REGUIDE: FROM TEST-TIME GUIDANCE TO SELF-IMPROVING DIFFUSION POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Behavior-cloned diffusion policies are expressive but remain vulnerable to covariate shift: small deviations from demonstrated states can compound into task failure. Existing methods address this either by expanding the training distribution through expert corrections or synthetic augmentation, or by steering a frozen policy at test time with guidance from a learned model. The former can be expensive or assumption-dependent, while the latter discards the corrected trajectories after execution. We introduce **ReGuide**, a self-improving framework that treats guided rollouts as reusable data. ReGuide first uses **Phase-Conditioned Guidance (PCG)** to generate corrective rollouts: it constructs phase-specific latent targets, applies guidance only in the drifted-but-recoverable regime, and guides through the estimated clean action. Successful guided rollouts are then absorbed back into the policy through **ReGuide-FT**, which fine-tunes the current checkpoint, or **ReGuide-FS**, which retrains from scratch on the augmented dataset; the two can also be composed and iterated. On Robomimic Can, Square, Transport, and Tool Hang, absorbing success-filtered rollouts lifts base-policy success by 1.2–$7.2\times$ from 10–80 demonstrations. At matched training-data volume, rollouts collected under PCG yield a better policy than unguided rollouts: by 0.089 across three training seeds on Can, and at every rollout count on Square, so guidance improves the quality of self-collected data rather than merely its quantity. PCG's role is to generate training data rather than to improve a single rollout: at test time it matches LPB and DynaGuide on the three tasks where no guidance method separates from the base policy, and beats both on the one where any does. On LIBERO, fine-tuning a pretrained multi-task checkpoint on guided rollouts improves it on the three tasks it handles worst. Code is available at anonymous.4open.science/r/reguide-anon-27B4.

1 INTRODUCTION

Imitation learning-based policies are vulnerable to covariate shift: small errors move the robot away from demonstrated states, after which the policy is queried on observations outside its training distribution and errors compound over time (Ross et al., 2011). This problem is especially acute for long-horizon manipulation, where an early deviation can make later subtasks unrecoverable. Diffusion policies improve the expressivity by modeling multimodal action distributions (Chi et al., 2023), but they do not remove this distribution-shift problem. A common solution is to expand the training distribution. DAgger-style methods collect expert corrections at learner-visited states (Ross et al., 2011; Hu et al., 2025; Xu et al., 2025), while data-augmentation methods synthesize additional examples using task structure, simulation, or local dynamics assumptions (Ke et al., 2024; Ankile et al., 2024; Jia et al., 2023). These methods address the right failure mode, but require either continued expert access or assumptions about which perturbations preserve task correctness. In contrast, recent test-time guidance methods such as DynaGuide (Du & Song, 2025) and Latent Policy Barrier (LPB) (Sun & Song, 2025) steer a frozen diffusion policy using gradients from a learned dynamics model, improving recovery without retraining or new demonstrations. However, these methods discard the guided trajectory after execution.

In this paper, we propose a simple yet highly effective solution for this problem: **use guided rollouts as training data for iterative self-improvement**. A successful guided rollout captures precisely the information missing from the original demonstrations: where the learned policy tends to drift, and

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.

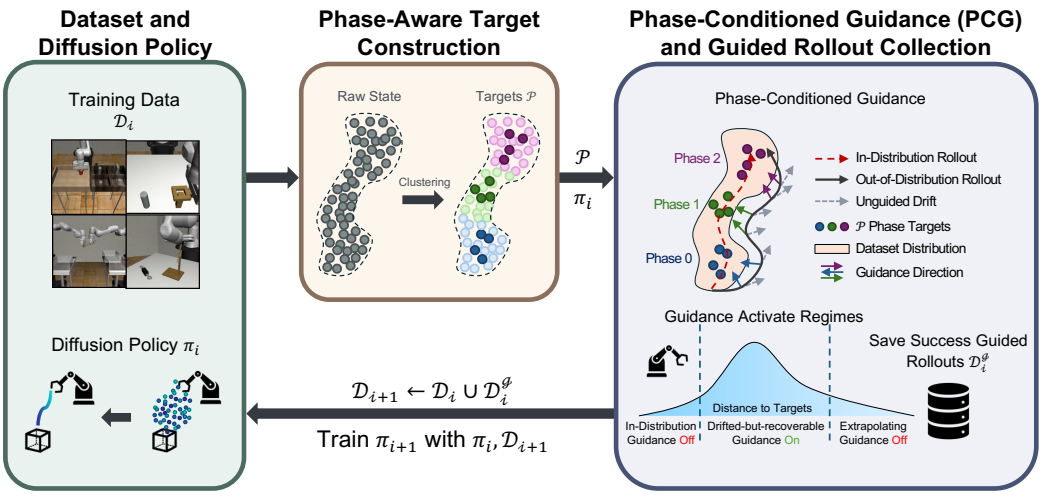


Figure 1: **ReGuide overview.** At iteration i: starting with π_i and $\mathcal{D}_i$, construct phase targets $\mathcal{P}$ by clustering the latent states, roll out π_i with phase-conditioned guidance, and merge the successful guided rollouts $\mathcal{D}_i^g$ into $\mathcal{D}_{i+1}$, from which π_{i+1} is obtained.

which corrective actions return the system toward task completion. Recycling such rollouts provides on-policy recovery data using only a trajectory-level success signal, rather than expensive per-state expert relabeling as used in Dagger-style algorithms.

We introduce **ReGuide**, a self-improving diffusion-policy framework. Turning guided rollouts into useful training data requires three choices that keep self-generated data from degrading the policy, and **Phase-Conditioned Guidance (PCG)** makes them. Long-horizon tasks are phase-structured, so a single global target set can pull a rollout toward the wrong stage or collapse valid behavior modes; PCG builds phase-specific latent targets from demonstrations and keeps several per phase. Dynamics gradients are reliable only near the data distribution, perturbing correct actions too close to the expert manifold and extrapolating too far from it; PCG gates guidance to the drifted-but-recoverable regime. And prior guidance differentiates the objective with respect to the noisy denoising iterate rather than the clean action chunk the dynamics model scores; PCG takes the gradient through the estimated clean action itself. Successful guided rollouts are then recycled into policy training through **ReGuide-FT**, which fine-tunes the current checkpoint on a mixture of demonstrations and guided rollouts, and **ReGuide-FS**, which retrains a fresh policy on the augmented dataset. Crucially, the updated policy can be rolled out again under the same guidance mechanism to generate a new batch of higher-quality data, yielding an iterative *rollout–collect–train* loop that improves the policy without additional expert demonstrations until the self-generated data reaches its performance ceiling.

Our central claim is that guidance-generated rollouts are a better data source than unguided rollouts for self-improving a diffusion policy. On Robomimic (Mandlekar et al., 2022), absorbing self-collected, success-filtered rollouts lifts base-policy success by 1.2–7.2×; the direct test of our claim is the comparison against filtered behavior cloning at matched data volume, where guided rollouts win at every count on Can, Square and Transport under both absorption modes. Iterating the loop improves the policy further. PCG itself is a data-generation mechanism rather than a test-time method: without fine-tuning it matches LPB and DynaGuide where no guidance method separates from the base policy, and beats both on Transport, the one task where any does. On LIBERO (Liu et al., 2023), the same absorption step improves a pretrained multi-task checkpoint on its three most difficult tasks.

2 Related Work

Diffusion policies and test-time guidance. We build on Diffusion Policy (Chi et al., 2023) as the base visuomotor policy. Recent work steers pretrained diffusion policies at inference time without modifying their weights: DynaGuide (Du & Song, 2025) uses gradients from an external dynamics model to steer toward a set of goal states, Latent Policy Barrier (LPB) (Sun & Song, 2025) treats expert latent embeddings as an implicit in-distribution barrier, PPGuide (Wang et al., 2026) learns a performance predictor for guidance, and Generative Predictive Control (GPC) (Qi et al., 2026) performs model-based look-ahead. These methods improve execution-time behavior, but the guided

trajectories are discarded after each episode. ReGuide instead uses guidance as a data-generation mechanism: successful guided rollouts are added back to the training set to improve the policy. ReGuide also differs in the guidance design: we adapt Manifold Preserving Guided Diffusion (He et al., 2024) to differentiate through the estimated clean action, and replace global targets with phase-conditioned target sets and a two-threshold gate. LPB is the closest baseline: we inherit its dynamics-model architecture and latent distance cost (Section 3), and its target—the demonstration distribution itself—is what PCG makes phase-local, whereas DynaGuide steers toward a separately specified set of goal states.

Self-improvement and data augmentation for imitation learning. Covariate shift in behavior cloning is commonly addressed by expanding the learner's training distribution. Interactive methods such as DAgger (Ross et al., 2011) and RaC (Hu et al., 2025) collect new supervision at policy-visited states, but require continued expert access. Other approaches augment the data without further expert intervention (Ke et al., 2024; Ankile et al., 2024; Jia et al., 2023; Mehta et al., 2025). The closest in spirit is CCIL (Ke et al., 2024), which learns a Lipschitz-regularized dynamics model from demonstrations and generates corrective action labels near demonstrated states. ReGuide differs in both mechanism and loop structure: it uses test-time dynamics gradients to recover rollouts that have already drifted from the demonstrations, filters successful trajectories, and iteratively retrains or fine-tunes the policy on the resulting on-policy recovery data.

Improving diffusion policies with environment interaction. ReGuide rolls out, scores the outcome, and updates the policy, so it shares its loop structure with reinforcement learning for diffusion policies—DPPO (Ren et al., 2025) and IN-RIL (Gao et al., 2025) at 10^6–10^7 environment steps from low-dimensional state, and the sample-efficient DSRL (Wagenmaker et al., 2025), PA-RL (Sobol Mark et al., 2024) and OGPO (Patil et al., 2026) at tens to hundreds of episodes, ReGuide's own budget. The same loop has been developed for flow-based policies (Park et al., 2025; Lv et al., 2025; Zhang et al., 2026; Yi et al., 2026). We therefore separate ReGuide from this literature not by interaction budget but by what the loop consumes: every method above fits a value function from reward-labeled interaction; ReGuide fits none, takes one binary success bit per trajectory, and updates by supervised behavior cloning. It also starts from 10–80 demonstrations against the 300 used throughout this literature. This is why filtered behavior cloning—the same demonstrations, the same supervision, the same absorption procedure—is our in-family baseline. Appendix A.1 gives the budgets side by side and the full discussion. Closest in spirit is AdaptDiffuser (Liang et al., 2023), an offline-RL planner that also guides a diffusion model and fine-tunes on the result, but steers with a differentiable reward over low-dimensional states; ReGuide has none and absorbs only executed, verified rollouts.

Other related work. Extended discussion of related work is provided in Appendix A.

3 Preliminaries

Diffusion policy. A *diffusion policy* (Chi et al., 2023) parameterizes π_θ as a conditional denoising diffusion model over action chunks. It conditions on a window of T_o past observations $O_t = (o_{t-T_o+1}, \ldots, o_t)$ and generates T_p future actions $A_t = (a_t, \ldots, a_{t+T_p-1})$, of which the first few are executed before re-planning; the denoiser conditions on a latent embedding $z_t = h(O_t)$ rather than the raw observation window.

Sampling starts from $A_t^K \sim \mathcal{N}(0, I)$ and denoises through K steps; at step k a learned noise predictor $\epsilon_\theta(A_t^k, k, z_t)$ gives

$$A_t^{k-1} \;=\; \frac{1}{\sqrt{\alpha_k}}\left(A_t^k \;-\; \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}}\,\epsilon_\theta(A_t^k, k, z_t)\right) \;+\; \sigma_k\,\xi, \qquad \xi \sim \mathcal{N}(0, I), \tag{1}$$

with $\alpha_k, \bar{\alpha}_k, \sigma_k$ set by the noise schedule and $\xi = 0$ at $k = 1$, so iterating to $k = 1$ yields the executed chunk $A_t = A_t^0$. **Guidance for diffusion policies.** Classifier guidance (Dhariwal & Nichol, 2021; Ho & Salimans, 2021) biases the reverse process by adding the gradient of a differentiable objective to the predicted noise. For a diffusion policy the generated variable is the action chunk, so at step k,

$$\hat{\epsilon}(A_t^k) \;=\; \epsilon_\theta(A_t^k) \;-\; \eta\sqrt{1-\bar{\alpha}_k}\,\nabla_{A_t^k}\,\delta(A_t^k), \tag{2}$$

where η controls the guidance strength and δ penalizes drift from the expert manifold, so each denoising step is nudged toward actions whose predicted consequences score well under δ. Appendix B gives the image-generation form this specializes, and the related guidance methods for control.

Dynamics model. Given the latent state z_t and a candidate action chunk A_t, a dynamics model returns the latent state the environment is expected to reach, $\hat{z}_{t+T_p} = d_\phi(z_t, A_t)$. We adopt the DINO-WM architecture (Zhou et al., 2025) for d_ϕ, following LPB (Sun & Song, 2025); it shares the visual encoder h with the diffusion policy, so latent-space distances are meaningful as a guidance signal, and is trained on rollouts (both successful and failed) of that policy. Following LPB (Sun & Song, 2025) we take $\delta(A_t^k) = \left\| d_\phi(z_t, A_t^k) - z^* \right\|^2$ for a target latent z^*, which shifts the guidance signal from "does this action look like an expert action" to "does this action lead to an expert-like future state."

4 Our Approach: ReGuide

ReGuide turns guidance into a data-generation mechanism for a behavior-cloned diffusion policy without collecting additional expensive (human) expert demonstrations. It consists of three coupled components. First, *phase-aware target construction* (Section 4.1) decomposes long-horizon demonstrations into temporally ordered phases and constructs a set of latent targets for each phase, so that "in-distribution" behavior is defined locally. Second, *phase-conditioned guidance and data collection* (Section 4.2) guides the diffusion denoising process toward the appropriate phase targets, but only in regimes where the predicted future latent is drifted from the demonstrations while still within the model's reliable operating range. Third, *iterative self-improvement learning* (Section 4.3) filters successful guided rollouts, merges them with the existing dataset, and updates the policy through either fine-tuning or retraining. The updated policy then generates a new distribution of guided rollouts, yielding an iterative rollout–collect–train loop driven by self-generated, success-filtered data.

