DS-GRPO: NOISE-SPACE ADAPTATION OF DIFFUSION POLICIES WITH ACTION-SPACE VERIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion steering adapts a pretrained generative policy without altering its weights: a small steering policy chooses the initial noise of the frozen sampler, re-weighting the base policy's behaviors within their support. Yet the signals one could adapt to, e.g., a demonstration, a success, a distance to a target, are all only verifiable in action space, while steering optimizes in noise space. Existing methods resolve this mismatch with different trade-offs. Critic-based noise-space RL verifies in action space but pays the cost of exploration and value estimation through environment transitions. Inversion-based steering instead maps demonstrated actions back to noise space, regressing on noise labels rather than minimizing action-space distance. We propose DS-GRPO, combining the benefits of both supervision from demonstrations and direct verification in action space. DS-GRPO samples a group of noises from the steering policy, decodes them through the frozen sampler, and uses the demonstration as an action-space verifier to score the resulting actions. These scores guide steering-policy updates through group-relative policy gradients. Across simulation and real-world experiments including offline behavior cloning, goal-conditioned behavior cloning, and online DAgger, DS-GRPO matches or outperforms pre-existing steering methods on most tasks, providing a concrete step towards efficient yet effective policy adaptation.

1 INTRODUCTION

Generative control policies (Chi et al., 2025; Pan et al., 2026), such as diffusion- and flow-based policies (Black et al., 2024), have largely broadened the capabilities of robot learning. By capturing complex, multi-modal action distributions and being robust in high-dimensional input or output spaces, they have become a prevailing paradigm for highly performant policies, as well as a crucial building block for embodied foundation models (Black et al., 2024; Intelligence, 2025; Ye et al., 2026; Barreiros et al., 2026). However, as these models get increasingly capable by scaling pretraining data and model parameters (Barreiros et al., 2026), the cost of adapting these models to new tasks and scenarios also becomes prohibitively high. As a consequence, efficient adaptation methods (Ankile et al., 2025a; Xu et al., 2026a; Su et al., 2026) have garnered increasing attention.

One prevailing approach for such adaptation is *Diffusion Steering* (Wagenmaker et al., 2025; Patil et al., 2026a; Seo et al., 2026). Instead of directly altering the weights of a pretrained policy $\pi_{\text{base}} : \mathcal{S} \times \mathcal{W} \mapsto \mathcal{A}$, it learns a steering policy $\pi^{\mathcal{W}}$ that adjusts the latent noise the base policy takes in. This essentially serves as a reweighting of the base policy's distribution, offering adaptation in a *parameter-efficient* and *support-constrained* fashion (Yu et al., 2026b).

However, there lies a conflict that every steering method has to resolve. Steering policies *optimize* within the *noise space* of the frozen sampler, but every signal one could adapt to, e.g., an expert demonstration, a success flag, a distance to a target, is only *verifiable* in *action space*. Prior works resolve this tension in one of two ways. One line keeps the objective in action space and pays for it with reinforcement learning: the steered policy is rolled out, the environment scores the result, and a critic carries the score back to noise space (Wagenmaker et al., 2025; Su et al., 2026). While it maintains direct verification in action space, it comes at the cost of a relatively large number of environment transitions, since accurate value propagation and policy extraction require sufficient data coverage (Chen & Jiang, 2019; Du et al., 2019). The learning signal comes solely from environment rewards during rollouts, whereas dense supervision from demonstrations is often neglected.

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.

