TP	57.4	**67.7**	59.3	46/51
	ACT	58.8	**68.2**	60.0	48/51
Diffusion-based	DP	47.6	**56.0**	46.2	46/51
	KDPE	53.5	**60.2**	56.7	42/51

low-dimensional information. Given the current observation, each policy predicts a 20-step action sequence. Each action is represented as $[\Delta q, g]$, where $\Delta q \in \mathbb{R}^7$ is a joint-space displacement and g is a gripper command. We use RLBench's `JointPosition` action mode with absolute joint-position targets. The predicted displacements are accumulated from the robot configuration at the beginning of the sequence to obtain these targets, while g is thresholded at 0.5 to issue a binary open/close command. For each task, we collect 100 expert demonstrations of a single task variation using RLBench's built-in demonstration generator and construct training pairs from each observation and its corresponding 20-step future action sequence. During evaluation, all 20 predicted actions are executed sequentially without re-observation or replanning. The policy then receives a new observation and predicts the next sequence. We report task success rate (%) over 50 evaluation episodes per task, each limited to 300 environment steps.

To evaluate PARS across action-sequence policy formulations, we consider three representative classes: flow-based, transformer-based, and diffusion-based policies. Their standard baselines are a flow-matching policy (FP) (Lipman et al., 2023), a transformer policy (TP) (Vaswani et al., 2017), and diffusion policy (DP) (Chi et al., 2025), respectively. We also include Action Chunking with Transformers (ACT) (Zhao et al., 2023), which models action sequences using a conditional VAE. To examine whether PARS complements existing robustness mechanisms, we evaluate Streaming Flow Policy (SFP) (Jiang et al., 2025), which learns a stabilizing flow field around demonstrated trajectories, and KDPE (Rosasco et al., 2025), which uses kernel density estimation to select a high-density action sequence from multiple DP candidates. Each trainable policy uses its respective BC objective, with the PARS instantiation described in Section 4.3. KDPE is applied only at inference time to the corresponding DP checkpoint.

5.2 Experimental Results

Improvement of closed-loop performance across different action-sequence methods. Table 1 compares task success rates with and without PARS across flow-based, transformer-based, and diffusion-based action-sequence policies. PARS improves the mean success rate for all evaluated methods and matches or exceeds the corresponding baseline on most tasks. These improvements span different policy formulations, suggesting that the benefits of PARS are not limited to a particular architecture or training objective. PARS also improves performance when combined with SFP and KDPE, suggesting that it can complement existing methods for improving robustness during execution. Overall, these results support the applicability of PARS across diverse action-sequence methods.

Benefit of Suffix Anchoring over Full-Target Supervision. We compare PARS with conventional noise augmentation to examine how supervision of perturbed observations affects closed-loop success. Both perturb proprioceptive observations during training, but conventional noise augmentation retains the original demonstrated action sequence as the target over the full prediction horizon, encouraging consistent predictions under perturbations. PARS instead anchors only the suffix to the demonstrated continuation, leaving the prefix free to produce corrective motion. As shown in Table 1, conventional noise augmentation yields modest gains or degrades performance, whereas PARS

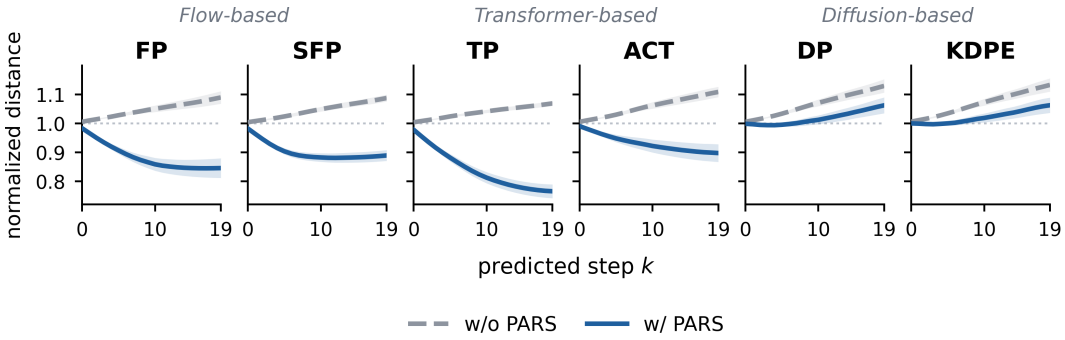

Figure 2: Normalized distance to the demonstrated configurations over the predicted action sequence after deviation. Values below 1.0 indicate recovery toward the demonstrations, whereas values above 1.0 indicate further deviation. Curves show the mean over 51 RLBench tasks with ± 1 standard error. PARS promotes recovery or reduces divergence across all evaluated policies.

improves mean success rates for all six policy methods. These results suggest that suffix anchoring yields better closed-loop performance than full-target supervision under input perturbations.

Recovery toward the Demonstrated Trajectory. To examine whether PARS's performance gains are associated with recovery, we evaluate predicted sequences at off-nominal observations reached by the unaugmented policies. Figure 2 shows, over predicted step k, the nearest-neighbor distance from each predicted joint configuration to the joint configurations in that task's demonstrations, normalized by its value at $k = 0$. The first $H_r = 10$ steps form the free recovery prefix and receive no direct augmented supervision. Without PARS, distances generally increase over the horizon. With PARS, the distances end below their initial levels for FP, SFP, TP, and ACT, although the FP and SFP curves flatten or rise slightly near the end. For DP and KDPE, the distances end above their initial levels, but PARS substantially reduces the final divergence relative to the unaugmented policies.

Kinematic behavior of the free recovery prefix. We examine whether leaving the recovery prefix without direct augmented supervision leads to excessively aggressive motion. Figure 3 compares the path-speed and path-acceleration distributions of the free prefix for the demonstrations, unaugmented policies, PARS, and conventional noise augmentation. Dashed lines indicate the joint-space speed and acceleration limits used to plan the demonstrations. Across most policy classes, PARS places less probability mass at high speeds and accelerations than noise augmentation, with particularly clear differences for FP, TP, and ACT. SFP also remains largely within the demonstrated motion range. For DP and KDPE, PARS reduces the high-acceleration tail relative to noise augmentation, although a noticeable fraction of predicted steps still exceeds the demonstration planning limit for acceleration. Thus, leaving the prefix free from direct augmented supervision does not necessarily produce aggressive motion and, in most cases, yields kinematics closer to those of the demonstrations than noise augmentation. Our feasibility analysis is conditional on the stated local-sensitivity and margin assumptions and therefore does not guarantee that every predicted prefix satisfies the planning limits.

6 Conclusion

We presented Perturbation-Augmented Recovery Supervision (PARS), a supervision scheme for improving recovery from covariate shift in action-sequence BC using existing demonstrations. PARS perturbs the proprioceptive observation and applies direct augmented supervision only to the suffix of the predicted action sequence, leaving the preceding prefix room for corrective motion. Our analysis characterizes the local sensitivity induced by suffix anchoring and derives conditions under which the free prefix remains close to demonstrated motion and satisfies the joint-space speed and acceleration limits used to plan the demonstrations. Across 51 RLBench tasks, PARS increases mean closed-loop success for all six evaluated policy methods and outperforms conventional full-target

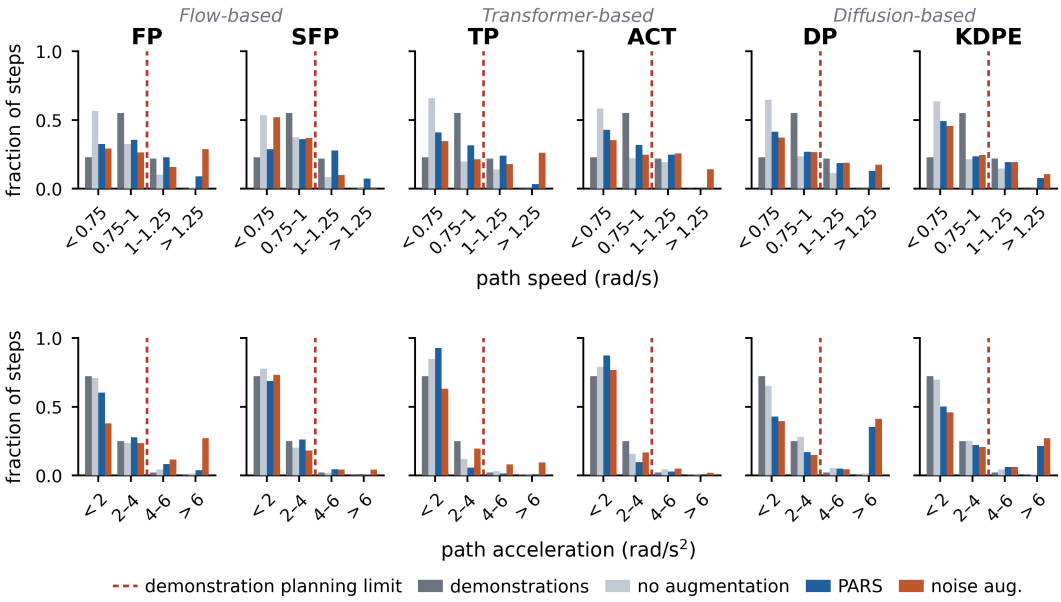

Figure 3: **Kinematics of the free recovery prefix.** Distributions of path speed ($\|\Delta q\|_2/\Delta t$) and path acceleration ($\|\Delta^2 q\|_2/\Delta t^2$) over the first $H_r = 10$ predicted steps. Bars show the fraction of steps in each range, averaged over 51 RLBench tasks. Dashed lines indicate the RLBench planning limits.