Algorithm 1 ReGuide

Require: Demonstrations $\mathcal{D}_{\text{demo}}$, iterations N
1: $\mathcal{D}_1 \leftarrow \mathcal{D}_{\text{demo}}$
2: Train π_1 on $\mathcal{D}_1$
3: Build $\{\mathcal{P}_j\}, \{\mathcal{F}_j\}$ via Alg. 2 ▷ §4.1
4: **for** $i = 1$ to N **do**
5: Collect $\mathcal{D}_i^g$ via Alg. 3 ▷ §4.2
6: $\mathcal{D}_{i+1} \leftarrow \mathcal{D}_i \cup \mathcal{D}_i^g$
7: Train π_{i+1} from $\mathcal{D}_{i+1}, \pi_i$ ▷ §4.3
8: **end for**
9: **return** π_{N+1}

Figure 1 summarizes the loop; Algorithm 1 states it.

4.1 Phase-Aware Target Construction

Guidance needs something to steer toward. This step builds it from the demonstrations: a set of latent targets, grouped by phase of the task. A task moves through distinct phases, and the phase matters: guiding a rollout toward a point far in the future says little about what the policy should do now, whereas a target from the current subtask gives a direction that is useful at the present state.

We recover the phases by clustering the demonstration latents, but two properties of the data make naive clustering fail. The latents are aliased across time—a gripper approaching an object and the same gripper retreating after a failed grasp produce nearly identical observations—so position alone does not identify the phase. We therefore cluster *temporally augmented* features $[\, z_t^{\text{vis}},\ z_t^{\text{pro}},\ \Delta z_t^{\text{vis}},\ \Delta z_t^{\text{pro}} \,]$, appending one-step finite differences $\Delta z_t = z_t - z_{t-1}$ (set to zero at $t = 0$), so that the direction of motion supplies what position cannot. And expert behavior within a phase is multimodal, so a single target per phase would treat valid modes as deviations and guide against them; we therefore keep M targets per phase, aggregated with a soft minimum.

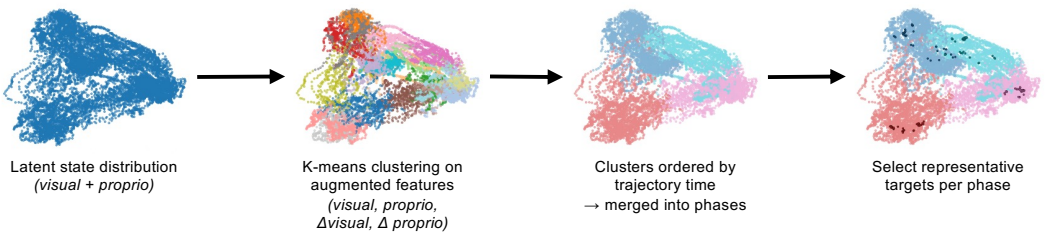


Figure 2: **Phase-aware target construction.** Latent states are clustered using temporally augmented features, ordered by trajectory time, and grouped into phases. Representative centroids from each phase define the target sets used for phase-conditioned guidance.

We reduce these features to d dimensions with PCA and partition them into k clusters with k-means. Clustering carries no temporal ordering, so we sort the clusters by the mean timestep of their members and group the sorted sequence into N_p contiguous phases, taking M representative latents from each as its target set $\mathcal{P}_j = \{p_{j,1}, \ldots, p_{j,M}\}$. The distance from a predicted latent z to phase j is a soft minimum over that set, $\mathcal{L}(z, \mathcal{P}_j) = -\tau \log \sum_{m=1}^{M} \exp\left(-\frac{|z-p_{j,m}|^2}{\tau}\right)$, where τ controls the sharpness; this preserves multimodality within the phase while leaving the objective differentiable. We also store the empirical distance distribution $\mathcal{F}_j = \{\mathcal{L}(z_t, \mathcal{P}_j) : z_t \text{ belongs to phase } j\}$, which Section 4.2 uses to calibrate the decision to guide to the local geometry of the current phase rather than to one global threshold. The target sets are built once from $\mathcal{D}_{\text{demo}}$, before the loop of Section 4.3 begins; Figure 2 illustrates the construction and Algorithm 2 in the Appendix the formal description.

4.2 Phase-Conditioned Guidance (PCG) and Rollout Data Collection

This step puts the phase targets to work at test time: as the diffusion policy denoises an action, we measure how far the predicted next latent lies from the current phase's targets and push the action toward them. Algorithm 3 states the procedure, and the successful rollouts it returns become training data in Section 4.3. Prior diffusion-policy guidance differentiates the objective with respect to the noisy iterate A_t^k, and handles the resulting mismatch with the dynamics model in different ways: LPB (Sun & Song, 2025) evaluates d_ϕ at an estimate of the clean action but still takes the gradient through A_t^k, while DynaGuide (Du & Song, 2025) trains its dynamics model on noisy actions so that A_t^k can be passed directly. We take a third route, following MPGD (He et al., 2024): we estimate the clean action and apply the guidance gradient *to that estimate*, so both the input and the differentiated variable lie in the distribution d_ϕ was trained on. Eqs. (3)–(5) give the three steps—estimate, guide, convert back to a noise prediction.

$$\hat{A}_t^0 = \frac{A_t^k - \sqrt{1-\bar{\alpha}_k}\,\epsilon_\theta}{\sqrt{\bar{\alpha}_k}}, \tag{3}$$

$$\tilde{A}_t^0 \leftarrow \hat{A}_t^0 - \eta\sqrt{1-\bar{\alpha}_k}\,\nabla_{\hat{A}_t^0}\,\mathcal{L}\big(d_\phi(z_t,\ \hat{A}_t^0),\ \mathcal{P}_j\big), \tag{4}$$

$$\hat{\epsilon} = \frac{A_t^k - \sqrt{\bar{\alpha}_k}\,\tilde{A}_t^0}{\sqrt{1-\bar{\alpha}_k}}. \tag{5}$$

Guidance is useful only when the rollout has drifted from the demonstrations but remains recoverable: too close to the phase targets it perturbs a correct action, too far away the dynamics model is extrapolating and its gradient is unreliable. ReGuide therefore gates guidance per phase. Let $\ell = \mathcal{L}\big(d_\phi(z_t,\ \hat{A}_t^0), \mathcal{P}_j\big)$ be the phase-conditioned distance at the current denoising step. We take $\ell_{j,\text{low}}$ and $\ell_{j,\text{high}}$ as fixed percentiles of $\mathcal{F}_j$, so the gate is calibrated to the local geometry of that stage, and guide only when $\ell_{j,\text{low}} < \ell < \ell_{j,\text{high}}$. The thresholds are computed once from the demonstrations and held fixed during evaluation.

During execution, ReGuide maintains the current phase index j. We advance from phase j to $j+1$ when the predicted latent is closer to $\mathcal{P}_{j+1}$ than to $\mathcal{P}_j$ by a margin Δ for K consecutive rollout steps, which prevents transient prediction noise from causing premature phase switches. Each rollout is stored as $\{(O_t, A_t^g)\}_{t=0}^{T-1}$, where A_t^g is the guided action chunk at step t.

4.3 Iterative Self-Improvement Learning

This step folds the guided rollouts back into the policy, so that the loop can repeat. The rollout procedure of Algorithm 3 turns guidance into DAgger-style (Ross et al., 2011) training data: at iteration i we keep only the successful trajectories and merge them into a guided buffer $\mathcal{D}_i^g$, and updating the policy on it produces π_{i+1}, which collects the next batch.

The guided batch is small compared to the original training data and is drawn from a different distribution—drifted states and the corrections that recover from them. So fine-tuning on it can erode competence the policy already has, while retraining from scratch discards a checkpoint that is usually already good and costs a full training run. The right choice also depends on how strong that checkpoint is: a weak base policy is a poor starting point to refine. We therefore consider three ways to absorb the guided data, trading computation against performance. Across all of them, two components are held fixed for efficiency: the dynamics model, trained once on the base policy's rollouts, and the phase targets and per-phase distance distributions, constructed from $\mathcal{D}_i$.

ReGuide-FT. The first variant fine-tunes the current policy checkpoint on a rehearsal buffer containing both prior training data and newly collected guided rollouts. At iteration i, minibatches are

sampled from the existing dataset $\mathcal{D}_i$ and the guided buffer $\mathcal{D}_i^g$ with a fixed ratio ρ. This update is computationally efficient and preserves the competence of the current policy while exposing it to recoverable states reached during guided execution.

ReGuide-FS. The second variant trains a fresh diffusion policy from scratch on the augmented dataset $\mathcal{D}_i \cup \mathcal{D}_i^g$. This removes dependence on the previous checkpoint and can yield a different solution when the base policy is a poor initialization, at the cost of a full retraining run.

ReGuide-FS→FT. Since ReGuide-FT operates on any starting checkpoint, we can stack the variants: apply ReGuide-FT on top of a ReGuide-FS policy rather than the base. ReGuide-FS provides a stronger initialization than the base, so the FT round refines an already-competent policy; the composition costs both runs, and we find its gain real but sensitive to the number of added rollouts (Section 5).

5 EXPERIMENTS

Experimental setup. We evaluate ReGuide on four Robomimic manipulation tasks (Mandlekar et al., 2022): Can, Square, Transport, and Tool Hang, and on three tasks from LIBERO (Liu et al., 2023). On Robomimic we train both the base policy and the dynamics model ourselves, the latter on rollouts from the corresponding base policy; on LIBERO both are checkpoints released by LPB (Sun & Song, 2025), trained jointly across 10 tasks, the phase targets are clustered from the pooled latents of all ten, and we apply ReGuide-FT to the three tasks that checkpoint handles worst. The rest of this section concerns Robomimic. For each Robomimic task we train a diffusion policy on a small subset of the available demonstration data—15 demonstrations for Can, 30 for Square, 10 for Transport and 80 for Tool Hang—and construct phase targets and per-phase distance distributions from the current training data. Every success rate is a mean over 2,500 episodes from initial states that are fixed per task and shared across every arm, so all within-task comparisons are paired; Appendix E gives the evaluation protocol and the remaining implementation details.

Baselines. *LPB* (Sun & Song, 2025) and *DynaGuide* (Du & Song, 2025) are the test-time guidance methods of Section 2, run here against the same frozen base checkpoint. *Filtered BC* is our matched baseline. Training on success-filtered, autonomously collected rollouts is an established route to self-improvement (Bousmalis et al., 2023; Mirchandani et al., 2024): the identical absorption procedure applied to success-filtered *unguided* rollouts of the same policy—*Filtered BC-FT* for the ReGuide-FT update and *Filtered BC-FS* for ReGuide-FS—holding the checkpoint, the task, the number of *retained successful* trajectories, and the buffer-share ratio ρ fixed, so that the only difference between the two arms is whether guidance was active during collection. It isolates guidance from data volume, and because the match is on retained successes rather than attempted episodes, the filtered-BC arm consumes *more* environment episodes than the guided arm to reach any given count. *Hyperparameters.* PCG's per-task values (Table 19) were selected by sweeping each and taking the setting with the highest mean success on the evaluation episodes; Appendix E states the protocol. LPB uses its published Robomimic settings on Square, Transport and Tool Hang, and on Can, which it does not report, was tuned by the same sweep-on-evaluation procedure, as was DynaGuide on every task. The guidance rows of Table 1 are therefore all selected on the evaluation set, PCG no less than the baselines.

Main results. All three ReGuide variants improve substantially over the base policy (Table 1, Figure 3). **ReGuide-FT** lifts success by 1.2–1.5× on Can, Square and Transport and 7.2× on Tool Hang after two iterations; **ReGuide-FS** reaches 1.3–1.5× and 5.0× on the same tasks. Neither uniformly dominates, and ReGuide-FT's second iteration is decisively ahead on Tool Hang (6.4 SE of the difference). The composition **ReGuide-FS→FT** leads at only one count of each three-count sweep: at the others it falls below its own ReGuide-FS starting checkpoint on Square and Transport (Table 11). Since the counts are nested prefixes of a single collection pool, this is sensitivity to data volume rather than to the draw, and we read the composition as a real but count-sensitive gain whose Table 1 value is the maximum of a noisy sweep. It does not lead on Tool Hang, where the base policy is too weak for ReGuide-FS to reach a stronger checkpoint. That 7.2× comes from absorbing self-collected rollouts and iterating from a 3% base policy; whether guidance improves data quality there is unresolved, since the guided and unguided arms are indistinguishable under FS and 1.3 SE apart under FT (Appendix D.1). The composition has no matched baseline, so it shows that the loop keeps improving, not that guidance is responsible for its additional lift. For iteration 2 we ran the

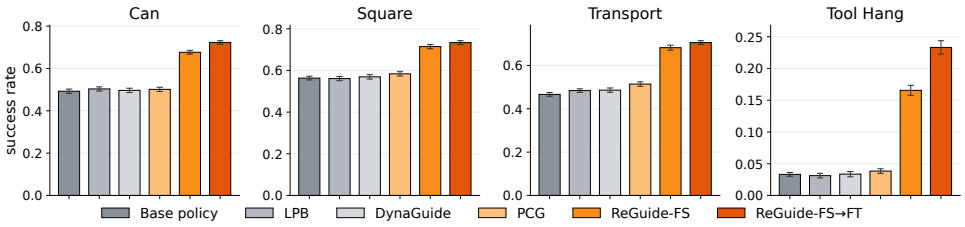


Figure 3: **Main results per task.** Success rate of the base policy, LPB, DynaGuide, PCG, ReGuide-FS and the composition ReGuide-FS→FT on each Robomimic task, with standard errors; every value is a column of Table 1. Note the different vertical scale on Tool Hang, where the base policy succeeds on 3.3% of episodes.