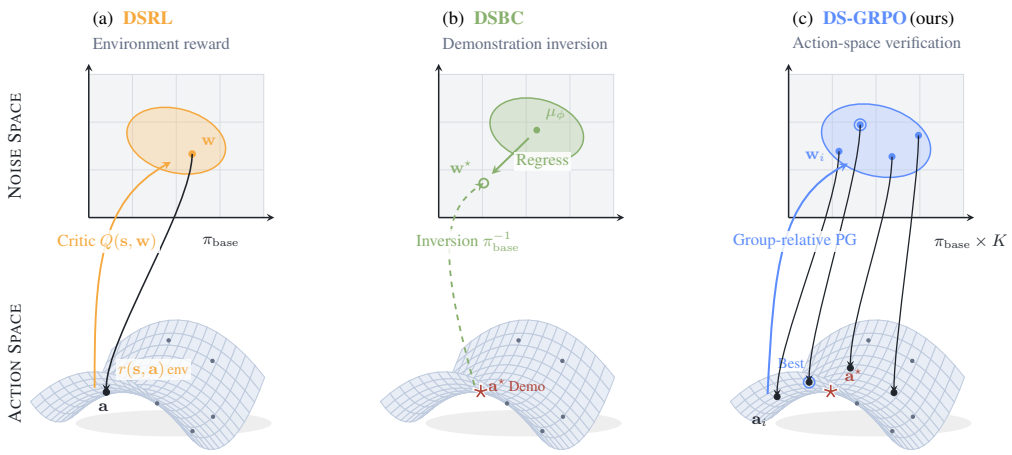


Figure 1: Three ways to train a noise-space steering policy. *(a)* DSRL: environment reward, carried back by a critic over the noise. *(b)* DSBC: the demonstration is inverted into a noise label used as the regression target. *(c)* DS-GRPO decodes K sampled noises through the frozen π_{base} and scores them against the demonstration $\mathbf{a}^\star$ in action space, using critic-free group-relative policy gradients to learn from demonstrations.

The other line learns from dense supervision by inverting demonstrations through the sampler into noise labels, and fitting the steering policy to these noise labels (Tang et al., 2026; Murray et al., 2026; Lu et al., 2026). While this enables learning from demonstrations, it minimizes Euclidean distance in noise space rather than action space, which may lead to suboptimal action-space fitting. Furthermore, it induces additional problems. First, the regression targets the conditional mean of the labels, and wherever the inverted labels are multimodal, the mean maps to an action that is not necessarily close to the expert output. Moreover, even for deterministic samplers, analytically solving for the noise corresponding to the expert action is intractable (Murray et al., 2026); the solution can only be approximated numerically, for example by fixed-point iteration.

In this paper, we argue that action-space verification and dense supervision through demonstrations can co-exist, introducing **DS-GRPO**. The key idea is to *treat demonstrations as action-space verifiers*, where policy proposals can be scored and ranked based on their distance to the expert action outputs. At each state we draw a group of noises from the steering policy, decode them through the frozen base policy, score the resulting actions against the demonstration, and update the steering policy with a group-relative policy gradient (Shao et al., 2024). By tying steering policy updates to demonstration agreement in action space, DS-GRPO avoids the metric mismatch introduced by noise-space regression, while still being able to learn from demonstrations.

Across a diverse set of environments and tasks (Liu et al., 2023; Mandlekar et al., 2021; Heo et al., 2023; Park et al., 2025), spanning both simulation and the real world, we validate DS-GRPO as an efficient yet highly performant adaptation method. For offline and goal-conditioned behavior cloning, DS-GRPO improves success rates of the base policies across multiple tasks and consistently outperforms DSBC. For online DAgger, DS-GRPO surpasses FlowDAgger, a pre-existing diffusion steering DAgger method, in final success rate on every task at the same transition budget, while being significantly more performant than DSRL on long-horizon assembly tasks. On two real-world dexterous manipulation tasks, offline adaptation from human intervention data via DS-GRPO improves success over the base policy and exceeds DSBC by a clear margin across both tasks.

Our contributions are:

- We identify a mismatch at the core of diffusion steering: improvement acts in noise space while verification lands in action space, leading to tradeoffs in prior methods.
- We propose **DS-GRPO**, combining direct action-space verification and dense supervision from expert demonstrations, enabled through group-relative policy gradients.
- We evaluate **DS-GRPO** across multiple tasks and setups in simulation and the real world, showcasing its effectiveness as an efficient yet highly performant adaptation method.

2 PRELIMINARIES

2.1 DIFFUSION- AND FLOW-BASED POLICIES

Flow matching (Lipman et al., 2022; Liu et al., 2022) learns a time-dependent velocity field that transports a simple base distribution to a target distribution. For robotic control (Black et al., 2024; Chi et al., 2025), the target is the conditional distribution of an action chunk $\mathbf{a} \in \mathbb{R}^{H\times A}$ given an observation $\mathbf{s}$. Given demonstrations $(\mathbf{s}, \mathbf{a}) \sim \mathcal{D}$, a noise sample $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ of the same shape, and $t \sim \mathcal{U}[0, 1]$, the interpolant $\mathbf{a}_t = (1-t)\,\mathbf{w} + t\,\mathbf{a}$ is formed and the velocity network is regressed onto the straight-path target:

$$\mathcal{L}(\theta) = \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D},\ \mathbf{w}\sim\mathcal{N}(0,I),\ t} \big\| \mathbf{v}_\theta(\mathbf{a}_t, t, \mathbf{s}) - (\mathbf{a} - \mathbf{w}) \big\|^2. \tag{2.1}$$