noise augmentation in mean success. Predicted trajectories approach demonstrated configurations for FP, SFP, TP, and ACT, while PARS reduces further divergence for DP and KDPE. These results suggest that recovery behavior can be encouraged using existing demonstrations without collecting explicit corrective trajectories or querying an expert during training.

Limitations. PARS uses task-specific perturbation settings that do not adapt to the stage of an episode or to individual proprioceptive dimensions. Our goal in this work is to demonstrate that perturbing proprioceptive observations in existing demonstrations and anchoring the predicted suffix can improve recovery, rather than to identify an optimal perturbation scheme for each task. Adaptive perturbations may better reflect how recovery demands change with context. We also use a fixed free recovery prefix of $H_r = 10$ steps within a 20-step prediction horizon. Its appropriate length may depend on the task and perturbation magnitude; larger deviations may require more steps before the predicted trajectory can approach demonstrated motion. Finally, our evaluation uses simulated RLBench tasks with low-dimensional observations and one variation per task. Extending the evaluation to visual observations and physical robots remains future work.

AI USE STATEMENT

In this work, we used generative AI tools to refine conceptual framing, provide feedback on research methodology and experimental design, and assist with translation of technical descriptions. We have not used generative AI tools to generate synthetic datasets, formulate mathematical claims, propose or refine hypotheses, implement methods, clean or reformat datasets, or interpret experimental results, and proof-related tasks and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for brainstorming, searching for and identifying relevant literature, summarizing existing literature, suggesting the structure of the manuscript, drafting parts of the manuscript, and editing the manuscript to improve clarity and readability. We have reviewed all AI-assisted work. All AI-assisted methodological suggestions and manuscript text were critically reviewed and revised by the authors, and cited literature and technical claims were independently checked against the original sources. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

Sections 4.1 and 4.3 define PARS and describe its instantiation across policy classes. Section 4.2 and Appendices B and C state the assumptions and provide the theoretical derivations. Section 5.1 describes the RLBench tasks, demonstration collection, action representation, and evaluation protocol. Appendices D.1–D.4 report the training settings, selected task-specific PARS settings, and protocols for the recovery and kinematic analyses. Table 8 reports success rates for all 51 tasks.

References

Rishabh Agrawal, Nathan Dahlin, Rahul Jain, and Ashutosh Nayyar. Markov balance satisfaction improves performance in strictly batch offline imitation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 15311–15319, 2025.

Ezra Ameperosa, Jeremy A Collins, Mrinal Jain, and Animesh Garg. Rocoda: Counterfactual data augmentation for data-efficient robot learning from demonstrations. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13250–13256. IEEE, 2025.

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.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

Sunshine Jiang, Xiaolin Fang, Nicholas Roy, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Siddharth Ancha. Streaming flow policy: Simplifying diffusion/flow-matching policies by treating action trajectories as flow trajectories. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 238–257. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/jiang25a.html.

Liyiming Ke, Jingqiang Wang, Tapomayukh Bhattacharjee, Byron Boots, and Siddhartha Srinivasa. Grasping with chopsticks: Combating covariate shift in model-free imitation learning for fine manipulation. In *2021 IEEE international conference on robotics and automation (ICRA)*, pp. 6185–6191. IEEE, 2021.

Liyiming Ke, Yunchu Zhang, Abhay Deshpande, Siddhartha Srinivasa, and Abhishek Gupta. Ccil: Continuity-based data augmentation for corrective imitation learning. In *International Conference on Learning Representations*, volume 2024, pp. 35813–35831, 2024.

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

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

Filippo Lazzati, Kyle Stachowicz, William Chen, Alberto Maria Metelli, Andrew Wagenmaker, and Sergey Levine. Why does action chunking improve behavioral cloning performance in robotic control? *arXiv preprint arXiv:2608.02547*, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Shaunak A Mehta, Yusuf Umut Ciftci, Balamurugan Ramachandran, Somil Bansal, and Dylan P Losey. Stable-bc: Controlling covariate shift with stable behavior cloning. *IEEE Robotics and Automation Letters*, 10(2):1952–1959, 2025.

Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2): 1–179, 2018.

Bumgeun Park, Hyukjun Yang, and Donghwan Lee. Refinement-based flow policy optimization. *arXiv preprint arXiv:2609.15123*, 2026.

Andrea Rosasco, Federico Ceola, Giulia Pasquale, and Lorenzo Natale. Kdpe: A kernel density estimation strategy for diffusion policy trajectory selection. *arXiv preprint arXiv:2508.10511*, 2025.

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.

Seokin Seo, Byung-Jun Lee, Jongmin Lee, HyeongJoo Hwang, Hongseok Yang, and Kee-Eung Kim. Mitigating covariate shift in behavioral cloning via robust stationary distribution correction. *Advances in Neural Information Processing Systems*, 37:109177–109201, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jin Wang, Pengcheng He, Ke Jiang, and Xiaoyang Tan. Koopman-assisted trajectory synthesis: A data augmentation framework for offline imitation learning. In *International Conference on Learning Representations*, volume 2026, pp. 20637–20655, 2026.

Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*, 2016.

Thomas T Zhang, Daniel Pfrommer, Chaoyi Pan, Nikolai Matni, and Max Simchowitz. Action chunking and data augmentation yield exponential improvements in behavior cloning for continuous spaces. In *International Conference on Learning Representations*, volume 2026, pp. 126689–126742, 2026.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

A Why We Focus on Action-Sequence Policies

Single-step behavior cloning is known to be particularly susceptible to compounding errors in closed-loop execution, where small prediction errors can shift the policy away from the state distribution covered by expert demonstrations (Ross et al., 2011). Recent studies further show that action-sequence prediction can substantially improve behavior-cloning performance by reducing compounding errors and exploiting temporal information across multiple actions (Lazzati et al., 2026; Zhang et al., 2026).

Consistent with these observations, the single-step BC-based baselines showed limited closed-loop performance on our manipulation tasks, as summarized in Table 2. Because these methods exhibit low closed-loop performance in our setting, we focus the main evaluation on action-sequence policies, which provide stronger baselines for evaluating the recovery and robustness improvements introduced by PARS.

Table 2: Closed-loop success rates (%) of single-step BC-based baselines, averaged over 51 tasks.

Vanilla BC	CCIL	DrilDICE	MBIL	KATS	Stable-BC
0.5	1.2	1.6	0.8	1.7	9.8

B Sensitivity Analyses across Policy Classes

Section 4.3 describes how the suffix-only supervision of PARS is instantiated across the policy classes considered in our experiments. We provide here the corresponding local sensitivity analyses. The exact form of the induced regularization depends on the quantity supervised by each training objective. Direct trajectory regression admits the Jacobian regularization derived in Section 4.2, whereas objectives based on L_1 reconstruction or generative-field regression require separate extensions.