Table 1: **Success rates across Robomimic tasks** (mean ± standard error). The three test-time guidance methods steer the same frozen base checkpoint from shared initial states, so differences among them cannot be attributed to training variance. *Filtered BC-FT* (defined under **Baselines**) is the matched baseline for ReGuide-FT iteration 1; both are reported at their best count, which for each of them is N ($0.25N$ on Tool Hang). ReGuide-FS has its own matched baseline in Table 10, iteration 2 has one on Square only (Table 13), and FS→FT has none. Per task, the highest success rate is in **bold** and the second-best is underlined.

Method	Can	Square	Transport	Tool Hang
Base policy	0.4924 ± 0.0101	0.5632 ± 0.0091	0.4664 ± 0.0086	0.0332 ± 0.0030
Test-time guidance (frozen policy)				
LPB (Sun & Song, 2025)	0.5032 ± 0.0099	0.5608 ± 0.0099	0.4840 ± 0.0082	0.0312 ± 0.0036
DynaGuide (Du & Song, 2025)	0.4964 ± 0.0102	0.5692 ± 0.0106	0.4860 ± 0.0101	0.0336 ± 0.0041
PCG (ours)	0.5012 ± 0.0100	0.5836 ± 0.0104	0.5140 ± 0.0102	0.0384 ± 0.0036
Absorbing rollouts into the policy				
Filtered BC-FT *(baseline)*	0.5664 ± 0.0093	0.6188 ± 0.0114	0.5692 ± 0.0088	0.1456 ± 0.0076
ReGuide-FT, iteration 1	0.5932 ± 0.0098	0.6592 ± 0.0095	0.6456 ± 0.0110	0.1592 ± 0.0077
ReGuide-FT, iteration 2	0.6236 ± 0.0090	0.7120 ± 0.0085	<u>0.6948</u> ± 0.0094	**0.2404** ± 0.0088
ReGuide-FS	<u>0.6764</u> ± 0.0089	<u>0.7144</u> ± 0.0101	0.6820 ± 0.0114	0.1656 ± 0.0078
ReGuide-FS → FT	**0.7228** ± 0.0083	**0.7336** ± 0.0091	**0.7056** ± 0.0086	<u>0.2332</u> ± 0.0106

matched control on Square (Table 13), matched at all six cumulative counts. Filtered BC improves under a second iteration too, from 0.6188 to 0.6920; both arms peak at 150, where guided leads by +0.020 (1.6 SE). The second iteration improves both arms, and guidance leads over most of the sweep but not uniformly. Among the rows that have one, guided rollouts win on every task under FT and on Can, Square and Transport under FS. Each cell is a single training run, so the load-bearing evidence is the seeded comparison in Appendix D.7: on Can, ReGuide-FS leads Filtered BC-FS by 0.089 across three training seeds, every ReGuide-FS seed above every Filtered BC-FS seed, and ReGuide-FT leads Filtered BC-FT on every seed by a smaller 0.023. Seed averaging also lowers the lifts over the base, sharply for ReGuide-FT on Can (+0.101 → +0.039) but not for ReGuide-FS.

Appendix D.1 gives the comparison at every count and under ReGuide-FS as well.

Comparison against test-time guidance baselines. We compare PCG to LPB (Sun & Song, 2025) and DynaGuide (Du & Song, 2025) in the test-time-only setting (no fine-tuning); the three guidance rows of Table 1 give the comparison. PCG has the highest point estimate on Square, Transport and Tool Hang, but the only separation beyond 2 SE anywhere in the block is PCG on Transport (+0.048 over base, ≈3.6 SE); on the other three tasks no guidance method separates from the base policy or from one another. We therefore claim only that PCG is at least as good as both baselines everywhere and better where any method moves. These test-time gains are much smaller than the full ReGuide gains, which is expected: the main role of PCG is not to improve a single rollout but to generate cleaner data for the subsequent ReGuide-FT and ReGuide-FS updates, where the larger gains are.

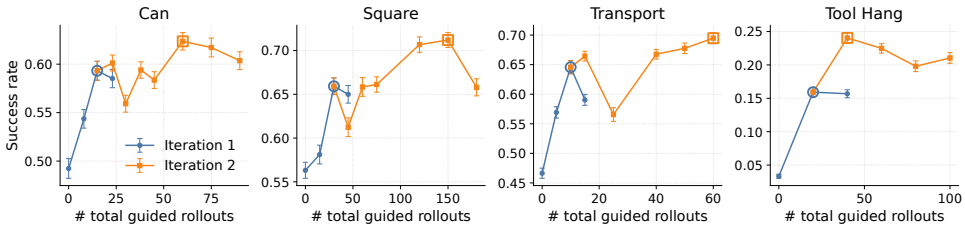


Figure 4: **Iterative self-improvement.** Success rate against the cumulative number of guided rollouts collected to update the policy, for ReGuide-FT iterations 1 and 2.

Table 2: **LIBERO: each arm at its own best rollout count.** The three tasks the released multi-task checkpoint handles worst. Each entry is that arm's highest cell over the four counts of Table 17, with the count in parentheses; A maximum over four evaluations of the same episodes is biased upward for both arms equally. Best of the two arms in **bold**.

Task	Base policy	Filtered BC-FT	ReGuide-FT
Living Room 5	0.4876 ± 0.0115	$0.5344 \pm 0.0099\,(0.25N)$	$\mathbf{0.5484} \pm 0.0104\,(0.50N)$
Living Room 6	0.1776 ± 0.0076	$0.2984 \pm 0.0095\,(0.75N)$	$\mathbf{0.3812} \pm 0.0099\,(0.75N)$
Kitchen 8	0.6804 ± 0.0138	$0.6824 \pm 0.0084\,(1.00N)$	$\mathbf{0.7380} \pm 0.0093\,(0.75N)$

Iterative self-improvement. Starting from the iteration-1 policy at added-rollouts $= N$ ($0.25N$ on Tool Hang), we collect a new batch of guided rollouts and run a second ReGuide-FT iteration (Figure 4; per-count values in Table 12). It outperforms the first on all four tasks, most sharply on Tool Hang, where the lift over the base grows from $4.8\times$ to $7.2\times$. The gains are not unbounded: the second-iteration peak needs more total rollouts, suggesting diminishing returns; we do not isolate the cause.

Improving a pretrained multi-task checkpoint (LIBERO). We fine-tune a separate copy of the unmodified LPB (Sun & Song, 2025) checkpoint for each of its three weakest tasks, so the output is three task-specialized policies rather than one improved multi-task policy; performance on the other seven tasks is not preserved or claimed. Table 2 reports each arm at its own best rollout count. ReGuide-FT lifts the base policy on all three tasks and leads Filtered BC-FT on each, decisively on Living Room 6 ($+0.083$, 6.0 SE) and Kitchen 8 ($+0.056$, 4.4 SE) and narrowly on Living Room 5 ($+0.014$, 1.0 SE). A maximum over four counts is biased upward for both arms, and per count the evidence is directional rather than decisive (Appendix D.9, Table 17). Test-time PCG transfers unevenly: it lowers success on seven of the ten tasks and the suite mean falls from 0.735 to 0.712, which we trace to the pooled construction: both the phase targets and the gate thresholds are clustered over all ten tasks, so on a task the policy already handles well the gate fires when the rollout is not drifting, and the coarse targets then pull it off a trajectory that was succeeding. Those seven are all tasks we do not collect from. On the three weakest tasks, the ones ReGuide-FT is applied to, PCG is not harmful—it gains 3.4 standard errors on Living Room 6 and ties on the other two—so guidance helps where drift is the failure mode and hurts where it is not. Appendix D.9 gives the setting, results, and caveats in full.

Positioning against RL fine-tuning of diffusion policies. Section 2 situates ReGuide among RL methods that fine-tune diffusion policies; here we say why none appears as a baseline. Table 4 in Appendix A.1 reports what each loop *consumes* rather than which performs better, and makes visible a third asymmetry beyond the supervision and demonstration budgets of Section 2: every RL row observes privileged object state, where ReGuide sees only pixels and proprioception. DSRL and ReGuide are the one pair whose cost and gain can both be read off: on Can, DSRL lifts a base policy of $\approx$0.46 to $\approx$0.97 using $\approx 1.3\times10^6$ environment steps; ReGuide trains its dynamics model on 300 unlabeled base-policy episodes and collects 50 guided ones, $\approx 1.4\times10^5$ steps in total, and lifts 0.4924 to 0.6764—roughly an order of magnitude less interaction. Appendix A.1 gives the full discussion, including ReGuide's offline training cost.

Ablation studies. Table 3 summarizes the ablations that isolate ReGuide's main algorithmic choices; the full sweeps are in Appendix D.

Table 3: **Summary of ablations.** The first three rows steer a single frozen checkpoint on Transport, where the base policy scores 0.4664, so no training variance enters them. *Observation noise* re-evaluates the matched-volume arms of Appendix D.1 under corruption rather than ablating a design choice. SE denotes standard errors of the stated difference; the last column gives the appendix table with the full sweep. Training variance exceeds evaluation error at fixed weights (Table 15), so differences below ≈ 0.05 anywhere in this table should be read as indistinguishable.

Component or check	Finding	App. table
Phase conditioning	$N_p{=}1$, prior work's global target set, sits at base level; $N_p{=}4$ gains $+0.048$ (3.6 SE); best at $M \approx 50$ targets per phase.	5, 6a
Two-threshold gate	Both thresholds beat either alone (1.4 and 1.9 SE) and gain $+0.048$ (3.6 SE) over no guidance.	6b
Clean-action guidance	Guiding $\hat{A}_t^0$ beats the noisy iterate A_t^k by $\approx$2 SE.	7
Guidance vs. added data	Guided wins every matched count on Can, Square and Transport, both modes; Tool Hang ties under FS.	10, 8
ReGuide-FT vs. FS	Neither dominates on Can (FT ahead at 8 rollouts, FS at 15); FT is cheaper from a strong checkpoint.	9
Observation noise	Guided leads in eight of nine task–noise cells; it degrades faster than the control only on Square (2.3 SE).	16

6 Conclusion and Limitations

ReGuide turns test-time guidance from a one-time inference correction into reusable on-policy recovery data: a successful guided rollout records where a behavior-cloned policy drifts and which actions recover task progress. On Robomimic this lifts base-policy success by 1.2–7.2×, and at matched data volume guided rollouts beat unguided ones at every count on Can, Square and Transport; on LIBERO the same absorption step improves a pretrained multi-task checkpoint on its three weakest tasks.

Limitations. We believe the three major limitations are the following. (i) *Training variance is measured on two tasks only.* Every number except the three-seed audit of Appendix D.7 comes from a single trained checkpoint, and that audit covers Can and Transport. Extending it to Square and Tool Hang, and to the iteration-2 and composition arms, is a matter of compute rather than method. (ii) *The dynamics model and phase targets are built once and reused.* We do not isolate why the per-iteration gain shrinks between iterations. Whether refreshing them between iterations extends the loop, and how far, is the measurement we would add. (iii) *The LIBERO gate is calibrated against a pooled distance distribution.* Targets clustered over ten tasks give only a coarse direction of task progress, and a pooled gate fires on strong tasks that did not need guidance (Appendix D.9). Per-task clustering is too data-starved to be stable, but calibrating only the per-phase thresholds per task while keeping pooled targets is inexpensive and untried. Two further constraints bound the setting rather than the evidence: PCG requires a dynamics model sharing the policy's visual encoder, and all our tasks are simulated, where rollouts of a partially failing policy are free—on hardware they cost scene resets, wear and supervision (50–550 attempted episodes per task, Appendix E).

Use of Generative AI

In this work, we used generative AI tools to **implement methods** and to **provide feedback on research methodology and experiments**: parts of the experiment, training and evaluation code were written with AI assistance, and an AI assistant advised on which additional experiments would strengthen the evaluation and on how tightly the reported claims should be scoped to the available evidence. The experiments themselves were designed, run and analysed by the authors. We have not used generative AI tools to generate synthetic datasets, to develop theoretical models or conceptual frameworks, to propose or refine hypotheses, to clean or reformat datasets, or to interpret results.

Formulating mathematical claims, providing critical ingredients for proofs and assisting in the writing of proofs are not applicable to this work, which contains no formal claims or proofs; translation and qualitative or thematic data analysis are likewise not applicable. Additionally, we used generative AI tools to polish the writing of this manuscript, including copy-editing and reorganising the paper.

We have reviewed all AI-assisted work. AI-assisted code was read, tested and verified by the authors before use, and every experimental number reported in this paper comes from runs that the authors executed and checked rather than from any AI tool. All AI-assisted text was reviewed and edited by the authors, and the claims, their scoping and the interpretation of results are our own. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Lars Ankile, Anthony Simeonov, Idan Shenfeld, and Pulkit Agrawal. Juicer: Data-efficient imitation learning for robotic assembly. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5096–5103, 2024. doi: 10.1109/IROS58592.2024.10802498.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X. Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, Antoine Laurens, Claudio Fantacci, Valentin Dalibard, Martina Zambelli, Murilo Martins, Rugile Pevceviciute, Michiel Blokzijl, Misha Denil, Nathan Batchelor, Thomas Lampe, Emilio Parisotto, Konrad Żołna, Scott Reed, Sergio Gómez Colmenarejo, Jon Scholz, Abbas Abdolmaleki, Oliver Groth, Jean-Baptiste Regli, Oleg Sushkov, Tom Rothörl, José Enrique Chen, Yusuf Aytar, Dave Barker, Joy Ortiz, Martin Riedmiller, Jost Tobias Springenberg, Raia Hadsell, Francesco Nori, and Nicolas Heess. RoboCat: A self-improving generalist agent for robotic manipulation. *Transactions on Machine Learning Research (TMLR)*, 2023.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023. doi: 10.15607/RSS.2023.XIX.026.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 8780–8794, 2021.

Maximilian Du and Shuran Song. Dynaguide: Steering diffusion polices with active dynamic guidance, 2025. URL https://arxiv.org/abs/2506.13922.

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Diffusion guidance is a controllable policy improvement operator, 2025. URL https://arxiv.org/abs/2505.23458.