B.1 Direct Chunk Prediction with a Squared Loss

We first derive the approximation in eq. (6), which applies directly to chunk-prediction policies trained with a squared trajectory loss. For a demonstration sample $(o_t, \mathbf{a}_t)$, PARS perturbs only the proprioceptive component of the observation:

$$\tilde{o}_t = (o_t^{\text{prop}} + \epsilon, o_t^{\text{env}}), \tag{11}$$

where

$$\mathbb{E}[\epsilon] = 0, \qquad \Sigma_\epsilon = \text{Cov}(\epsilon). \tag{12}$$

The augmented loss considered in the analysis is

$$\mathcal{L}_{\text{aug}}(\epsilon) = \frac{1}{2} \left\| [f_\theta(\tilde{o}_t)]_{H_r:H} - [\mathbf{a}_t]_{H_r:H} \right\|^2. \tag{13}$$

Assuming that f_θ is sufficiently smooth around o_t, a second-order Taylor expansion with respect to the proprioceptive input gives

$$\begin{aligned} [f_\theta(\tilde{o}_t)]_{H_r:H} =& [f_\theta(o_t)]_{H_r:H} + J_{\text{suf}}\epsilon \\ &+ \frac{1}{2} Q_{\text{suf}}(\epsilon, \epsilon) + O(\|\epsilon\|^3), \end{aligned} \tag{14}$$

where

$$J_{\text{suf}} = \left. \frac{\partial [f_\theta(o)]_{H_r:H}}{\partial o^{\text{prop}}} \right|_{o=o_t}, \tag{15}$$

and $Q_{\text{suf}}(\epsilon, \epsilon)$ collects the second-order curvature terms of the predicted suffix.

Let

$$r_t = [f_\theta(o_t)]_{H_r:H} - [\mathbf{a}_t]_{H_r:H} \tag{16}$$

denote the suffix prediction residual at the original observation. Substituting eq. (14) into eq. (13) and retaining terms up to second order yields

$$\begin{aligned}\mathcal{L}_{\text{aug}}(\epsilon) =& \frac{1}{2}\|r_t\|^2 + r_t^\top J_{\text{suf}}\epsilon \\ &+ \frac{1}{2}\epsilon^\top J_{\text{suf}}^\top J_{\text{suf}}\epsilon \\ &+ \frac{1}{2} r_t^\top Q_{\text{suf}}(\epsilon,\epsilon) + O(\|\epsilon\|^3).\end{aligned} \tag{17}$$

Taking expectation over the perturbation eliminates the first-order term because $\mathbb{E}[\epsilon] = 0$. Moreover,

$$\mathbb{E}_\epsilon\left[\epsilon^\top J_{\text{suf}}^\top J_{\text{suf}}\epsilon\right] = \operatorname{tr}\left(J_{\text{suf}}\Sigma_\epsilon J_{\text{suf}}^\top\right). \tag{18}$$

Hence,

$$\begin{aligned}\mathbb{E}_\epsilon\left[\mathcal{L}_{\text{aug}}(\epsilon)\right] =& \frac{1}{2}\|r_t\|^2 + \frac{1}{2}\operatorname{tr}\left(J_{\text{suf}}\Sigma_\epsilon J_{\text{suf}}^\top\right) \\ &+ \frac{1}{2} r_t^\top \mathbb{E}_\epsilon\left[Q_{\text{suf}}(\epsilon,\epsilon)\right] + O\big(\mathbb{E}_\epsilon\|\epsilon\|^3\big).\end{aligned} \tag{19}$$

When the trained policy closely matches the demonstrated suffix, r_t is small. For sufficiently small perturbations, the residual-dependent curvature term and higher-order terms can therefore be neglected, giving

$$\mathbb{E}_\epsilon\left[\mathcal{L}_{\text{aug}}(\epsilon)\right] \approx \frac{1}{2}\|r_t\|^2 + \frac{1}{2}\operatorname{tr}\left(J_{\text{suf}}\Sigma_\epsilon J_{\text{suf}}^\top\right), \tag{20}$$

which gives eq. (6).

For independent uniform perturbations $\epsilon_i \sim \mathcal{U}(-\delta_e, \delta_e)$,

$$\Sigma_\epsilon = \frac{\delta_e^2}{3} I, \tag{21}$$

and therefore

$$\frac{1}{2}\operatorname{tr}\left(J_{\text{suf}}\Sigma_\epsilon J_{\text{suf}}^\top\right) = \frac{\delta_e^2}{6}\|J_{\text{suf}}\|_F^2. \tag{22}$$

Thus, the augmented squared loss locally penalizes the sensitivity of the predicted suffix to proprioceptive perturbations. Because the augmented loss contains only the trajectory suffix, no corresponding Jacobian penalty is directly introduced for the trajectory prefix. This analysis directly covers the squared-loss chunk-prediction policy considered in our experiments.

B.2 Direct Chunk Prediction with an L_1 Loss

We next consider a direct chunk-prediction policy trained using an L_1 reconstruction loss, as in ACT. Only the reconstruction term is affected by the PARS suffix restriction; the KL regularization term is left unchanged.

Using the same suffix residual r_t and Jacobian J_{suf} defined above, a first-order Taylor expansion gives

$$[f_\theta(\tilde{o}_t)]_{H_r:H} = [f_\theta(o_t)]_{H_r:H} + J_{\text{suf}}\epsilon + O(\|\epsilon\|^2). \tag{23}$$

The augmented L_1 reconstruction loss is therefore

$$\mathcal{L}_{\text{aug}}^{L_1}(\epsilon) = \left\|r_t + J_{\text{suf}}\epsilon + O(\|\epsilon\|^2)\right\|_1. \tag{24}$$

In the near-zero-residual regime, $r_t \approx 0$. For sufficiently small perturbations, neglecting the residual and higher-order terms gives

$$\mathcal{L}_{\text{aug}}^{L_1}(\epsilon) \approx \left\|J_{\text{suf}}\epsilon\right\|_1. \tag{25}$$

Taking expectation over the perturbation yields

$$\mathbb{E}_\epsilon\left[\mathcal{L}_{\text{aug}}^{L_1}(\epsilon)\right] \approx \mathbb{E}_\epsilon\left[\left\|J_{\text{suf}}\epsilon\right\|_1\right]. \tag{26}$$

Thus, although the exact form of the induced regularizer differs from the quadratic Jacobian penalty obtained under a squared loss, the augmented L_1 reconstruction likewise penalizes the local sensitivity of the supervised suffix to proprioceptive perturbations. The trajectory prefix does not receive an analogous invariance penalty from the augmented reconstruction loss.

B.3 STREAMING FLOW POLICIES

We next analyze SFP, for which the augmented objective supervises a velocity field rather than directly supervising the predicted trajectory. Let $v_\theta(a, \tau \mid o)$ denote the observation-conditioned velocity field, where $\tau \in [0, 1]$ denotes normalized trajectory time. As described in Section 4.3, for perturbed samples the learned flow is first evolved up to τ_r using the same stepwise integration as at inference, and suffix supervision is then applied from the state reached at τ_r. We analyze below the resulting velocity-regression objective over the supervised suffix.

For clarity, we consider a fixed demonstration trajectory ξ and omit the outer expectation over the demonstration dataset. Let $v_\xi(a, \tau)$ denote the target conditional velocity field associated with ξ. The suffix velocity-regression loss can be written as

$$\mathcal{L}^{\mathrm{SFP}}_{\mathrm{aug}}(\epsilon) = \mathbb{E}_{\tau,a}\left[\frac{1}{2}\left\|v_\theta(a, \tau \mid \tilde{o}_t) - v_\xi(a, \tau)\right\|^2\right], \qquad \tau \geq \tau_r, \tag{27}$$

where the expectation is taken over the sampling distribution used by the SFP training objective restricted to the supervised suffix.

Define

$$r_v(a, \tau) = v_\theta(a, \tau \mid o_t) - v_\xi(a, \tau) \tag{28}$$

and

$$J_v(a, \tau) = \left.\frac{\partial v_\theta(a, \tau \mid o)}{\partial o^{\mathrm{prop}}}\right|_{o=o_t}. \tag{29}$$

A second-order Taylor expansion with respect to the proprioceptive perturbation gives

$$\begin{aligned} v_\theta(a, \tau \mid \tilde{o}_t) =& v_\theta(a, \tau \mid o_t) + J_v(a, \tau)\epsilon \\ &+ \frac{1}{2}Q_v(a, \tau; \epsilon, \epsilon) + O(\|\epsilon\|^3), \end{aligned} \tag{30}$$

where Q_v collects the second-order curvature terms.

Proceeding as in Appendix B.1, taking expectation over a zero-mean perturbation gives

$$\begin{aligned} \mathbb{E}_\epsilon\left[\mathcal{L}^{\mathrm{SFP}}_{\mathrm{aug}}(\epsilon)\right] =& \mathcal{L}^{\mathrm{SFP}}_{\mathrm{aug}}(0) \\ &+ \frac{1}{2}\mathbb{E}_{\tau,a}\left[\mathrm{tr}\left(J_v \Sigma_\epsilon J_v^\top\right)\right] \\ &+ \frac{1}{2}\mathbb{E}_{\tau,a}\left[r_v^\top \mathbb{E}_\epsilon\left[Q_v(\epsilon, \epsilon)\right]\right] \\ &+ O\big(\mathbb{E}_\epsilon\|\epsilon\|^3\big). \end{aligned} \tag{31}$$

When the learned velocity field closely matches its target over the supervised suffix, the residual-dependent curvature term becomes small. For sufficiently small perturbations,

$$\begin{aligned} \mathbb{E}_\epsilon\left[\mathcal{L}^{\mathrm{SFP}}_{\mathrm{aug}}(\epsilon)\right] \approx& \mathcal{L}^{\mathrm{SFP}}_{\mathrm{aug}}(0) \\ &+ \frac{1}{2}\mathbb{E}_{\tau,a}\left[\mathrm{tr}\left(J_v \Sigma_\epsilon J_v^\top\right)\right]. \end{aligned} \tag{32}$$

Thus, the PARS augmentation locally penalizes the conditioning sensitivity of the velocity field over the supervised suffix.

Relation to trajectory sensitivity. Because SFP supervises a velocity field rather than the final trajectory, the field-level regularization above does not directly equal the trajectory Jacobian regularization in eq. (6). To characterize their relationship, let $a_o(\tau)$ denote the generated trajectory under observation o, satisfying

$$\frac{da_o(\tau)}{d\tau} = v_\theta(a_o(\tau), \tau \mid o). \tag{33}$$

Define its sensitivity to the proprioceptive observation as

$$S(\tau) = \frac{\partial a_o(\tau)}{\partial o^{\mathrm{prop}}}. \tag{34}$$

Differentiating eq. (33) gives

$$\frac{dS(\tau)}{d\tau} = A(\tau)S(\tau) + B(\tau), \tag{35}$$

where

$$A(\tau) = \left.\frac{\partial v_\theta(a, \tau \mid o)}{\partial a}\right|_{a=a_o(\tau)} \tag{36}$$

and

$$B(\tau) = \left.\frac{\partial v_\theta(a, \tau \mid o)}{\partial o^{\text{prop}}}\right|_{a=a_o(\tau)}. \tag{37}$$

Let $\Phi(\tau, s)$ denote the state-transition matrix associated with $A(\tau)$. We assume that the learned flow is locally contractive over the relevant suffix region, such that for some $\kappa > 0$,

$$\|\Phi(\tau, s)\| \le e^{-\kappa(\tau - s)}, \qquad \tau \ge s \ge \tau_r. \tag{38}$$

This is a local assumption on the learned dynamics and does not imply global contraction of the policy.

The variation-of-constants formula yields

$$S(\tau) = \Phi(\tau, \tau_r)S(\tau_r) + \int_{\tau_r}^{\tau} \Phi(\tau, s)B(s)\, ds. \tag{39}$$

Taking norms and using eq. (38),

$$\begin{aligned}\|S(\tau)\| \le e^{-\kappa(\tau-\tau_r)}\|S(\tau_r)\| \\ + \int_{\tau_r}^{\tau} e^{-\kappa(\tau-s)}\|B(s)\|\, ds.\end{aligned} \tag{40}$$

The integral term satisfies

$$\int_{\tau_r}^{\tau} e^{-\kappa(\tau-s)}\|B(s)\|\, ds \le \frac{1 - e^{-\kappa(\tau-\tau_r)}}{\kappa} \sup_{s\in[\tau_r,\tau]} \|B(s)\|. \tag{41}$$

Since

$$\frac{1 - e^{-\kappa(\tau-\tau_r)}}{\kappa} \le \tau - \tau_r \le 1, \tag{42}$$

we obtain

$$\left\| \frac{\partial a_o(\tau)}{\partial o^{\text{prop}}} - \Phi(\tau, \tau_r)\frac{\partial a_o(\tau_r)}{\partial o^{\text{prop}}} \right\| \le \sup_{s\in[\tau_r,\tau]} \left\| \frac{\partial v_\theta(a_o(s), s \mid o)}{\partial o^{\text{prop}}} \right\|. \tag{43}$$

The second term on the left represents sensitivity inherited from the beginning of the supervised suffix after attenuation by the locally contractive flow. The right-hand side captures the contribution of the observation-conditioned velocity field over the suffix. Therefore, PARS does not directly regularize the complete trajectory-suffix Jacobian in SFP; rather, it regularizes one source of trajectory sensitivity through the velocity field.

This analysis is local. In particular, eq. (32) controls the velocity-field sensitivity under the SFP training distribution, whereas eq. (43) evaluates this sensitivity along a generated trajectory. Their connection therefore assumes that the generated suffix remains in a region where the learned field is locally controlled by the augmented training objective.

B.4 Joint Generative Policies

We finally consider joint flow-matching and diffusion policies. Unlike SFP, these models generate the entire action sequence jointly along a generative axis that is distinct from sequence time. For the purpose of analysis, we represent the effect of the refinement-based optimization as inducing a surrogate recovery trajectory $\hat{\mathbf{a}}_t(\epsilon)$ that is consistent with the suffix-level objective.

The following analysis characterizes the corresponding generative-field sensitivity using this surrogate representation. Let s denote the generative time and ω denote the auxiliary randomness used by

the training objective, such as a base sample or sampled noise. Given the surrogate trajectory $\hat{\mathbf{a}}_t(\epsilon)$, let

$$z_s(\epsilon) = z_s\left(\hat{\mathbf{a}}_t(\epsilon), \omega\right) \tag{44}$$

denote the intermediate noisy or interpolated sample supplied to the model, and let

$$u_s(\epsilon) = u_s\left(\hat{\mathbf{a}}_t(\epsilon), \omega\right) \tag{45}$$

denote the corresponding velocity, noise, or other regression target.

For compactness, define

$$\mathcal{F}_\theta(\epsilon) = F_\theta\left(z_s(\epsilon), s \mid \tilde{o}_t\right) \tag{46}$$

and

$$\mathcal{U}(\epsilon) = u_s(\epsilon), \tag{47}$$

where F_θ denotes the field predicted by the generative model. The augmented generative loss takes the generic squared form

$$\mathcal{L}^{\text{gen}}_{\text{aug}}(\epsilon) = \mathbb{E}_{s,\omega}\left[\frac{1}{2}\left\|\mathcal{F}_\theta(\epsilon) - \mathcal{U}(\epsilon)\right\|^2\right]. \tag{48}$$

Define the residual at the unperturbed observation as

$$r_g = \mathcal{F}_\theta(0) - \mathcal{U}(0), \tag{49}$$

and define the total perturbation sensitivities

$$J_F = \left.\frac{\partial \mathcal{F}_\theta(\epsilon)}{\partial \epsilon}\right|_{\epsilon=0}, \qquad J_U = \left.\frac{\partial \mathcal{U}(\epsilon)}{\partial \epsilon}\right|_{\epsilon=0}. \tag{50}$$

Here, J_F is a total derivative: it includes both the direct dependence of the generative field on the perturbed observation and the dependence induced through the intermediate generative input $z_s(\epsilon)$.

For sufficiently smooth $\mathcal{F}_\theta$ and $\mathcal{U}$, first-order Taylor expansions give

$$\mathcal{F}_\theta(\epsilon) = \mathcal{F}_\theta(0) + J_F\epsilon + O(\|\epsilon\|^2) \tag{51}$$

and

$$\mathcal{U}(\epsilon) = \mathcal{U}(0) + J_U\epsilon + O(\|\epsilon\|^2). \tag{52}$$

Therefore,

$$\mathcal{F}_\theta(\epsilon) - \mathcal{U}(\epsilon) = r_g + (J_F - J_U)\epsilon + O(\|\epsilon\|^2). \tag{53}$$

For a small zero-mean perturbation and when the generative regression residual r_g is small, taking expectation in eq. (48) yields the local approximation

$$\begin{aligned}\mathbb{E}_\epsilon\left[\mathcal{L}^{\text{gen}}_{\text{aug}}(\epsilon)\right] \approx\ & \mathcal{L}^{\text{gen}}_{\text{aug}}(0) \\ & + \frac{1}{2}\mathbb{E}_{s,\omega}\left[\operatorname{tr}\left((J_F - J_U)\Sigma_\epsilon(J_F - J_U)^\top\right)\right].\end{aligned} \tag{54}$$

Unlike direct trajectory regression, the induced regularizer therefore does not generally penalize J_F alone. Instead, it penalizes the difference between the perturbation response of the learned generative field and that of the surrogate generative target.