Dechen Gao, Hang Wang, Hanchu Zhou, Nejib Ammar, Shatadal Mishra, Ahmadreza Moradipari, Iman Soltani, and Junshan Zhang. IN-RIL: Interleaved reinforcement and imitation learning for policy fine-tuning, 2025. URL https://arxiv.org/abs/2505.10442.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1025–1037, 2020.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies, 2023. URL https://arxiv.org/abs/2304.10573.

Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J. Zico Kolter, Ruslan Salakhutdinov, and Stefano Ermon. Manifold preserving guided diffusion. In *International Conference on Learning Representations (ICLR)*, 2024.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS Workshop on Deep Generative Models and Downstream Applications*, 2021.

Zheyuan Hu, Robyn Wu, Naveen Enock, Jasmine Li, Riya Kadakia, Zackory Erickson, and Aviral Kumar. Rac: Robot learning for long-horizon tasks by scaling recovery and correction, 2025. URL https://arxiv.org/abs/2509.07953.

Mingxi Jia, Dian Wang, Guanang Su, David Klee, Xupeng Zhu, Robin Walters, and Robert Platt. SEIL: Simulation-augmented equivariant imitation learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1845–1851, 2023.

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 (ICLR)*, 2024.

Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. AdaptDiffuser: Diffusion models as adaptive self-evolving planners. In *International Conference on Machine Learning (ICML)*, 2023.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL https://arxiv.org/abs/2306.03310.

Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22825–22855, 2023.

Lei Lv, Yunfei Li, Yu Luo, Fuchun Sun, Tao Kong, Jiafeng Xu, and Xiao Ma. Flow-based policy for online reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Ajay Mandlekar, Fabio Ramos, Byron Boots, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Dieter Fox. IRIS: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4414–4420, 2020.

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. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, volume 164 of *Proceedings of Machine Learning Research*, pp. 1678–1690, 2022.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. MimicGen: A data generation system for scalable robot learning using human demonstrations. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1820–1864, 2023.

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. doi: 10.1109/LRA.2025.3526439.

Suvir Mirchandani, Suneel Belkhale, Joey Hejna, Evelyn Choi, Md Sazzad Islam, and Dorsa Sadigh. So you think you can scale up autonomous robot data collection? In *Proceedings of the Conference on Robot Learning (CoRL)*, 2024.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *International Conference on Machine Learning (ICML)*, 2025.

Sarvesh Patil, Mitsuhiko Nakamoto, Manan Agarwal, Shashwat Saxena, Jesse Zhang, Giri Anantharaman, Cleah Winston, Chaoyi Pan, Douglas Chen, Nai-Chieh Huang, Zeynep Temel, Oliver Kroemer, Sergey Levine, Abhishek Gupta, Hongkai Dai, Paarth Shah, and Max Simchowitz. OGPO: Sample efficient full-finetuning of generative control policies, 2026. URL https://arxiv.org/abs/2605.03065.

Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. Inference-time enhancement of generative robot policies via predictive world modeling, 2026. URL https://arxiv.org/abs/2502.00622.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations (ICLR)*, 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 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, 2011.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 205 of *Proceedings of Machine Learning Research*, pp. 785–799, 2022.

Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy agnostic RL: Offline RL and online RL fine-tuning of any class and backbone, 2024. URL https://arxiv.org/abs/2412.06685.

Zhanyi Sun and Shuran Song. Latent policy barrier: Learning robust visuomotor policies by staying in-distribution, 2025. URL https://arxiv.org/abs/2508.05941.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning, 2025. URL https://arxiv.org/abs/2506.15799.

Zixing Wang, Devesh K. Jha, Ahmed H. Qureshi, and Diego Romeres. Ppguide: Steering diffusion policies with performance predictive guidance, 2026. URL https://arxiv.org/abs/2603.10980.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Xiaomeng Xu, Yifan Hou, Chendong Xin, Zeyi Liu, and Shuran Song. Compliant residual dagger: Improving real-world contact-rich manipulation with human corrections, 2025. URL https://arxiv.org/abs/2506.16685.

Brent Yi, Hongsuk Choi, Himanshu Gaurav Singh, Xiaoyu Huang, Takara E. Truong, Carmelo Sferrazza, Yi Ma, Rocky Duan, Pieter Abbeel, Guanya Shi, Karen Liu, and Angjoo Kanazawa. Flow policy gradients for robot control, 2026. URL https://arxiv.org/abs/2602.02481.

Yixian Zhang, Shu'ang Yu, Tonghe Zhang, Mo Guang, Haojia Hui, Kaiwen Long, Yu Wang, Chao Yu, and Wenbo Ding. SAC flow: Sample-efficient reinforcement learning of flow-based policies via velocity-reparameterized sequential modeling. In *International Conference on Learning Representations (ICLR)*, 2026.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 79115–79135, 2025.

A Extended Related Work

This section expands on the related work discussion in Section 2, providing additional detail on individual methods and covering directions that did not fit in the main paper.

Test-time guidance for diffusion policies. DynaGuide (Du & Song, 2025) uses an external dynamics model to compute gradients that bias the denoising process toward a set of user-specified goal states, with guidance applied to the noisy denoising iterate at each reverse-diffusion step. That goal set is global: it is specified once and applied unchanged for the whole episode, so DynaGuide has no notion of task phase and a single set of targets must serve every stage of a long-horizon task at once. This is precisely the design point Section 4.1 argues against—a global target set can pull a

rollout toward the wrong stage, or mark a valid mode of the current stage as out-of-distribution—and it, rather than the number of targets, is what separates DynaGuide from PCG. Latent Policy Barrier (LPB) (Sun & Song, 2025) adapts the same gradient mechanism to keep rollouts close to the expert distribution, treating expert latent embeddings as an implicit in-distribution barrier rather than as a goal to reach. PPGuide (Wang et al., 2026) replaces the dynamics-gradient signal with a learned performance predictor, steering toward action chunks that score well under the predictor. Generative Predictive Control (GPC) (Qi et al., 2026) takes a different approach: rather than gradient-based guidance, it samples multiple action candidates and uses an action-conditioned world model to rank them via lightweight model-based look-ahead. PCG differs from these methods along three axes: (i) it differentiates through the dynamics model with respect to the estimated clean action rather than the noisy iterate, matching the dynamics model's training distribution; (ii) it replaces a single global target set with phase-conditioned target sets that preserve expert multimodality within each phase; and (iii) it gates guidance to the drifted-but-recoverable regime via a two-threshold rule calibrated to per-phase distance distributions.

Data augmentation for imitation learning. Beyond CCIL discussed in the main paper, several methods expand the training distribution offline using task structure or assumptions about which perturbations preserve correctness. JUICER (Ankile et al., 2024) combines expressive architectures with dataset-expansion and simulation-based augmentation for long-horizon assembly. SEIL (Jia et al., 2023) supplements expert trajectories with simulated transitions and exploits manipulation symmetries to generate equivalent demonstrations. Stable-BC (Mehta et al., 2025) regularizes the policy so that its closed-loop behavior is locally stable around expert states. These methods share with ReGuide the goal of expanding training coverage without additional expert supervision, but rely on assumption-driven synthesis: known symmetries, local dynamics models, or stability conditions. ReGuide instead generates corrective data *on-policy* by rolling out the current policy under guidance and keeping successful trajectories, requiring only a trajectory-level success signal rather than assumptions about which perturbations preserve task correctness.

Phase / sub-task decomposition in manipulation. Decomposing long-horizon manipulation into temporally ordered phases or sub-skills is a recurring theme in robot learning, instantiated at several different levels of the pipeline. IRIS (Mandlekar et al., 2020) factors the control problem into a high-level goal planner and a low-level goal-conditioned controller, learning sub-policies from offline demonstrations. Relay Policy Learning (Gupta et al., 2020) learns goal-conditioned hierarchical policies from unsegmented demonstrations via a data-relabeling trick and then fine-tunes them with reinforcement learning. PerAct (Shridhar et al., 2022) sidesteps dense trajectory prediction altogether by predicting discrete keyframe end-effector poses that serve as phase anchors, with a motion planner handling the low-level control between them. MimicGen (Mandlekar et al., 2023) uses an object-centric subtask decomposition for offline data generation, transforming and stitching demonstration segments to synthesize new trajectories. JUICER (Ankile et al., 2024) similarly leverages subtask structure for both architecture design and augmentation in long-horizon assembly. ReGuide uses phases differently from all of these: the diffusion policy itself stays unfactored, the action representation stays continuous, and the demonstration data is not segmented for retargeting. Phases here serve only as an inference-time tool for localizing the "what counts as in-distribution" check that gates our guidance, adapting the threshold to local task geometry without imposing a structural decomposition on the policy, the data, or the action space.

Iterative self-improvement and rehearsal-based learning. Our rollout–collect–train loop is closer in spirit to self-training in supervised learning (Xie et al., 2020), where a model labels new data, filters by confidence, and retrains, than to classical continual learning, even though we borrow the rehearsal buffer construction from the latter. Rehearsal-based continual learning methods retain or replay a subset of past data during new updates to mitigate forgetting (Rebuffi et al., 2017; Buzzega et al., 2020); our two-buffer formulation in Section 4.3 reuses this trick, with the buffer-share ratio ρ functioning as a rehearsal fraction. The setting differs, though: continual learning typically targets sequential acquisition of *distinct* tasks, while we iteratively refine a policy on the *same* task using progressively higher-quality self-collected data. The closest neighbors are therefore self-training methods that filter generated data by an external success signal, with ReGuide's distinguishing element being that the data-generation step is itself a guided diffusion rollout rather than a forward pass of the current model.

A.1 Positioning against RL Fine-Tuning of Diffusion Policies

This section expands the discussion in Section 5 accompanying Table 4.

Table 4: **What the loop consumes: RL fine-tuning of diffusion policies vs. ReGuide on Robomimic.** Success rates are omitted because they are not commensurate across rows; the text discusses the one pair whose cost and gain can both be read off. *Demos* is the number of demonstrations used to pretrain the policy being improved. *Obs.* distinguishes policies given privileged object state from those given only camera images plus robot proprioception; both include proprioception. *Value fn.* marks whether the method fits a critic or value function from reward-labeled online interaction. *Env. interaction* is the online interaction cost, excluding gradient steps. ReGuide appears on two rows: its guided collection (50, 100 and 50 attempted episodes on Can, Square and Transport), and that plus the 300 unlabeled base-policy episodes that train the dynamics model once and are reused by every arm (Table 20). At Robomimic horizons (400, 400, 700) the latter is at most 1.4, 1.6 and 2.5×10^5 steps, an upper bound since successful episodes terminate early. The RL methods train no auxiliary model from environment data, so their single figure is already the total. "n.r." marks a quantity the source paper does not report; "—" marks one that does not apply. No RL method in this table evaluates Tool Hang, so that task is omitted.

			Supervision		
Method	Demos	Obs.	Reward	Value fn.	Env. interaction
DPPO (Ren et al., 2025)	300	proprio. + obj.	terminal	✓	$\sim10^6$ steps
IN-RIL (Gao et al., 2025)	300	proprio. + obj.	terminal	✓	n.r.
DSRL (Wagenmaker et al., 2025)	300	proprio. + obj.	terminal	✓	1.3×10^6–1.5×10^7 steps
SAC Flow (Zhang et al., 2026)	300	proprio. + obj.	terminal	✓	$\sim10^6$ steps
Ours					
Base policy	15/30/10	image + proprio.	—	×	—
ReGuide-FS ($1N$, one iteration)	15/30/10	image + proprio.	terminal, 1 bit	×	$\approx$2–4$\times10^4$ steps
incl. dynamics-model data					$\lesssim$1.4–2.5$\times10^5$ steps

Positioning against RL fine-tuning of diffusion policies. ReGuide's loop—roll out, score the outcome, update the policy—is structurally the same as reinforcement learning for diffusion policies, and a reader is entitled to ask why no RL method appears as a baseline. Table 4 answers by placing the two next to each other on Robomimic. It is emphatically *not* a like-for-like comparison: every RL row is transcribed from its own paper under that paper's demonstration budget, observation space, reward definition and evaluation protocol, and none of these methods was rerun by us. What the table reports is what each loop consumes. Three asymmetries carry the argument. First, every RL entry pretrains on 300 demonstrations, against the 10–30 used here on the same three tasks. Second, every RL row observes low-dimensional proprioception *together with privileged object state*, where ReGuide receives camera images and robot proprioception but no object pose, and must infer object configuration from pixels. Third, and most important for the loop structure, all three fit a value function online from reward-labeled interaction; ReGuide fits none, and consumes one bit per trajectory. We also note that none of these methods reports Tool Hang, the hardest of our four tasks and the one on which ReGuide's lift is largest (7.2×). Only DSRL and ReGuide report both a starting checkpoint and a final number, so they are the one pair whose cost and gain can both be read off rather than guessed. On Can, DSRL lifts a base policy of $\approx$0.46 to $\approx$0.97 using $\approx1.3\times10^6$ environment steps; ReGuide collects 50 attempted episodes for guided data, at most $\approx2\times10^4$ environment steps at Can's horizon, and $\approx1.4\times10^5$ once the 300 unlabeled episodes that train the dynamics model are included (Table 4); it lifts 0.4924 to 0.6764. Even counting the dynamics-model data, that is roughly an order of magnitude less interaction. Interaction is not the only cost, and the table does not show the other one: ReGuide-FS retrains a policy from scratch on the augmented dataset, $\approx$4–10$\times10^4$ gradient steps depending on the task, and ReGuide-FT pays a smaller fine-tuning cost in its place. That compute is incurred offline, between rollout batches, and so is cheap in the currency that matters on hardware—scene resets, wear and supervision—but it is not free, and it is the reason we report ReGuide-FT as the practical variant where the two are statistically comparable. We deliberately do not reduce this to a cost-per-point ratio. Returns diminish as a policy approaches ceiling, so a method that continues to $\approx$0.97 pays for an expensive tail that a method stopping at 0.6764 never enters, and any such ratio would flatter whichever method stopped earlier. We report no success-rate

column for these methods, because such a column would invite a ranking that the numbers cannot support. The RL rows reach higher absolute success than ReGuide, and they do so from 10–30$\times$ the demonstrations with a denser supervision signal; conversely, ReGuide's numbers are reached without a critic, without per-step reward, and from a base policy that succeeds on under half of its episodes. The table is a statement about cost, not about which method is better.