The selective structure of PARS appears through the construction of the surrogate trajectory. The suffix of $\hat{\mathbf{a}}_t(\epsilon)$ is encouraged to remain consistent with the demonstrated future, whereas its prefix is allowed to vary in response to the perturbed observation. Under a locally suffix-preserving surrogate construction and the usual generative corruption or interpolation process, the target associated with the suffix is approximately insensitive to the perturbation:

$$J_{U,\text{suf}} \approx 0. \tag{55}$$

In this regime, the suffix component of eq. (54) approximately reduces to

$$\frac{1}{2}\mathbb{E}_{s,\omega}\left[\operatorname{tr}\left(J_{F,\text{suf}}\Sigma_\epsilon J_{F,\text{suf}}^\top\right)\right]. \tag{56}$$

Thus, the generative training objective locally penalizes sensitivity associated with the supervised suffix.

In contrast, the surrogate prefix is allowed to vary with the perturbation. Consequently, the intermediate generative input $z_s(\epsilon)$ and, depending on the particular parameterization, the regression target $u_s(\epsilon)$ may also vary with ϵ. The prefix contribution in eq. (54) therefore penalizes the total perturbation-response mismatch $J_{F,\mathrm{pre}} - J_{U,\mathrm{pre}}$. Because J_F includes both the direct dependence on the perturbed observation and the dependence induced through $z_s(\epsilon)$, this term should not in general be interpreted as forcing the direct observation-conditioning sensitivity of the prefix toward zero. This distinction is consistent with the design of PARS: the recovery prefix is allowed to respond to the perturbed state, while the later portion of the trajectory is encouraged to return toward the demonstrated future.

As with SFP, this field-level regularization should not be interpreted as direct regularization of the complete Jacobian of the final generated trajectory. For flow-matching models, the final trajectory is obtained by integrating the learned velocity field, while for diffusion policies it is obtained through a sequence of denoising transitions. Under local regularity of the corresponding generation process, perturbations of the learned field induce locally bounded perturbations of the generated sample. The field-level regularization in eq. (56) can therefore be viewed as an indirect mechanism for controlling one source of suffix sensitivity in the final generated trajectory.

The exact forms of z_s, u_s, and J_U depend on the particular flow-matching or diffusion parameterization. Accordingly, eq. (54) is intended as a generic local characterization of surrogate-target generative training rather than as a parameterization-specific identity.

C Local Analysis of the Free Prefix

We provide the detailed derivations of the local-consistency bound in eq. (9) and the trajectory-planning feasibility condition in eq. (10).

Local consistency with demonstrated motion. Let $\mathbf{q}_\theta^{\mathrm{pre}}(o) = (q_{\theta,0}(o), \dots, q_{\theta,H_r-1}(o))$ denote the desired-configuration prefix generated by the policy. We define the trajectory-level metric

$$d_{\mathrm{traj}}(\mathbf{q}, \mathbf{q}') = \max_{0 \le k < H_r} \|q_k - q'_k\|_2 . \tag{57}$$

Let $\mathcal{S}_{\mathrm{data}}^{\mathrm{pre}}$ denote the set of configuration prefixes represented in the demonstrations, and define the distance to this set as

$$\mathrm{dist}(\mathbf{q}, \mathcal{S}_{\mathrm{data}}^{\mathrm{pre}}) = \inf_{\boldsymbol{\xi} \in \mathcal{S}_{\mathrm{data}}^{\mathrm{pre}}} d_{\mathrm{traj}}(\mathbf{q}, \boldsymbol{\xi}) . \tag{58}$$

For any $\boldsymbol{\xi} \in \mathcal{S}_{\mathrm{data}}^{\mathrm{pre}}$, the triangle inequality gives

$$\begin{aligned} d_{\mathrm{traj}}(\mathbf{q}_\theta^{\mathrm{pre}}(\tilde{o}_t), \boldsymbol{\xi}) \le\ & d_{\mathrm{traj}}(\mathbf{q}_\theta^{\mathrm{pre}}(\tilde{o}_t), \mathbf{q}_\theta^{\mathrm{pre}}(o_t)) \\ & + d_{\mathrm{traj}}(\mathbf{q}_\theta^{\mathrm{pre}}(o_t), \boldsymbol{\xi}) . \end{aligned} \tag{59}$$

Taking the infimum over $\boldsymbol{\xi} \in \mathcal{S}_{\mathrm{data}}^{\mathrm{pre}}$ and applying the assumptions in eq. (8) yields

$$\begin{aligned} \mathrm{dist}(\mathbf{q}_\theta^{\mathrm{pre}}(\tilde{o}_t), \mathcal{S}_{\mathrm{data}}^{\mathrm{pre}}) &\le d_{\mathrm{traj}}(\mathbf{q}_\theta^{\mathrm{pre}}(\tilde{o}_t), \mathbf{q}_\theta^{\mathrm{pre}}(o_t)) \\ &\quad + \mathrm{dist}(\mathbf{q}_\theta^{\mathrm{pre}}(o_t), \mathcal{S}_{\mathrm{data}}^{\mathrm{pre}}) \\ &\le \eta + L_q \|\epsilon\|_2 . \end{aligned} \tag{60}$$

Since each perturbed coordinate satisfies $|\epsilon_i| \le \delta_e$ and the perturbation has dimension d_e,

$$\|\epsilon\|_2 \le \sqrt{d_e}\, \delta_e . \tag{61}$$

Substituting this bound into eq. (60) gives

$$\mathrm{dist}(\mathbf{q}_\theta^{\mathrm{pre}}(\tilde{o}_t), \mathcal{S}_{\mathrm{data}}^{\mathrm{pre}}) \le \eta + L_q \sqrt{d_e}\, \delta_e , \tag{62}$$

which recovers eq. (9).

Feasibility under trajectory-planning limits. We next show how the same local-sensitivity assumption bounds changes in the joint-space speed and acceleration of the free prefix. For a fixed control interval Δt, define

$$v_{\theta,k}(o) = \frac{q_{\theta,k+1}(o) - q_{\theta,k}(o)}{\Delta t}, \qquad 0 \le k < H_r - 1, \tag{63}$$

and

$$a_{\theta,k}(o) = \frac{q_{\theta,k+2}(o) - 2q_{\theta,k+1}(o) + q_{\theta,k}(o)}{\Delta t^2}, \qquad 0 \le k < H_r - 2. \tag{64}$$

For convenience, define the perturbation-induced change in the predicted configuration at step k as

$$\Delta q_{\theta,k} = q_{\theta,k}(\tilde{o}_t) - q_{\theta,k}(o_t). \tag{65}$$

From eqs. (8) and (57),

$$\|\Delta q_{\theta,k}\|_2 \le L_q \|\epsilon\|_2 \tag{66}$$

for every step in the free prefix.

The corresponding change in velocity is

$$v_{\theta,k}(\tilde{o}_t) - v_{\theta,k}(o_t) = \frac{\Delta q_{\theta,k+1} - \Delta q_{\theta,k}}{\Delta t}. \tag{67}$$

Therefore,

$$\begin{aligned} \|v_{\theta,k}(\tilde{o}_t) - v_{\theta,k}(o_t)\|_2 &\le \frac{\|\Delta q_{\theta,k+1}\|_2 + \|\Delta q_{\theta,k}\|_2}{\Delta t} \\ &\le \frac{2L_q\|\epsilon\|_2}{\Delta t} \\ &\le \frac{2L_q\sqrt{d_e}\,\delta_e}{\Delta t}. \end{aligned} \tag{68}$$

Similarly,

$$a_{\theta,k}(\tilde{o}_t) - a_{\theta,k}(o_t) = \frac{\Delta q_{\theta,k+2} - 2\Delta q_{\theta,k+1} + \Delta q_{\theta,k}}{\Delta t^2}, \tag{69}$$

which gives

$$\begin{aligned} \|a_{\theta,k}(\tilde{o}_t) - a_{\theta,k}(o_t)\|_2 &\le \frac{\|\Delta q_{\theta,k+2}\|_2 + 2\|\Delta q_{\theta,k+1}\|_2 + \|\Delta q_{\theta,k}\|_2}{\Delta t^2} \\ &\le \frac{4L_q\|\epsilon\|_2}{\Delta t^2} \\ &\le \frac{4L_q\sqrt{d_e}\,\delta_e}{\Delta t^2}. \end{aligned} \tag{70}$$

Let $v^{\max}$ and $a^{\max}$ denote the joint-space speed and acceleration limits used for trajectory planning. As stated in the main text, suppose that the nominal prefix satisfies these limits with positive margins m_v and m_a, such that

$$\|v_{\theta,k}(o_t)\|_2 \le v^{\max} - m_v, \qquad \|a_{\theta,k}(o_t)\|_2 \le a^{\max} - m_a \tag{71}$$

for all valid steps k.