B Background: Guidance for Diffusion Models

This section gives the image-generation form of guidance that Section 3 specializes to action chunks, together with the guidance methods for control that it builds on.

Diffusion models admit a natural mechanism for steering generation at inference time: the reverse process can be biased toward a desired property with the gradient of a differentiable reward function. Concretely, when generating a sample x via reverse diffusion from $x_K \sim \mathcal{N}(0, I)$ down to x_0, classifier guidance (Dhariwal & Nichol, 2021) and its classifier-free variants (Ho & Salimans, 2021) modify the predicted noise at step k as

$$\hat{\epsilon}(x_k) \;=\; \epsilon_\theta(x_k) \;-\; \eta\sqrt{1-\bar{\alpha}_k}\,\nabla_{x_k} r(x_k), \tag{6}$$

where r is a reward (e.g., a classifier log-probability for a target class, or text-to-image fidelity), and η controls the guidance strength. The gradient $\nabla_{x_k} r(x_k)$ pushes each denoising step toward samples that score higher under r, so the final x_0 tends to satisfy the reward criterion while still lying on the data manifold learned by ϵ_θ. This principle has been used to steer image generation toward target classes, text descriptions, and aesthetic properties without retraining the underlying diffusion model.

Recent work transfers this idea from image generation to control policies represented as diffusion models. In offline reinforcement learning, several methods guide a diffusion policy using the gradient of a learned value function (Frans et al., 2025; Lu et al., 2023; Hansen-Estruch et al., 2023), steering action generation toward high-value behavior. In imitation learning, DynaGuide (Du & Song, 2025) and LPB (Sun & Song, 2025) instead use a learned dynamics model to keep rollouts close to the expert distribution. Specializing this to a diffusion policy replaces the generated sample x with the action chunk A_t and the reward r with a cost δ to be minimized, which gives the form used in Section 3,

$$\hat{\epsilon}(A_t^k) \;=\; \epsilon_\theta(A_t^k) \;-\; \eta\sqrt{1-\bar{\alpha}_k}\,\nabla_{A_t^k} \delta(A_t^k), \tag{7}$$

with $x_k \leftrightarrow A_t^k$ throughout. PCG modifies this in two places: δ becomes the phase-conditioned distance of Section 4.1, and the gradient is taken with respect to the estimated clean action rather than A_t^k (Section 4.2).

C ALGORITHM

Algorithm 2 Phase-Aware Target Construction

Require: Training data $\mathcal{D}_{\text{training}} = \{(o_t, a_t)\}$, visual encoder E_v, proprio encoder E_p, number of clusters k, number of phases N_p, targets per phase M, PCA dimension d

Ensure: Per-phase target sets $\{\mathcal{P}_j\}_{j=1}^{N_p}$ and distance distributions $\{\mathcal{F}_j\}_{j=1}^{N_p}$

Encode states with temporal augmentation:
for each $(o_t, a_t) \in \mathcal{D}_{\text{training}}$ **do**
  $z_t^{\text{vis}} \leftarrow E_v(o_t), \;\; z_t^{\text{pro}} \leftarrow E_p(o_t)$ ▷ visual and proprio features
  $\Delta z_t^{\text{vis}} \leftarrow z_t^{\text{vis}} - z_{t-1}^{\text{vis}}, \;\; \Delta z_t^{\text{pro}} \leftarrow z_t^{\text{pro}} - z_{t-1}^{\text{pro}}$ ▷ zero at $t = 0$
  $f_t \leftarrow [\, z_t^{\text{vis}}, \; z_t^{\text{pro}}, \; \Delta z_t^{\text{vis}}, \; \Delta z_t^{\text{pro}} \,]$ ▷ clustering feature
end for
$\{\tilde{f}_t\} \leftarrow \text{PCA}(\{f_t\}, d)$
$\{c_t\} \leftarrow \text{KMeans}(\{\tilde{f}_t\}, k)$
Order clusters temporally:
for each cluster $c \in \{1, \ldots, k\}$ **do**
  $\bar{t}_c \leftarrow \text{mean}\{t : c_t = c\}$
end for
Sort clusters by $\bar{t}_c$ in ascending order
Group into phases: Partition the k ordered clusters into N_p contiguous groups, $\{\Phi_1, \ldots, \Phi_{N_p}\}$
Select phase targets and build distance distributions:
for $j = 1$ to N_p **do**
  $\mathcal{P}_j \leftarrow M$ representative points of Φ_j, expressed in the dynamics-model latent space
  $\mathcal{F}_j \leftarrow \{\mathcal{L}(z_t, \mathcal{P}_j) : t \in \Phi_j\}$ ▷ $z_t = h(O_t)$, as in Sec. 3
end for
return $\{\mathcal{P}_j\}_{j=1}^{N_p}, \{\mathcal{F}_j\}_{j=1}^{N_p}$

Algorithm 3 Phase-Conditioned Guidance and Rollout Collection.

Require: Number of phases N_p, Phase targets $\{\mathcal{P}_j\}_{j=1}^{N_p}$, guided denoising steps K_g, thresholds $\{(\ell_{j,\text{low}}, \ell_{j,\text{high}})\}_{j=1}^{N_p}$.

$j \leftarrow 1$
for $t = 0$ to $T - 1$ **do**
  Sample $A_t^K \sim \mathcal{N}(0, I)$
  for $k = K$ down to 1 **do**
    $\hat{\epsilon} \leftarrow \epsilon_\theta(A_t^k, k)$
    $\hat{A}_t^0 \leftarrow$ Eq. (3)
    $\ell \leftarrow \mathcal{L}(d_\phi(z_t, \hat{A}_t^0), \mathcal{P}_j)$
    if $k \le K_g$ **and** $\ell_{j,\text{low}} < \ell < \ell_{j,\text{high}}$ **then**
      Update $\hat{\epsilon}$ via Eqs. (4)–(5)
    end if
    $A_t^{k-1} \leftarrow$ Eq. (1) with $\hat{\epsilon}$
  end for
  Guidance action $A_t^g \leftarrow A_t^0$. Execute A_t^g
  Update the current phase index j (Sec. 4.2)
end for
return $\mathcal{D}^g = \{(O_t, A_t^g)\}_{t=0}^{T-1}$

D ABLATION STUDIES

This appendix collects the ablations referenced from Table 3. Sections D–D.7 below are ordered as that table lists them.

- **Phase conditioning**: number of phases N_p (Table 5) and number of targets per phase M (Table 6a), both on Transport.

- **Two-threshold gate**: the four gating rules on Transport (Table 6b).
- **Clean-action guidance**: clean action versus noisy iterate as the differentiated variable, on Transport (Table 7).
- **Guidance vs. additional data** (Section D.1): guided versus unguided rollouts at matched volume, under both absorption modes, on Can, Square and Transport.
- **ReGuide-FT vs. ReGuide-FS** (Section D.2): the two variants on the same checkpoint at matched rollout counts, on Can.
- **Per-task rollout-count sweeps**: ReGuide-FS (Section D.3), the composition (Section D.4), and two-iteration ReGuide-FT (Section D.5).
- **Buffer-share ratio** (Section D.6): sensitivity of ReGuide-FT to ρ on Can.
- **Training-seed variance** (Section D.7): five arms retrained end to end under three seeds, on Can and Transport.
- **Observation noise** (Section D.8): the matched-volume arms re-evaluated under state noise, on Can, Square and Transport.

Number of Phases The sweep in the following paragraph varies the number of targets *within* each phase and leaves the phase decomposition intact. Table 5 instead varies the number of phases N_p itself, which makes it a direct ablation of phase conditioning: at $N_p = 1$ the per-phase target sets collapse into a single global set applied for the whole episode—exactly the non-phase-aware setting of LPB (Sun & Song, 2025) and DynaGuide (Du & Song, 2025)—while every other component of PCG is left unchanged, including the two-threshold gate, the clean-action guidance target, and the guidance scale. Because all rows steer the same frozen base checkpoint, this sweep carries no training variance.

At $N_p = 1$ guidance reaches 0.4788 against the 0.4664 of the unguided base policy, a difference of 1.0 SE: with a single global target set, the rest of PCG's machinery buys nothing on Transport. Success then rises with N_p to a peak of 0.5140 at $N_p = 4$, which is $+0.0352$ over $N_p = 1$ (2.5 SE) and $+0.0476$ over the base policy (3.6 SE), before falling back to 0.4904 at $N_p = 5$. Phase conditioning is therefore necessary on this task: the two-threshold gate and the clean-action target yield no gain without it, just as phase-conditioned guidance yields none with only one of the two gate thresholds (Table 6b). Too many phases hurts for the same reason too many targets per phase does: each phase is then fitted from a thinner slice of the demonstration latents, so its centroids become noisier estimates of where the data manifold lies. One caveat bounds the reading. Since M is held at 50 targets per phase, varying N_p also varies the total target count $N_p \times M$. The M sweep in Table 6a partially accounts for this—at $N_p = 4$, raising the total from 200 to 400 and then 600 targets *lowers* success to 0.5126 and 0.4758—so the N_p effect is not explained by total target count alone.

Table 5: **Number of phases N_p on Transport: a direct ablation of phase conditioning.** $N_p = 1$ collapses the per-phase target sets into a single global set applied for the whole episode—the non-phase-aware setting of prior test-time guidance—with the two-threshold gate and the clean-action guidance target left in place, and it performs at base-policy level. Other hyperparameters are fixed at $M = 50$ targets per phase, $k = 40$, $\tau = 0.7$, and thresholds at $50/90$. All rows steer the same frozen base checkpoint, so no training variance enters this comparison. The $N_p = 4$ row is the same run as the $M = 50$ row of Table 6a. Base policy (no guidance): 0.4664 ± 0.0086. Best result in **bold**.

# Phases N_p	Success Rate	vs. base policy
1 (single global target set)	0.4788 ± 0.0098	$+1.0$ SE
2	0.4864 ± 0.0098	$+1.5$ SE
3	0.4700 ± 0.0095	$+0.3$ SE
4	$\mathbf{0.5140} \pm 0.0102$	$\mathbf{+3.6}$ **SE**
5	0.4904 ± 0.0090	$+1.9$ SE

Number of Phase targets In Section 4.1, we construct multiple targets per phase to respect the multimodal nature of expert manipulation behavior. Table 6a evaluates the effect of this choice on Transport. Performance is non-monotone in M: $M = 1$ (one target per phase) and very large M both underperform; $M = 50$ achieves the best success rate.

Intuitively, too few phase targets over-commit the policy to whichever single modes happen to be captured, while too many dilute the gradient signal across loosely-related targets; the middle range

balances these effects. Note that this sweep varies the number of targets *within* each phase and leaves the phase decomposition intact; it is therefore not a test of phase conditioning itself, which the $M = 1$ row still retains.

Guidance Threshold Table 6b compares two gating strategies for test-time guidance: applying guidance whenever the soft-minimum distance exceeds a lower threshold (the standard approach in prior work (Sun & Song, 2025)), versus our two-threshold rule that additionally disables guidance when the distance exceeds an upper threshold. With both thresholds, guidance improves over the no-guidance baseline by $+0.048$ (3.6 standard errors of the difference). With only the lower threshold the improvement is $+0.027$ (2.1 SE) and with only the upper threshold $+0.020$ (1.5 SE), so the full gate is worth about as much as the two single-threshold rules combined, though it exceeds each of them by only 1.4 and 1.9 SE. The two thresholds are complementary, consistent with the three-regime view in Section 4.2: without the upper threshold, guidance also fires when the predicted future state is far from the data manifold, where the gradient toward distant phase targets is unreliable; without the lower threshold, it also fires on rollouts that needed no correction and can perturb a correct action. The benefit of guiding the drifted-but-recoverable band appears only once both regimes are excluded.

Table 6: Test-time guidance ablations on Transport. Base policy (no guidance): 0.4664 ± 0.0086. Best result in each sub-table in **bold**.

# Phase targets M	Success Rate
1	0.4816 ± 0.0098
10	0.4844 ± 0.0098
50	$\mathbf{0.5140} \pm 0.0102$
100	0.4896 ± 0.0095
150	0.4740 ± 0.0097

(a) Number of phase targets M per phase. Other hyperparameters fixed at $N_p = 4$, $k = 40$, $\tau = 0.7$, thresholds at 50/90.

Gating strategy	Success Rate
No guidance	0.4664 ± 0.0086
Lower only (ℓ_{low}=50%)	0.4936 ± 0.0101
Upper only (ℓ_{high}=90%)	0.4864 ± 0.0102
Both (ℓ_{low}=50%, ℓ_{high}=90%)	$\mathbf{0.5140} \pm 0.0102$

(b) Effect of guidance gating thresholds. With $M = 50$, $N_p = 4$.

Guidance Target: Clean Action vs. Noisy Iterate Section 4.2 argues that the gradient guidance should be computed with respect to the estimated clean action $\hat{A}_t^0$ (the MPGD-style choice (He et al., 2024)) rather than the noisy iterate A_t^k (the formulation used by prior diffusion-policy guidance work (Du & Song, 2025; Sun & Song, 2025)), because our dynamics model d_ϕ is trained on clean actions and backpropagating through it with noisy inputs violates its training distribution. Table 7 reports the comparison on Transport with all other guidance hyperparameters held fixed. Clean-action guidance outperforms noisy-iterate guidance on average (0.5140 vs. 0.4788, a gap of 2.4 SE), with the direction consistent across seeds and consistent with the MPGD argument: matching the dynamics model's input distribution is what makes its gradient signal informative.

Table 7: Effect of the guidance target on Transport: differentiating through the dynamics model with respect to the estimated clean action $\hat{A}_t^0$ (ours, MPGD-style) vs. the noisy iterate A_t^k (prior diffusion-policy guidance work). Both share the same phase targets, thresholds, and guidance scale; only the gradient target differs. Mean $\pm$ standard error of the mean.

Guidance target	Success Rate
Noisy iterate A_t^k (Du & Song, 2025; Sun & Song, 2025)	0.4788 ± 0.0103
Clean action $\hat{A}_t^0$ (MPGD-style, ours) (He et al., 2024)	$\mathbf{0.5140} \pm 0.0102$

D.1 Guidance vs. Additional Rollouts

A natural concern is that ReGuide's gains come from extra data rather than guidance specifically. We test this at matched rollout count in both absorption modes of Section 4.3.

ReGuide-FS. Appendix Table 10 covers all four tasks. On Can, Square, and Transport, ReGuide-FS outperforms Filtered BC-FS at every matched retained-success count—at each arm's best count, by 10.1 points on Can, 6.6 on Square, and 7.4 on Transport. Each cell is a single training run, and Appendix D.7 reports how far these margins move under retraining. The Can margin is stable across three training seeds (+0.089, with every ReGuide-FS seed above every Filtered BC-FS seed), whereas the Transport margin does not survive seed averaging (+0.049, 1.1 SE), so we rest this comparison on Can and Square rather than on Transport.

Tool Hang is the exception, and we report it as such: the two arms are indistinguishable at the single count we ran (0.1656 guided against 0.1652 Filtered BC-FS at 20 added rollouts, or $0.25N$). We ran only that count there because the Tool Hang base policy succeeds on 3.3% of episodes, so even 20 successful guided trajectories require a large number of attempts, and extending the sweep would have cost more interaction than the comparison is worth.

ReGuide-FT. Table 8 reports the comparison under ReGuide-FT across all four tasks. On Can, guided rollouts win at every count, with gains of 1.8–4.8 points. On Tool Hang, guided rollouts lead Filtered BC-FT at the matched count of 20 (0.1592 vs. 0.1456; +1.4 points, +9.3% relative), in the same direction as the other tasks. We do not claim this margin as an independent confirmation: at 1.3 rollout-SE it sits within evaluation noise, and because each cell is a single training run the reported interval excludes training variance entirely.

Table 8: **ReGuide-FT vs. Filtered BC-FT at matched rollout counts**, across all four Robomimic tasks. *Filtered BC-FT* is the identical ReGuide-FT procedure run on success-filtered *unguided* base-policy rollouts: same base checkpoint, same task, same buffer-share ratio $\rho = 0.8$, and the same number of *retained successful* trajectories. Because the match is on retained successes rather than raw rollout count, the filtered-BC arm consumed *more* environment episodes than the guided arm to reach each count. Δ is ReGuide-FT minus Filtered BC-FT, in success-rate points; Δ (%) expresses that difference relative to the Filtered BC-FT value. Rollout counts are not tuned per task: they are fixed multiples of each task's demonstration count N, given in parentheses and rounded to the nearest integer where the multiple is fractional (Appendix E). Best result per row in **bold**. Mean ± standard error.

Task	# Added Rollouts	Filtered BC-FT	ReGuide-FT	Δ	Δ (%)
Can (N=15)	0 (base policy)	0.4924 ± 0.0101	—	—	—
	8 (0.5N)	0.5256 ± 0.0111	**0.5436** ± 0.0096	+0.0180	+3.4%
	15 (1.0N)	0.5664 ± 0.0093	**0.5932** ± 0.0098	+0.0268	+4.7%
	23 (1.5N)	0.5376 ± 0.0101	**0.5852** ± 0.0095	+0.0476	+8.9%
Square (N=30)	0 (base policy)	0.5632 ± 0.0091	—	—	—
	15 (0.5N)	0.5484 ± 0.0109	**0.5812** ± 0.0107	+0.0328	+6.0%
	30 (1.0N)	0.6188 ± 0.0114	**0.6592** ± 0.0095	+0.0404	+6.5%
	45 (1.5N)	0.5972 ± 0.0106	**0.6500** ± 0.0101	+0.0528	+8.8%
Transport (N=10)	0 (base policy)	0.4664 ± 0.0086	—	—	—
	5 (0.5N)	0.5404 ± 0.0093	**0.5692** ± 0.0096	+0.0288	+5.3%
	10 (1.0N)	0.5692 ± 0.0088	**0.6456** ± 0.0110	+0.0764	+13.4%
	15 (1.5N)	0.4796 ± 0.0083	**0.5904** ± 0.0092	+0.1108	+23.1%
Tool Hang (N=80)	0 (base policy)	0.0332 ± 0.0030	—	—	—
	20 (0.25N)	0.1456 ± 0.0076	**0.1592** ± 0.0077	+0.0136	+9.3%

Together, the two modes rule out the "just more data" explanation on Can, Square, and Transport: at matched volume, guidance contributes distinct value there, presumably by covering the drifted-but-recoverable region that the base policy reaches but cannot exit alone. On Tool Hang the evidence is weaker—a 1.3-SE lead under ReGuide-FT and a tie under ReGuide-FS—and we do not claim the mechanism is established on that task.

D.2 REGUIDE-FT VS. REGUIDE-FS

The two variants of Section 4.3 play different roles: ReGuide-FS crosses the capacity gap between the base policy and a stronger initialization, while ReGuide-FT refines an already-competent policy. When the loop iterates, that role assignment leaves an engineering question: at each new iteration, should we re-spend compute on another ReGuide-FS run over the accumulated data, or just add another ReGuide-FT round?

Table 9 compares both options on Can at matched rollout counts. Starting from the same ReGuide-FS checkpoint and a newly collected batch of guided rollouts $\mathcal{D}_2^g$, we either (a) run ReGuide-FS on $\mathcal{D}_{\text{demo}} \cup \mathcal{D}_1^g \cup \mathcal{D}_2^g$, or (b) run ReGuide-FT on the merged buffer. Neither dominates, but the two counts disagree: ReGuide-FT is ahead at 8 rollouts by 0.0330 (2.8 SE of the difference), while ReGuide-FS is ahead at 15 by 0.0172 (1.5 SE, within noise). Since the ordering flips between adjacent counts and each cell is a single training run, we read the two as comparable overall rather than reading either count as decisive.

Table 9: ReGuide-FT vs. ReGuide-FS on Can at matched rollout counts, both starting from the same ReGuide-FS checkpoint (success rate 0.6764, row 0) and seeing the same newly collected guided rollouts; ReGuide-FT uses $\rho = 0.8$. Counts are multiples of Can's demonstration count $N = 15$, rounded to the nearest integer. Performance is statistically comparable; ReGuide-FT wins on compute.

# Added Rollouts	ReGuide-FT ($\rho = 0.8$)	ReGuide-FS
0	0.6764 ± 0.0089	—
8 ($0.5N$)	$\mathbf{0.7208} \pm 0.0086$	0.6880 ± 0.0078
15 ($1.0N$)	0.6916 ± 0.0081	$\mathbf{0.7088} \pm 0.0087$

Given comparable performance, ReGuide-FT is the better engineering choice for iterations beyond the composition step: it converges in far fewer steps from an already-competent policy, while ReGuide-FS repeats most of the original training run. This is what makes the iterative loop practical—each additional round is a ReGuide-FT update, not a ReGuide-FS run.

D.3 REGUIDE-FS PER TASK

Per-task data for ReGuide-FS across all four tasks: guided rollouts vs. Filtered BC-FS at matched dataset sizes.

Table 10: **ReGuide-FS vs. Filtered BC-FS at matched dataset sizes**, across all four Robomimic tasks. In both arms the rollouts are merged with the demonstration data and used to train a fresh policy. *Filtered BC-FS* is the identical procedure on success-filtered *unguided* base-policy rollouts, matched on the number of *retained successful* trajectories, so the filtered-BC arm consumed *more* environment episodes to reach each count. Counts are fixed multiples of each task's demonstration count N, given in parentheses and rounded to the nearest integer where the multiple is fractional. Δ is ReGuide-FS minus Filtered BC-FS, in success-rate points; Δ (%) expresses that difference relative to the Filtered BC-FS value. Guided rollouts win at every count on Can, Square, and Transport, and the guided arm peaks at $1.0N$ on all three. Tool Hang is run at a single count: its base policy succeeds on 3.3% of episodes, so collecting even 20 successful guided trajectories takes a large number of attempts, and at that count the two arms are indistinguishable, so neither is marked. Better arm per row in **bold** where the arms separate. Mean $\pm$ standard error.

Task	# Added Rollouts	Filtered BC-FS	ReGuide-FS	Δ	Δ (%)
Can (N=15)	0 (base policy)	0.4924 ± 0.0101	—	—	—
	8 ($0.5N$)	0.5280 ± 0.0099	$\mathbf{0.5324} \pm 0.0100$	$+0.0044$	$+0.8\%$
	15 ($1.0N$)	0.5756 ± 0.0094	$\mathbf{0.6764} \pm 0.0089$	$+0.1008$	$+17.5\%$
	23 ($1.5N$)	0.5528 ± 0.0093	$\mathbf{0.6264} \pm 0.0094$	$+0.0736$	$+13.3\%$
Square (N=30)	0 (base policy)	0.5632 ± 0.0091	—	—	—
	15 ($0.5N$)	0.5628 ± 0.0101	$\mathbf{0.6908} \pm 0.0096$	$+0.1280$	$+22.7\%$
	30 ($1.0N$)	0.6488 ± 0.0083	$\mathbf{0.7144} \pm 0.0101$	$+0.0656$	$+10.1\%$
	45 ($1.5N$)	0.6476 ± 0.0094	$\mathbf{0.6888} \pm 0.0075$	$+0.0412$	$+6.4\%$
Transport (N=10)	0 (base policy)	0.4664 ± 0.0086	—	—	—
	5 ($0.5N$)	0.5436 ± 0.0114	$\mathbf{0.6324} \pm 0.0104$	$+0.0888$	$+16.3\%$
	10 ($1.0N$)	0.5924 ± 0.0092	$\mathbf{0.6820} \pm 0.0114$	$+0.0896$	$+15.1\%$
	15 ($1.5N$)	0.6080 ± 0.0096	$\mathbf{0.6544} \pm 0.0096$	$+0.0464$	$+7.6\%$
Tool Hang (N=80)	0 (base policy)	0.0332 ± 0.0030	—	—	—
	20 ($0.25N$)	0.1652 ± 0.0070	0.1656 ± 0.0078	$+0.0004$	$+0.2\%$

D.4 Composition (ReGuide-FT on top of ReGuide-FS) Across Tasks

Per-task detail for the composition: ReGuide-FT starting from the ReGuide-FS checkpoint at added-rollouts $= N$ ($0.25N$ on Tool Hang, its only count), across all four Robomimic tasks.

Table 11: ReGuide-FT applied across tasks starting from the ReGuide-FS checkpoint at added-rollouts $= N$, $0.25N$ on Tool Hang (the composition). The "Starting ReGuide-FS" column gives that checkpoint's success rate. Best result per task in **bold**.

Task	Starting ReGuide-FS	# Rollouts	Success Rate
Can	0.6764 ± 0.0089	8	0.7208 ± 0.0086
		15	0.6916 ± 0.0081
		23	$\mathbf{0.7228} \pm 0.0083$
Square	0.7144 ± 0.0101	15	0.6528 ± 0.0110
		30	0.6596 ± 0.0104
		45	$\mathbf{0.7336} \pm 0.0091$
Transport	0.682 ± 0.0114	5	0.5896 ± 0.0107
		10	$\mathbf{0.7056} \pm 0.0086$
		15	0.6664 ± 0.0106
Tool Hang	0.1656 ± 0.0078	20	$\mathbf{0.2332} \pm 0.0106$
		40	0.1776 ± 0.0071
		60	0.1976 ± 0.0075
		80	0.1836 ± 0.0076

D.5 ITERATING REGUIDE-FT FROM THE BASE POLICY

Per-task detail across all four tasks for both ReGuide-FT iterations. This is the data underlying Figure 4 and the iteration discussion in the main paper.

Table 12: Iterative ReGuide-FT from the base policy across tasks: (*top*) Iteration 1, starting from the base diffusion policy; (*bottom*) Iteration 2, starting from the Iteration 1 policy at added-rollouts $= N$ ($0.25N$ on Tool Hang). **# Total Guided Rollouts** counts the cumulative guided rollouts in the training set across both iterations; Iteration 2 rows include the Iteration 1 budget that produced its starting policy (Can: 15, Square: 30, Transport: 10, Tool Hang: 20). Best result per task in **bold**.

Task	Base	# Total Guided Rollouts	Success Rate
Can	0.4924 ± 0.0101	8	0.5436 ± 0.0096
		15	$\mathbf{0.5932} \pm 0.0098$
		23	0.5852 ± 0.0095
Square	0.5632 ± 0.0091	15	0.5812 ± 0.0107
		30	$\mathbf{0.6592} \pm 0.0095$
		45	0.6500 ± 0.0101
Transport	0.4664 ± 0.0086	5	0.5692 ± 0.0096
		10	$\mathbf{0.6456} \pm 0.0110$
		15	0.5904 ± 0.0092
Tool Hang	0.0332 ± 0.003	20	$\mathbf{0.1592} \pm 0.007$
		40	0.1568 ± 0.006

(a) Iteration 1 (from base policy).

Task	Iter. 1 Best	# Total Guided Rollouts	Success Rate
Can	0.5932 ± 0.0098	23	0.6012 ± 0.0082
		30	0.5592 ± 0.0087
		38	0.5940 ± 0.0084
		45	0.5836 ± 0.0086
		60	$\mathbf{0.6236} \pm 0.0090$
		75	0.6172 ± 0.0097
		90	0.6036 ± 0.0092
Square	0.6592 ± 0.0095	45	0.6124 ± 0.0107
		60	0.6584 ± 0.0107
		75	0.6612 ± 0.0088
		120	0.7068 ± 0.0088
		150	$\mathbf{0.7120} \pm 0.0085$
		180	0.6580 ± 0.0098
Transport	0.6456 ± 0.0110	15	0.6644 ± 0.0082
		25	0.5656 ± 0.0115
		40	0.6676 ± 0.0081
		50	0.6776 ± 0.0089
		60	$\mathbf{0.6948} \pm 0.0094$
Tool Hang	0.1592 ± 0.007	40	$\mathbf{0.2404} \pm 0.008$
		60	0.2248 ± 0.007
		80	0.1980 ± 0.008
		100	0.2104 ± 0.008

(b) Iteration 2 (from Iteration 1 at N; $0.25N$ on Tool Hang).