Using eq. (68),

$$\begin{aligned} \|v_{\theta,k}(\tilde{o}_t)\|_2 &\le \|v_{\theta,k}(o_t)\|_2 + \|v_{\theta,k}(\tilde{o}_t) - v_{\theta,k}(o_t)\|_2 \\ &\le v^{\max} - m_v + \frac{2L_q\sqrt{d_e}\,\delta_e}{\Delta t}. \end{aligned} \tag{72}$$

Hence,

$$\frac{2L_q\sqrt{d_e}\,\delta_e}{\Delta t} \le m_v \tag{73}$$

is sufficient to ensure

$$\|v_{\theta,k}(\tilde{o}_t)\|_2 \le v^{\max}. \tag{74}$$

Likewise, from eq. (70),

$$\begin{aligned}\|a_{\theta,k}(\tilde{o}_t)\|_2 &\le \|a_{\theta,k}(o_t)\|_2 + \|a_{\theta,k}(\tilde{o}_t) - a_{\theta,k}(o_t)\|_2 \\ &\le a^{\max} - m_a + \frac{4L_q\sqrt{d_e}\,\delta_e}{\Delta t^2}.\end{aligned} \tag{75}$$

Thus,

$$\frac{4L_q\sqrt{d_e}\,\delta_e}{\Delta t^2} \le m_a \tag{76}$$

is sufficient to ensure

$$\|a_{\theta,k}(\tilde{o}_t)\|_2 \le a^{\max}. \tag{77}$$

Together, these conditions recover eq. (10).

Interpretation of the assumptions. The analysis is conditional on three local properties. First, the nominal BC prediction is assumed to remain close to configuration trajectories represented in the demonstrations. Second, a small proprioceptive perturbation is assumed to induce only a proportionally small change in the predicted configuration prefix. Third, the nominal prefix is assumed to satisfy the joint-space speed and acceleration limits used for trajectory planning with positive margins. These assumptions are not guaranteed by the suffix-anchoring objective itself.

Scope of the analysis. The two results characterize complementary properties of the free recovery prefix. The local-consistency bound limits how far the perturbed prefix can move from configuration trajectories represented in the demonstrations, whereas the feasibility condition ensures that its joint-space speed and acceleration remain within the specified trajectory-planning limits when the perturbation is sufficiently small. Neither result requires the free prefix to reproduce a particular demonstrated trajectory, leaving room for corrective motion. However, the analysis is local and does not guarantee task success, collision avoidance, or full dynamic feasibility under arbitrarily large perturbations. In particular, satisfying these joint-space speed and acceleration limits alone does not guarantee perfect tracking by the low-level controller.

D Experimental Settings

D.1 Training and Policy Settings

Table 3: Common policy-training and success-rate evaluation settings.

Setting	Value
Training steps	40,000 gradient steps
Optimizer	AdamW
Learning rate	3×10^{-4}
Learning-rate schedule	Cosine decay to 10^{-6}
Batch size	128
Random seed	0
Policy training	Separate policy trained for each task
Evaluation episodes	50 per task
Maximum episode length	300 environment steps

Table 4: Policy-specific settings. Architecture and policy hyperparameters are selected for the corresponding unaugmented baseline and held fixed across the unaugmented, PARS, and noise-augmentation conditions.

Method	Architecture	Generation / inference	Method-specific settings
FP	Width 144, 4 layers, 8 heads	10 ODE steps	—
SFP	MLP field, width 256	3 Euler substeps per action	$\sigma_0 = \sigma_1 = 0.1$
TP	Width 152, 4 layers, 8 heads	—	—
ACT	Width 256, 4 layers, 8 heads	—	CVAE latent dim. 32, KL weight 10
DP	U-Net $(128, 256, 512)$; per-task encoder width (256 default)	100 training / 20 inference steps	Squared-cosine noise schedule
KDPE	DP checkpoint	32 candidate chunks	KDE bandwidth 0.2; $20\times$ bandwidth for gripper dimension

For DP, we use the squared-cosine noise schedule rather than the linear schedule because the latter is calibrated for $T = 1000$ and leaves $\bar{\alpha}$ far from zero with 100 diffusion steps, causing sampling to start outside the noise range encountered during training.

D.2 PARS and Augmentation Settings

Table 5 summarizes the PARS and augmentation settings used in our experiments. For each perturbed sample, the perturbation scale δ_e is sampled from the task-specific range $[\sigma_{\mathrm{lo}}, \sigma_{\mathrm{hi}}]$, after which the proprioceptive perturbation is drawn as $\epsilon \sim \mathcal{U}(-\delta_e, \delta_e)$. Each training sample uses the augmented objective with task-specific probability p and the standard BC objective otherwise. The expected objective is therefore $(1-p)\mathcal{L}_{\mathrm{BC}} + p\mathcal{L}_{\mathrm{aug}}$; after factoring out $(1-p)$, the relative augmented-loss coefficient in Eq. 4 is $\lambda_{\mathrm{aug}} = p/(1-p)$. For each task, the augmentation setting is determined by grid search. The selected PARS settings used for Table 1 are reported in Tables 6 and 7.

Table 5: PARS and augmentation settings. Settings marked "per task" are selected separately for each task.

Setting	Applies to	Value
Prediction horizon H	All methods	20
Free recovery prefix H_r	PARS	10
	Noise augmentation	0
Perturbation probability p	PARS, noise augmentation	Per task
Perturbation scale range	PARS, noise augmentation	Per task
Inner refinement steps	FP, DP	Per task
Inner refinement step size	FP, DP	0.1
Augmentation warm-up	DP	Per task

DP uses an initial warm-up period before augmentation is enabled, as applying refinement to samples from the poorly trained policy early in training was empirically less stable in preliminary runs. For SFP, the stepwise integration required to obtain the state at τ_r is performed on 16 perturbed samples per batch because sequential integration is computationally more expensive.

D.3 Recovery and Free-Prefix Analysis

Figures 2 and 3 analyze the predicted action sequences at off-nominal observations reached during closed-loop execution. Both analyses use the predicted action sequence rather than the subsequently

executed rollout, since the quantities of interest describe the motion proposed by the policy from an off-nominal observation.

Off-nominal observation collection. The off-nominal observations are not synthetically generated. For each trainable policy class, we collect observations from 20 rollouts per task of the corresponding unaugmented policy. The configurations visited at action-sequence boundaries are ranked by their nearest-neighbor distance to the demonstrated joint configurations in z-scored joint space. We retain the top quartile and query all compared variants at the same recorded observations. The comparison is therefore paired at the level of individual observations, and the analyzed states reflect deviations actually reached during closed-loop execution. For KDPE, the observations corresponding to DP are used, since KDPE shares the DP checkpoint and differs only in its inference-time selection rule.

Kinematic metrics and reference limits. We use a control period of $\Delta t = 0.05\,\mathrm{s}$. The RLBench demonstration planner constrains the rate of progress along the planned joint-space path rather than the velocity of each individual joint. We therefore measure the Euclidean norm over the seven arm joints, $\|\Delta q\|_2/\Delta t$ for speed and $\|\Delta^2 q\|_2/\Delta t^2$ for acceleration. Using the maximum individual-joint value would instead compare against a constraint that was not used to generate the demonstrations.

The dashed reference lines in Figure 3 correspond to the trajectory-planning limits used by the RLBench demonstration generator: $1.0\,\mathrm{rad/s}$ for joint-space speed and $4.0\,\mathrm{rad/s^2}$ for joint-space acceleration. These values are planning constraints on the demonstration trajectories rather than hardware limits of the robot. Numerical differentiation at $20\,\mathrm{Hz}$ produces small apparent exceedances around the speed limit: 22% of demonstrated steps lie slightly above $1.0\,\mathrm{rad/s}$, whereas only 2.7% exceed $4.0\,\mathrm{rad/s^2}$ in acceleration. We therefore separate the speed range immediately above the planning limit from the more aggressive region above $1.25\,\mathrm{rad/s}$.