Table 13: **Iteration 2 on Square: guided rollouts vs. the matched filtered-BC control.** Both arms run a second ReGuide-FT round from their own iteration-1 policy at 30 added rollouts—0.6592 for ReGuide-FT, 0.6188 for Filtered BC-FT—and the count is *cumulative* over both iterations, so the two sweeps are matched at every row. Δ is ReGuide-FT minus Filtered BC-FT, in units of the standard error of the difference. Each cell is a single training run. Best per arm in **bold**.

Cumulative added rollouts	Filtered BC-FT	ReGuide-FT	Δ
45	0.6136 ± 0.0102	0.6124 ± 0.0107	−0.0012 (0.1 SE)
60	0.5580 ± 0.0107	0.6584 ± 0.0107	+0.1004 (6.6 SE)
75	0.5736 ± 0.0105	0.6612 ± 0.0088	+0.0876 (6.4 SE)
120	0.6888 ± 0.0094	0.7068 ± 0.0088	+0.0180 (1.4 SE)
150	**0.6920** ± 0.0096	**0.7120** ± 0.0085	+0.0200 (1.6 SE)
180	0.6916 ± 0.0087	0.6580 ± 0.0098	−0.0336 (2.6 SE)

D.6 Buffer-Share Ratio Ablation

Ablation over the buffer-share ratio ρ used in ReGuide-FT (Section 4.3).

Table 14: Ablation over buffer-share ratio ρ on Can with 15 added guided rollouts during continual learning; the $\rho = 0.8$ row is the ReGuide-FT iteration-1 result of Table 1. Share denotes the percentage of each minibatch drawn from the newly collected guided rollouts. We fix $\rho = 0.8$ for every task and every experiment in the paper; $\rho = 0.7$ is nominally higher on Can, by 0.0076, or 0.55 standard errors of the difference, which this sweep cannot resolve.

ρ	Success Rate
0.6	0.5800 ± 0.0109
0.7	**0.6008** ± 0.0098
0.8	0.5932 ± 0.0098
0.9	0.5680 ± 0.0090
1	0.5608 ± 0.0110

D.7 Training-Seed Variance

Every success rate elsewhere in the paper is a 2,500-episode evaluation of a single trained checkpoint, so its reported interval is evaluation error at fixed weights: it measures how precisely we have estimated *that* policy's success rate, not how much the policy itself would change if the pipeline were rerun. To measure the second source we retrained five arms end-to-end under three independent training seeds on Can and Transport: the base policy, Filtered BC-FS, ReGuide-FS, Filtered BC-FT, and ReGuide-FT, all at added-rollouts $= 1.0N$. A seed re-draws the base-policy training run, the rollouts collected from it, and the downstream update, so the spread reflects the whole pipeline's variance rather than that of the final fit alone. Each seed is then evaluated over 2,500 episodes as usual, and Table 15 reports the mean over the three seeds with the standard error *across seeds*. That interval already subsumes evaluation noise—a single cell's evaluation standard error of ≈0.010 contributes only ≈0.006 to the mean of three—so the two are not added. Seed 1 is our default training seed: it was used for every run in the paper before this audit and is the run reported in every other table, and seeds 2 and 3 were trained only for this audit.

Table 15: **Mean over three independent training seeds.** Each cell averages three end-to-end reruns of the pipeline, each evaluated over 2,500 episodes; the interval is the standard error *across the three seeds* and therefore reports training variance, whereas every other table in the paper reports evaluation variance at fixed weights. All arms are at added-rollouts $= 1.0N$. *Filtered BC-FS* and *Filtered BC-FT* are the matched baselines of Tables 10 and 8, each placed beside the ReGuide variant it is matched to. ReGuide-FS leads Filtered BC-FS on both tasks, decisively on Can, where the two arms' seed ranges are disjoint, but by only 1.1 SE on Transport; ReGuide-FT leads Filtered BC-FT by 1.4 and 0.5. Best result per row in **bold**.

Task	Base Policy	Filtered BC-FS	ReGuide-FS	Filtered BC-FT	ReGuide-FT
Can (N=15)	0.5387 $\pm$ 0.0233	0.5845 $\pm$ 0.0048	**0.6737** $\pm$ 0.0073	0.5552 $\pm$ 0.0139	0.5780 $\pm$ 0.0082
Transport (N=10)	0.5152 $\pm$ 0.0354	0.6387 $\pm$ 0.0336	**0.6877** $\pm$ 0.0323	0.5603 $\pm$ 0.0063	0.5788 $\pm$ 0.0390

Training variance is the larger of the two sources on these arms. The seed-to-seed standard deviation ranges from 0.008 to 0.068 against an evaluation standard error of $\approx$0.010, and on Transport four of the five arms span 11–14 success-rate points across their three seeds (Filtered BC-FT spans 2). Differences below $\approx$0.05 in the per-count sweeps of this appendix should therefore be read as indistinguishable, and orderings that flip between adjacent counts should not be interpreted. The seed intervals are themselves estimated from three points and are correspondingly imprecise: the $\pm$0.0048 on Can Filtered BC-FS sits at the floor that evaluation noise alone would produce, and should not be read as evidence that that arm is intrinsically more stable than the others.

Averaged over seeds, ReGuide-FS remains ahead of the base policy on both tasks, by $+0.135$ on Can (5.5 SE of the difference) and $+0.173$ on Transport (3.6 SE), and the separation is clean at the level of individual runs: the worst ReGuide-FS seed exceeds the best base-policy seed on both tasks (0.6600 against 0.5668 on Can, 0.6348 against 0.5840 on Transport). The comparison against Filtered BC-FS, which isolates guidance from data volume, separates on Can—$+0.089$, with the two arms' seed ranges disjoint—but not on Transport, where the seed-averaged margin falls to $+0.049$ (1.1 SE) from the $+0.090$ of the single run reported in Table 10. We therefore rest the matched-volume claim on Can and Square rather than on Transport.

ReGuide-FT at $1.0N$ is the arm most affected by seed averaging. Its gain over the base policy is $+0.039$ on Can and $+0.064$ on Transport (1.6 and 1.2 SE), against $+0.101$ and $+0.179$ at seed 1 alone. Against its matched baseline the evidence is directional: ReGuide-FT leads Filtered BC-FT by $+0.023$ on Can (1.4 SE of the difference, ahead on all three seeds) and by $+0.019$ on Transport (0.5 SE, ahead on two of three), so guided data helps ReGuide-FT consistently in sign but not by a margin three seeds can resolve. The headline ReGuide-FT results in Table 1 are iteration 2 and the composition, neither of which is seeded.

D.8 Robustness to Observation Noise

This section gives the full noise sweep summarized in the final row of Table 3. Two readings are worth separating. In absolute terms the guided arm leads in eight of the nine task–noise cells, the exception being Square at $\sigma = 0.012$, where the two arms are statistically tied. In *relative* terms the picture is flatter: on Can and Transport both arms lose the same fraction of their clean-evaluation success as σ grows (47.0% against 45.8% and 53.5% against 52.2% at $\sigma = 0.012$, both under 0.6 SE), so on those tasks guidance buys a constant proportional offset rather than greater robustness; on Square the guided arm degrades measurably faster (56.8% against 51.8%, 2.3 SE), which is what erases its lead there. We note that σ is an absolute perturbation of the state vector and the three tasks differ in dimension and scale, so the levels are not strictly comparable across tasks; the within-task comparisons, which share a σ and an initial-state set, are.

Table 16: **Success rate under varying state noise.** Zero-mean Gaussian noise of standard deviation σ is added to the state observation at evaluation time; both arms are at 15 added rollouts ($1.0N$), so the only difference between them is whether guidance was active during collection. Δ is ReGuide-FS minus Filtered BC-FS in success-rate points, and Δ (%) the same difference relative to the Filtered BC-FS value. The leading arm in each row is in **bold**. For reference the base policies score 0.4924 (Can), 0.5632 (Square) and 0.4664 (Transport) at $\sigma = 0$; they were not evaluated under noise.

Task	State noise σ	Filtered BC-FS	ReGuide-FS	Δ	Δ (%)
Can	0 (none)	0.5756 ± 0.0094	**0.6764** ± 0.0089	+0.1008	+17.5%
	0.006	0.5344 ± 0.0108	**0.6232** ± 0.0110	+0.0888	+16.6%
	0.012	0.3048 ± 0.0097	**0.3668** ± 0.0092	+0.0620	+20.3%
Square	0 (none)	0.6488 ± 0.0083	**0.7144** ± 0.0101	+0.0656	+10.1%
	0.006	0.5764 ± 0.0087	**0.6272** ± 0.0083	+0.0508	+8.8%
	0.012	**0.3124** ± 0.0093	0.3084 ± 0.0097	−0.0040	−1.3%
Transport	0 (none)	0.5924 ± 0.0092	**0.6820** ± 0.0114	+0.0896	+15.1%
	0.006	0.5376 ± 0.0114	**0.6120** ± 0.0100	+0.0744	+13.8%
	0.012	0.2752 ± 0.0092	**0.3260** ± 0.0097	+0.0508	+18.5%

D.9 LIBERO: Analysis and Per-Count Results

This section expands the LIBERO discussion in Section 5: the setting, the full ten-task test-time result, the mechanism behind it, the per-count matched-volume sweep, and the limits of what these experiments establish.

Setting. LIBERO (Liu et al., 2023) changes the starting point rather than the loop. The base policy is a single diffusion policy trained jointly on 10 tasks (500 demonstrations, 50 per task) rather than a per-task policy, one dynamics model serves the whole suite, and both are checkpoints released by LPB (Sun & Song, 2025): we train neither them nor the encoder they share, and contribute only the phase targets, the per-phase gate thresholds, and the ReGuide-FT updates. The phase targets and the per-phase distance distributions are clustered from the pooled latents of all ten tasks rather than from a single task's demonstrations. The setting therefore asks two questions the Robomimic experiments cannot: whether ReGuide can improve an existing pretrained checkpoint on the tasks it handles worst without retraining it, and whether temporally ordered phases remain meaningful when clustered over heterogeneous tasks requiring different steps. It is also a stricter test than Robomimic, where we both train the base policy and fit the dynamics model to that policy's own rollouts, so the LIBERO results cannot be attributed to our choice of base-policy training setup. We select the three tasks with the lowest base-policy success, collect guided rollouts on each, and run ReGuide-FT separately per task; the output is three task-specialized policies, one per target task, rather than a single policy serving all ten. Because fine-tuning draws 80% of each minibatch from the newly collected single-task rollouts, the guided and filtered-BC arms specialize the checkpoint toward the target task to the same degree; that is intended here, and performance on the other nine tasks is neither preserved nor claimed.

Test-time PCG across the suite. Table 18 gives test-time PCG against the shared base policy on all ten tasks of the suite, not a selection. The phases remain usable in the sense that the fine-tuning loop still works, but the test-time guidance component does not transfer to this regime, and we report that directly. PCG improves only Living Room Scene 6 (0.1776 → 0.2124, 3.4 standard errors of the difference) and is statistically indistinguishable on Kitchen Scene 8 and Living Room Scene 5. On six of the remaining seven it *degrades* success by 2.8–4.7 standard errors, and the ten-task mean falls from 0.7348 to 0.7119. Averaged over the suite, test-time PCG is net negative here. Other guidance scales and gate thresholds we tried on LIBERO gave results similar to Table 18, so the degradation is not an artifact of one particular setting.

Why the degradation tracks base-policy competence. The pattern tracks base-policy competence, and the mechanism is the one our gate is designed to prevent. Section 4.2 notes that guidance applied when the predicted future latent is already close to the phase targets is unnecessary and can perturb a correct action; the lower threshold $\ell_{j,\text{low}}$ exists to suppress exactly that. Here both the targets and the distance distributions $\mathcal{F}_j$ that set the thresholds are pooled across ten tasks. For a task on

which the checkpoint is already strong, that pooled reference is miscalibrated: the gate opens on rollouts that needed no correction, and coarse pooled targets then pull them off a trajectory that was already succeeding. Where the policy fails grossly, as on Living Room Scene 6, the same coarse targets supply a useful direction and success improves. The degradation is therefore evidence for the two-threshold design rather than against it, but it also shows that a gate calibrated against a pooled distribution is not a substitute for one calibrated per task.

Per-count results. Across the twelve task–count cells, guided rollouts are significantly better than Filtered BC-FT in six and significantly worse in one, and their mean over the four counts is higher on all three tasks ($+0.014$, $+0.042$, $+0.027$). Table 17 gives the full sweep. ReGuide-FT leads on all three tasks both at each arm's best count (Table 2) and on the mean over counts. A maximum over four evaluations of the same test set is biased upward for both arms. Pooled across all twelve cells the paired difference is $+0.027 \pm 0.016$ ($t_{11} = 1.76$), so the evidence is directional and consistent rather than decisive at any single point.

What these results do not show. Two caveats bound what this shows. First, the useful budget is smaller here than on Robomimic. The rule of adding guided rollouts equal to the demonstration count, which holds on all three Robomimic tasks under both absorption modes, does not transfer: at $1.00N$ the guided arm has returned to base-policy performance on Living Room Scene 5 and Kitchen Scene 8. Fine-tuning a checkpoint trained on 500 demonstrations with up to 50 rollouts of a single task over-specializes sooner than retraining a policy whose entire training set was those N demonstrations. Second, guided-versus-filtered-BC is not the same claim as guided-versus-base. On Living Room Scene 6 every count improves substantially over the base policy, and on Living Room Scene 5 three of four do; but on Kitchen Scene 8 only the $0.75N$ setting exceeds the base policy and the guided mean (0.6778) sits fractionally below it (0.6804), while Filtered BC-FT is clearly below. On that task the result establishes that guidance yields better training data than unguided collection, not that this fine-tuning budget reliably improves the task. Each cell is a single training run, and the filtered-BC arm swings by 6.8 standard errors between adjacent counts on Living Room Scene 5, which is far larger than evaluation noise and indicates that run-to-run variance is a material term in this setting.