Recovery-distance normalization. For each predicted step k, we compute the predicted joint configuration as $\hat{q}_k = q_{\mathrm{init}} + \sum_{i=0}^{k} \Delta q_i$, where q_{init} is the joint configuration at the observed action-sequence boundary, and measure its nearest-neighbor distance to the demonstrated joint configurations. The distance is divided by its value at $k = 0$ to make the curves comparable across tasks with different absolute distance scales. Consequently, a normalized value of 1.0 corresponds to the reference distance at $k = 0$, values below 1.0 indicate movement toward the demonstrated configurations, and values above 1.0 indicate further deviation. Values are first averaged across observations within each task and then across the 51 tasks.

D.4 Selected Augmentation Settings for Each Task

Tables 6 and 7 report the task-specific augmentation settings used for the PARS results in Table 1. DP and KDPE share the same settings because KDPE is applied only at inference time to the corresponding DP checkpoint.

Table 6: Task-specific PARS settings for FP, SFP, and TP. Range denotes σ_{lo}–σ_{hi}, p is the perturbation probability, and K is the number of inner refinement steps.

RLBench Task	**FP**			**SFP**		**TP**	
	Range	p	K	Range	p	Range	p
basketball in hoop	0.05–0.15	0.65	5	0.03–0.08	0.65	0.04–0.12	0.5
beat the buzz	0.02–0.06	0.35	5	0.03–0.08	0.35	0.01–0.04	0.15
change clock	0.03–0.1	0.5	25	0.05–0.08	0.65	0.04–0.12	0.65
close box	0.03–0.1	0.5	25	0.05–0.08	0.65	0.02–0.06	0.5
close door	0.02–0.06	0.5	5	0.05–0.15	0.3	0.05–0.15	0.5
close drawer	0.03–0.1	0.5	25	0.04–0.12	0.5	0.03–0.1	0.5
close fridge	0.02–0.06	0.35	5	0.05–0.15	0.65	0.02–0.04	0.35
close grill	0.05–0.08	0.5	40	0.04–0.12	0.5	0.03–0.1	0.5
close jar	0.03–0.1	0.5	5	0.03–0.08	0.5	0.03–0.1	0.5
close laptop lid	0.03–0.08	0.5	25	0.03–0.1	0.3	0.02–0.04	0.35
close microwave	0.03–0.08	0.5	25	0.03–0.08	0.5	0.02–0.06	0.35
lamp off	0.03–0.1	0.5	15	0.03–0.1	0.65	0.05–0.15	0.35
lamp on	0.03–0.1	0.5	15	0.05–0.15	0.3	0.05–0.15	0.5
lift numbered block	0.03–0.1	0.5	25	0.03–0.1	0.65	0.05–0.15	0.5
meat off grill	0.03–0.1	0.5	25	0.03–0.08	0.5	0.02–0.04	0.15
meat on grill	0.01–0.04	0.5	15	0.01–0.03	0.3	0.02–0.06	0.15
open box	0.02–0.06	0.35	5	0.03–0.1	0.5	0.03–0.1	0.25
open door	0.03–0.1	0.5	5	0.05–0.15	0.5	0.02–0.06	0.35
open drawer	0.05–0.15	0.5	25	0.03–0.1	0.3	0.05–0.08	0.65
open fridge	0.03–0.1	0.5	15	0.05–0.08	0.65	0.03–0.08	0.35
open grill	0.04–0.12	0.5	25	0.04–0.12	0.5	0.01–0.04	0.5
open jar	0.02–0.06	0.35	5	0.03–0.08	0.65	0.03–0.1	0.5
open microwave	0.03–0.1	0.5	25	0.03–0.08	0.15	0.05–0.08	0.5
open oven	0.01–0.04	0.5	15	0.03–0.08	0.5	0.02–0.06	0.35
open washing machine	0.03–0.1	0.5	25	0.03–0.1	0.65	0.01–0.04	0.25
open wine bottle	0.03–0.08	0.5	25	0.03–0.08	0.65	0.01–0.04	0.25
phone on base	0.03–0.1	0.5	25	0.04–0.12	0.35	0.02–0.06	0.15
pick and lift	0.03–0.1	0.5	15	0.03–0.08	0.65	0.03–0.08	0.35
pick and lift small	0.03–0.1	0.5	5	0.05–0.08	0.5	0.04–0.12	0.65
pick up cup	0.03–0.1	0.5	25	0.05–0.08	0.5	0.03–0.08	0.5
press switch	0.03–0.1	0.5	15	0.04–0.12	0.5	0.03–0.08	0.5
push button	0.03–0.1	0.5	5	0.03–0.08	0.65	0.05–0.08	0.65
push buttons	0.03–0.1	0.5	5	0.01–0.03	0.3	0.01–0.04	0.25
put item in drawer	0.04–0.12	0.5	15	0.03–0.1	0.65	0.02–0.06	0.5
put knife on chopping board	0.03–0.1	0.5	25	0.05–0.08	0.65	0.02–0.04	0.25
put money in safe	0.03–0.1	0.5	25	0.04–0.12	0.5	0.03–0.1	0.15
put rubbish in bin	0.03–0.1	0.5	15	0.04–0.12	0.5	0.02–0.06	0.15
put umbrella in umbrella stand	0.01–0.04	0.5	15	0.04–0.12	0.5	0.03–0.1	0.25
reach and drag	0.03–0.1	0.5	40	0.01–0.03	0.3	0.04–0.12	0.5
reach target	0.02–0.06	0.35	5	0.03–0.08	0.5	0.03–0.1	0.5
slide block to target	0.03–0.1	0.5	5	0.03–0.08	0.5	0.04–0.12	0.5
stack blocks	0.03–0.1	0.5	5	0.03–0.08	0.5	0.03–0.1	0.5
take item out of drawer	0.03–0.1	0.65	25	0.05–0.08	0.5	0.02–0.06	0.35
take lid off saucepan	0.05–0.15	0.65	5	0.04–0.12	0.5	0.03–0.1	0.5
take umbrella out of umbrella stand	0.03–0.1	0.5	15	0.05–0.08	0.5	0.03–0.1	0.5
toilet seat down	0.02–0.06	0.35	5	0.03–0.08	0.65	0.03–0.08	0.35
toilet seat up	0.03–0.1	0.5	15	0.03–0.08	0.65	0.02–0.06	0.5
turn oven on	0.03–0.1	0.5	5	0.03–0.08	0.5	0.04–0.12	0.5
turn tap	0.05–0.15	0.65	5	0.03–0.1	0.65	0.05–0.15	0.5
unplug charger	0.03–0.1	0.5	5	0.04–0.12	0.65	0.05–0.08	0.65
water plants	0.03–0.1	0.5	25	0.03–0.1	0.65	0.05–0.08	0.65

Table 7: Task-specific PARS settings for ACT and DP/KDPE. Range denotes σ_{lo}–σ_{hi}; K is the number of inner refinement steps. Warm-up is reported in gradient steps, with "k" denoting $1{,}000$.