A remedy already ruled out. One remedy is ruled out already. Constructing phase targets separately per task did not recover the gains on the stronger tasks: with roughly a tenth of the data per clustering problem, the per-task phase structure is too noisy to cluster reliably. Calibrating only the thresholds per task while keeping pooled targets is a lighter-weight alternative—$\mathcal{F}_j$ is a scalar distribution and needs far less data than k-means does—and we leave it to future work.

Table 17: **ReGuide-FT from a pretrained multi-task checkpoint (LIBERO): guided rollouts vs. Filtered BC-FT at matched volume.** The base policy and dynamics model are the checkpoints released by LPB (Sun & Song, 2025), trained jointly across 10 LIBERO tasks (500 demonstrations, 50 per task); we train neither, and contribute only the phase targets, clustered from the pooled latents of all ten tasks, and the ReGuide-FT updates. The three tasks are those with the lowest base-policy success in Table 18. Guided rollouts are collected per task and ReGuide-FT is run separately for each, so the output is three task-specialized policies rather than one policy serving all ten. *Filtered BC-FT* is the identical procedure on success-filtered *unguided* base-policy rollouts: same checkpoint, same task, same retained-success count, same $\rho = 0.8$. Counts are multiples of that task's 50 demonstrations. z is the difference in units of its own standard error; $|z| > 1.96$ in **bold**. Mean $\pm$ standard error.

Task (base policy)	Added rollouts	Filtered BC-FT	ReGuide-FT	z
Living Room 5 0.4876 ± 0.0115	13 ($0.25N$)	0.5344 ± 0.0099	0.5448 ± 0.0095	$+0.76$
	25 ($0.50N$)	0.4824 ± 0.0094	$\mathbf{0.5484} \pm 0.0104$	$\mathbf{+4.71}$
	38 ($0.75N$)	0.5016 ± 0.0103	$\mathbf{0.5304} \pm 0.0095$	$\mathbf{+2.06}$
	50 ($1.00N$)	0.4992 ± 0.0104	0.4848 ± 0.0119	-0.91
	mean over counts	0.5044	*0.5271*	
Living Room 6 0.1776 ± 0.0076	13 ($0.25N$)	0.2632 ± 0.0099	0.2532 ± 0.0091	-0.74
	25 ($0.50N$)	$\mathbf{0.2972} \pm 0.0095$	0.2540 ± 0.0087	$\mathbf{-3.35}$
	38 ($0.75N$)	0.2984 ± 0.0095	$\mathbf{0.3812} \pm 0.0099$	$\mathbf{+6.03}$
	50 ($1.00N$)	0.2084 ± 0.0095	$\mathbf{0.3448} \pm 0.0086$	$\mathbf{+10.64}$
	mean over counts	0.2668	*0.3083*	
Kitchen 8 0.6804 ± 0.0138	13 ($0.25N$)	0.6044 ± 0.0099	$\mathbf{0.6348} \pm 0.0102$	$\mathbf{+2.14}$
	25 ($0.50N$)	0.6576 ± 0.0087	0.6688 ± 0.0086	$+0.92$
	38 ($0.75N$)	0.6600 ± 0.0090	$\mathbf{0.7380} \pm 0.0093$	$\mathbf{+6.03}$
	50 ($1.00N$)	0.6824 ± 0.0084	0.6696 ± 0.0103	-0.96
	mean over counts	0.6511	*0.6778*	

Table 18: **Test-time PCG on all ten LIBERO tasks.** A single diffusion policy and a single dynamics model, both released by LPB (Sun & Song, 2025) and trained jointly across the suite, are used unmodified; only the phase targets and the per-phase distance distributions that set the gate thresholds are ours, and both are pooled over all ten tasks. Rows are the complete task list, not a selection, split by whether we collect guided rollouts from the task. PCG improves only Living Room Scene 6 ($+3.4$ SE of the difference), is indistinguishable on Kitchen Scene 8 and Living Room Scene 5, and degrades the other seven, six of them by 2.8–4.7 SE; the suite mean falls from 0.7348 to 0.7119. The split is the point: guidance helps where the base policy fails grossly and hurts where it is already strong, which Section 5 explains as a pooled-calibration failure of the lower gate. Mean $\pm$ standard error.

Task	Base Policy	PCG
Tasks we do not collect from		
Kitchen Scene 3: Turn on stove, put moka pot on it	0.9524 ± 0.0070	0.9112 ± 0.0083
Kitchen Scene 4: Put black bowl in bottom drawer, close it	0.8804 ± 0.0098	0.8576 ± 0.0102
Kitchen Scene 6: Put yellow/white mug in microwave, close it	0.8140 ± 0.0087	0.7564 ± 0.0106
Living Room Scene 1: Put alphabet soup & cream cheese in basket	0.8672 ± 0.0076	0.8260 ± 0.0094
Living Room Scene 2a: Put alphabet soup & tomato sauce in basket	0.7264 ± 0.0086	0.6808 ± 0.0105
Living Room Scene 2b: Put cream cheese & butter in basket	0.8404 ± 0.0078	0.8092 ± 0.0093
Study Scene 1: Pick up book, place in caddy back compartment	0.9216 ± 0.0065	0.8892 ± 0.0081
Tasks ReGuide-FT is applied to (lowest base-policy success)		
Kitchen Scene 8: Put both moka pots on stove	0.6804 ± 0.0138	0.6868 ± 0.0128
Living Room Scene 5: Put mugs on left/right plates	0.4876 ± 0.0115	0.4892 ± 0.0113
Living Room Scene 6: Put mug on plate, pudding to its right	0.1776 ± 0.0076	0.2124 ± 0.0069

E SETUP DETAILS

This section collects per-task implementation details deferred from the Setup paragraph in Section 5.

Evaluation protocol. Each reported success rate is computed over 2,500 episodes with distinct initial states, executed as 50 evaluation shards of 50 episodes from a fixed trained checkpoint. We report the mean success rate and the standard error $\mathrm{SE} = \mathrm{std}(\{\hat{p}_s\}_{s=1}^{50})/\sqrt{50}$, where $\hat{p}_s$ is the success rate of shard s. The evaluation initial states are fixed per task and shared across the base policy, LPB, DynaGuide, PCG, ReGuide-FT and ReGuide-FS, and the ablations of Appendix D reuse them, so every cell in the paper rests on 2,500 episodes from that task's shared initial states and all within-task comparisons are paired.

Phase construction and guidance hyperparameters. Table 19 lists every hyperparameter introduced by ReGuide, per task. Each task uses its own values; the Transport ablations in Appendix D show how sensitive PCG is to M, N_p, and the gate thresholds on one of them.

Hyperparameter selection. Each value in Table 19 was chosen per task by sweeping that hyperparameter and keeping the setting with the highest mean success rate on the 2,500 evaluation episodes; the Transport sweeps in Appendix D are the sweeps behind that task's row. Because selection and reporting use the same episodes, every PCG number is the maximum of a sweep on the evaluation set and is optimistic to that extent. The baselines were treated as follows: LPB uses its published Robomimic hyperparameters on Square, Transport and Tool Hang; on Can, which LPB does not report, and for DynaGuide on all four tasks, we swept guidance scale and thresholds by the same procedure and report the best evaluation-set setting. The guidance rows of Table 1 are therefore comparably optimistic.

Table 19: ReGuide hyperparameters.

Symbol	Meaning	Can	Square	Transport	Tool Hang
Phase construction (Alg. 2)					
d	PCA dimension	128	128	128	128
k	# k-means clusters	30	30	40	40
N_p	# phases	3	3	4	4
M	# targets per phase	100	50	50	100
τ	soft-min temperature	0.1	0.7	0.7	0.1
Guidance (Alg. 3)					
ℓ_{low}	lower gate percentile	30%	50%	50%	20%
ℓ_{high}	upper gate percentile	90%	80%	90%	90%
K_g	# guided denoising steps	10	20	30	20
η	guidance scale	0.01	0.01	0.01	0.01
Δ	phase-advance margin	0.2	0.2	0.2	0.2
K	consecutive steps to advance	2	2	2	2
Absorption (Sec. 4.3)					
ρ	buffer-share ratio	80%	80%	80%	80%

Rollout-count sweeps. The number of guided rollouts added per iteration is set as a small multiple of the demonstration count for each task: approximately $0.5\times$, $1\times$, and $1.5\times$ in Iteration 1 for Can, Square, and Transport, with higher multiples in Iteration 2 to extend the iteration-2 sweep (see Table 12). Tool Hang uses smaller relative multipliers ($0.25\times$ and $0.5\times$), and under ReGuide-FS only $0.25\times$, for two reasons: retraining cost is highest on the largest base policy, and that base policy succeeds on only 3.3% of episodes, so each retained guided trajectory costs roughly thirty attempted episodes and the larger multiples were not worth the interaction budget.

Collection protocol and environment-episode cost. Rollouts are collected in fixed passes of 50 episodes. Within a pass, trajectories that reach task success are written to a file and failures are discarded; for each checkpoint round we run as many passes as are needed for the retained pool to cover the largest count in that round's sweep. The retained successes are then merged and

shuffled once, and every entry of that round's sweep is the first k trajectories of that single shuffled ordering. Two consequences are worth stating. First, subsetting is uniform at random rather than by return, trajectory length, or any guidance-derived quantity, so no quality-based selection enters the comparison. Second, the counts within one sweep are nested prefixes of one pool, so they differ only in data volume with the collection draw held fixed—a clean data-volume ablation, but not independent draws: the three counts in a task's row block of Tables 8 and 10 share one collection run, and the iteration-1 guided pool is the same pool for both tables.

Because retention is success-filtered, the environment episodes attempted exceed the retained counts by the reciprocal of the collecting policy's success rate, and it is the attempted episodes, not the retained ones, that set the interaction budget a real robot would pay. Table 20 reports them for every collection pool. Two readings matter. The *total guided* row is what our experiments consumed, including the counts we explored and discarded: 250, 450, and 200 episodes on Can, Square, and Transport, and 2,100 on Tool Hang, whose base policy succeeds on 3.3% of episodes and which therefore dominates the budget. The final row is what a practitioner following the $1N$ rule pays, collecting only the recommended count from the base policy in a single iteration: 50, 100, 50, and 550 episodes. On LIBERO the guided and unguided pools behind Table 17 cost 150 and 150 episodes on Living Room 5, 250 and 300 on Living Room 6, and 100 and 100 on Kitchen 8. Guided collection is cheaper than unguided collection at every matched count, since PCG raises the collecting policy's success rate, but on Can and Square the margin is small—1.8% and 3.6% fewer attempted episodes per retained trajectory—and only on Transport (10.2%) and Tool Hang (15.7%) is it appreciable. The advantage of guided collection is in what the retained trajectories contain, not in how cheaply they are obtained.

Table 20: **Environment episodes attempted per collection pool**, Robomimic. Each pool is one round of collection from a named checkpoint: passes of 50 episodes, run until the retained successes cover the largest count in that round's sweep, then merged and shuffled once and consumed as nested prefixes (see text). The iteration-1 guided pool serves both Table 8 and Table 10 and is counted once. *Total guided* sums the three guided pools and is the full guided interaction cost of our Robomimic experiments, sweeps included. *Deployment cost* is what the $1N$ rule requires instead: one iteration, one pool, the recommended count only ($0.25N$ on Tool Hang). Pass counts are reconstructed as $\lceil k_{\max}/(50\,\hat{p}) \rceil$ from each collecting policy's measured success rate $\hat{p}$ (Tables 1 and 12); because realized per-pass yield fluctuates, entries near a pass boundary may differ from the logged count by one pass (± 50 episodes).

Collection pool (collecting policy)	Can	Square	Transport	Tool Hang
Iteration 1, guided (base policy)	50	100	50	1,050
Iteration 1, unguided (filtered BC baseline)	50	100	50	1,250
Dynamics model, unguided (base policy)	300	300	300	300
Iteration 2, guided (ReGuide-FT iteration 1 at $1N$)	150	250	100	550
Composition, guided (ReGuide-FS at $1N$)	50	100	50	500
Total guided	250	450	200	2,100
Deployment cost at the $1N$ *rule*	50	100	50	550
incl. dynamics-model data	350	400	350	850

Dynamics-model training. Each Robomimic dynamics model is trained once on 300 unlabeled rollouts of the base policy, successes and failures alike, and is reused unchanged by every arm and both iterations. This budget was fixed a priori and is the one hyperparameter of ReGuide we did not sweep; it is identical across tasks and arms. Table 20 lists these episodes separately. They are not part of the matched-volume comparisons of Appendix D.1: those match the number of retained successful trajectories used to train the policy, and the filtered-BC arm, which needs no dynamics model, consumes 300 fewer environment episodes in total than the guided arm at every count.

Buffer-share ratio. For both ReGuide-FT and the FT step of the composition we use a merged buffer with $\rho \in [0.7, 0.8]$, with ρ chosen best-per-task and the newly collected guided rollouts weighted more heavily in each minibatch. The ratio is a stability/plasticity knob: a higher proportion of original demonstration data preserves learned behavior and keeps the update stable but limits how much new behavior the policy can absorb, while a higher proportion of new rollouts accelerates

learning of new behavior at the risk of distributional shift and forgetting. The choice is mild within our chosen range—the ablation in Table 14 shows performance is roughly flat on Can.

Supervision assumption. The rollout-collection step filters for successful trajectories, so the pipeline assumes access to a binary task-success signal at trajectory completion. This is the same assumption made implicitly by any self-improvement scheme that selects among self-collected data, and is strictly weaker than DAgger-style per-state expert relabeling: success-filtering needs one bit per trajectory, not an action label at every visited state. In Robomimic this signal is provided by the simulator; in deployment it would come from a task-specific success classifier, a sparse reward, or human verification at episode end.