RLBench Task	**ACT**		**DP / KDPE**			
	Range	p	Range	p	K	Warm-up
basketball in hoop	0.02–0.06	0.35	0.03–0.1	0.5	25	15k
beat the buzz	0.03–0.08	0.1	0.05–0.08	0.5	5	15k
change clock	0.01–0.04	0.25	0.03–0.1	0.5	15	15k
close box	0.02–0.04	0.25	0.03–0.1	0.5	25	15k
close door	0.02–0.04	0.35	0.05–0.15	0.5	15	15k
close drawer	0.05–0.15	0.65	0.03–0.1	0.5	25	15k
close fridge	0.03–0.1	0.5	0.03–0.1	0.65	25	20k
close grill	0.02–0.04	0.5	0.04–0.12	0.5	25	15k
close jar	0.03–0.08	0.5	0.03–0.1	0.5	25	15k
close laptop lid	0.03–0.1	0.65	0.03–0.1	0.5	25	20k
close microwave	0.01–0.04	0.15	0.03–0.1	0.65	5	15k
lamp off	0.01–0.04	0.5	0.03–0.08	0.5	5	15k
lamp on	0.04–0.12	0.15	0.03–0.1	0.5	15	15k
lift numbered block	0.02–0.04	0.15	0.03–0.08	0.65	25	15k
meat off grill	0.01–0.03	0.25	0.05–0.08	0.65	25	15k
meat on grill	0.01–0.03	0.25	0.03–0.08	0.5	25	15k
open box	0.02–0.04	0.35	0.03–0.08	0.5	25	15k
open door	0.01–0.04	0.25	0.03–0.08	0.5	25	15k
open drawer	0.02–0.04	0.05	0.04–0.12	0.5	15	20k
open fridge	0.03–0.08	0.65	0.04–0.12	0.5	25	15k
open grill	0.02–0.04	0.25	0.03–0.08	0.5	15	20k
open jar	0.05–0.08	0.5	0.03–0.1	0.5	25	15k
open microwave	0.02–0.06	0.1	0.04–0.12	0.5	25	15k
open oven	0.02–0.06	0.65	0.05–0.08	0.65	25	15k
open washing machine	0.03–0.1	0.5	0.03–0.1	0.5	25	15k
open wine bottle	0.03–0.08	0.25	0.03–0.08	0.5	5	20k
phone on base	0.03–0.1	0.15	0.04–0.12	0.5	25	15k
pick and lift	0.05–0.08	0.35	0.04–0.12	0.5	15	15k
pick and lift small	0.03–0.08	0.5	0.03–0.1	0.5	25	15k
pick up cup	0.03–0.08	0.35	0.03–0.08	0.5	5	15k
press switch	0.02–0.06	0.35	0.04–0.12	0.5	15	20k
push button	0.05–0.08	0.5	0.03–0.1	0.5	25	20k
push buttons	0.03–0.08	0.2	0.03–0.1	0.5	25	20k
put item in drawer	0.03–0.1	0.5	0.03–0.08	0.5	5	15k
put knife on chopping board	0.04–0.12	0.25	0.05–0.08	0.65	15	15k
put money in safe	0.01–0.03	0.15	0.04–0.12	0.5	15	15k
put rubbish in bin	0.03–0.1	0.15	0.04–0.12	0.5	15	20k
put umbrella in umbrella stand	0.03–0.08	0.35	0.04–0.12	0.5	15	20k
reach and drag	0.01–0.04	0.35	0.03–0.1	0.5	25	15k
reach target	0.03–0.1	0.5	0.03–0.1	0.5	25	15k
slide block to target	0.05–0.15	0.5	0.03–0.08	0.5	15	15k
stack blocks	0.03–0.1	0.5	0.03–0.1	0.5	25	15k
take item out of drawer	0.03–0.1	0.25	0.04–0.12	0.5	15	20k
take lid off saucepan	0.02–0.04	0.25	0.03–0.08	0.5	5	15k
take umbrella out of umbrella stand	0.02–0.06	0.5	0.03–0.1	0.5	15	15k
toilet seat down	0.01–0.04	0.35	0.03–0.1	0.65	5	20k
toilet seat up	0.03–0.1	0.25	0.03–0.1	0.5	5	15k
turn oven on	0.01–0.03	0.35	0.05–0.08	0.5	5	15k
turn tap	0.05–0.08	0.65	0.03–0.1	0.5	25	20k
unplug charger	0.02–0.06	0.65	0.04–0.12	0.5	15	20k
water plants	0.01–0.04	0.25	0.03–0.1	0.5	25	15k

E FULL RESULTS

Table 8: Per-task success rate (%) on all 51 RLBench tasks.

RLBench Task	Flow-based		Transformer-based		Diffusion-based	
	FP	SFP	TP	ACT	DP	KDPE
basketball in hoop	94→**96**	96→**96**	94→**96**	96→**96**	94→84	96→**98**
beat the buzz	66→**74**	48→**58**	76→72	78→**90**	28→**52**	38→**50**
change clock	6→**30**	8→**22**	28→**34**	18→**22**	6→**24**	20→**26**
close box	88→**88**	78→**90**	90→**96**	92→**92**	68→**88**	82→**94**
close door	12→**18**	16→**16**	4→**38**	12→**24**	10→**14**	10→4
close drawer	96→**100**	98→**100**	100→**100**	100→**100**	100→**100**	100→**100**
close fridge	88→**98**	78→**96**	90→**100**	94→**96**	94→**94**	98→92
close grill	94→92	90→**90**	90→**92**	88→**98**	74→**80**	74→**88**
close jar	0→**0**	0→**0**	0→**0**	0→**0**	0→**0**	0→**0**
close laptop lid	98→96	100→92	96→**96**	98→**98**	76→**98**	82→**100**
close microwave	100→**100**	88→**98**	96→**98**	100→**100**	100→94	100→**100**
lamp off	84→**92**	72→**86**	84→**90**	80→**88**	88→82	86→**90**
lamp on	80→**90**	80→78	78→**86**	86→82	48→**80**	56→**88**
lift numbered block	44→**60**	32→**44**	22→**44**	24→**52**	42→36	30→**34**
meat off grill	96→**98**	78→**90**	94→**96**	92→**100**	70→**74**	90→74
meat on grill	72→**90**	70→**90**	78→**82**	52→**92**	80→**96**	80→**98**
open box	98→**100**	86→84	90→**100**	98→**100**	78→**88**	90→**96**
open door	86→**96**	82→**90**	94→**100**	88→**88**	74→**76**	88→**92**
open drawer	64→**66**	58→**58**	54→**76**	66→**68**	38→**60**	48→**68**
open fridge	10→**30**	2→**18**	16→**32**	14→**34**	6→**10**	10→4
open grill	52→**58**	46→**56**	56→**74**	38→**40**	12→**30**	6→**18**
open jar	0→**2**	0→**0**	0→**0**	0→**0**	0→**0**	0→**0**
open microwave	92→**92**	96→**98**	92→**94**	96→**96**	62→**78**	80→**82**
open oven	18→**30**	10→**20**	18→**58**	36→**68**	18→**20**	22→**32**
open washing machine	34→**56**	18→**44**	46→**64**	46→**58**	4→**18**	24→**38**
open wine bottle	66→**68**	38→**74**	52→**82**	50→**84**	52→**62**	38→**66**
phone on base	58→**66**	22→20	44→20	44→**58**	30→**52**	34→**44**
pick and lift	66→**82**	32→**86**	54→**78**	44→**90**	46→**58**	56→**68**
pick and lift small	0→**0**	2→0	0→**2**	0→**2**	0→**0**	0→**0**
pick up cup	90→88	74→**86**	78→**88**	82→**90**	82→**92**	94→**94**
press switch	34→**78**	18→**62**	50→**82**	68→**82**	38→**56**	62→60
push button	86→**94**	82→**88**	82→**98**	80→**98**	92→**96**	94→**98**
push buttons	96→**100**	100→74	90→44	100→68	98→**100**	100→98
put item in drawer	0→**6**	0→**4**	0→**10**	6→**8**	0→**2**	0→**0**
put knife on chopping board	10→**34**	4→**24**	12→10	8→**32**	20→**20**	22→18
put money in safe	82→**92**	68→**78**	72→**78**	82→**86**	42→**64**	38→**88**
put rubbish in bin	50→**62**	8→**66**	50→40	64→52	22→**30**	36→**36**
put umbrella in umbrella stand	10→**26**	14→**26**	12→**18**	6→**24**	2→**12**	10→**14**
reach and drag	58→**64**	64→**76**	56→**62**	52→**68**	58→**70**	68→**80**
reach target	100→**100**	98→**100**	98→**100**	100→**100**	100→**100**	100→**100**
slide block to target	66→**90**	58→**96**	82→**98**	80→**86**	64→**76**	74→**88**
stack blocks	0→**0**	0→**0**	0→**0**	0→**0**	0→**0**	0→**0**
take item out of drawer	0→**10**	0→**10**	4→**30**	4→**10**	0→**2**	4→0
take lid off saucepan	84→**96**	82→**98**	94→**98**	84→**92**	94→80	90→**96**
take umbrella out of umbrella stand	92→**98**	88→**98**	86→**100**	90→**94**	92→**96**	100→**100**
toilet seat down	90→**96**	100→98	96→**100**	92→**98**	92→**94**	90→**96**
toilet seat up	82→**86**	54→**86**	38→**90**	74→**92**	46→**72**	66→**72**
turn oven on	18→**52**	4→**28**	24→**72**	28→**46**	6→**24**	16→**52**
turn tap	54→**70**	54→**64**	78→**82**	74→**84**	26→**54**	56→**68**
unplug charger	24→**30**	6→**30**	40→**66**	30→**70**	14→**14**	12→**12**
water plants	70→**74**	38→**56**	50→**86**	64→**80**	42→**56**	58→56