At inference, an action is produced by integrating the learned field from the noise. For flow-based robot policies, this is often done via Euler integration across N steps:

$$\mathbf{a}^{(k+1)} = \mathbf{a}^{(k)} + \Delta t_k\, \mathbf{v}_\theta\big(\mathbf{a}^{(k)}, t_k, \mathbf{s}\big), \qquad \mathbf{a}^{(0)} = \mathbf{w}, \qquad \mathbf{a} := \mathbf{a}^{(N)}. \tag{2.2}$$

The point we rely on is that, once the sampler is fixed, Eq. (2.2) is a *deterministic* map from the initial noise to the action. We write it as

$$\mathbf{a} = \pi_{\text{base}}(\mathbf{s}, \mathbf{w}), \qquad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{2.3}$$

so that the base policy, as a distribution over actions, is the push-forward of a standard Gaussian through $\pi_{\text{base}}(\mathbf{s}, \cdot)$. Diffusion policies with a deterministic sampler such as DDIM (Song et al., 2020) have the same form, and nothing below depends on which of the two produced the map.

2.2 DIFFUSION STEERING

Diffusion steering (Wagenmaker et al., 2025) adapts a pretrained generative policy by reweighting its noise distribution. It replaces the Gaussian in Eq. (2.3) with a learned *Steering Policy* over the noise,

$$\mathbf{w} \sim \pi_\phi^{\mathcal{W}}(\cdot \mid \mathbf{s}), \qquad \mathbf{a} = \pi_{\text{base}}(\mathbf{s}, \mathbf{w}). \tag{2.4}$$

The steered policy is the push-forward of $\pi_\phi^{\mathcal{W}}(\cdot \mid \mathbf{s})$ through the same frozen map. Since $\pi_{\text{base}}(\mathbf{s}, \cdot)$ is unchanged, steering only *re-weights* the base policy's behaviors and cannot leave their support. The steering policy is typically a small network, parameterized as a Gaussian $\mathcal{N}\big(\mu_\phi(s), \operatorname{diag}\sigma_\phi^2(s)\big)$.

What distinguishes existing steering methods is the objective that trains $\pi_\phi^{\mathcal{W}}$. **RL-based methods** such as DSRL (Wagenmaker et al., 2025) optimize an environment return through the sampler,

$$\max_\phi\ \mathbb{E}\Big[\textstyle\sum_t \gamma^t\, r(s_t, \mathbf{a}_t)\Big], \qquad \mathbf{w}_t \sim \pi_\phi^{\mathcal{W}}(\cdot \mid \mathbf{s}_t),\ \mathbf{a}_t = \pi_{\text{base}}(\mathbf{s}_t, \mathbf{w}_t), \tag{2.5}$$

with an off-policy actor-critic whose critic is defined over the noise, $Q(\mathbf{s}, \mathbf{w})$, so that the return can be attributed to w without differentiating through π_{base}.

BC-based methods use demonstrations $(\mathbf{s}, \mathbf{a}^\star)$ and bring them into noise space by *inverting* the sampler: integrating Eq. (2.2) backward from $\mathbf{a}^\star$ (or solving the corresponding fixed-point problem) yields a noise label $w^\star$ with $\pi_{\text{base}}(\mathbf{s}, \mathbf{w}^\star) \approx \mathbf{a}^\star$, onto which the steering policy is regressed,

$$\mathbf{w}^\star = \pi_{\text{base}}^{-1}(\mathbf{s}, \mathbf{a}^\star), \qquad \min_\phi\ \mathbb{E}_{(\mathbf{s},\mathbf{a}^\star)}\Big[-\log \pi_\phi^{\mathcal{W}}(\mathbf{w}^\star \mid \mathbf{s})\Big]. \tag{2.6}$$

The steering policy is learned either offline on a fixed dataset (Tang et al., 2026) or online on states visited by the learner with an expert in the loop (Murray et al., 2026; Lu et al., 2026).

3 DS-GRPO: STEERING IN NOISE SPACE, VERIFYING IN ACTION SPACE

DS-GRPO keeps the *form* of Eq. (2.5), where the score operates in action space, and the *data* of Eq. (2.6), enabling learning from demonstrations. In Section 3.1, we formulate the optimization objective, and in Section 3.2, we show a tractable algorithm for optimizing this objective through group-relative policy gradients. Importantly, this results in a learning algorithm that uses only forward passes through the frozen sampler, avoiding the drawbacks of sampler inversion.

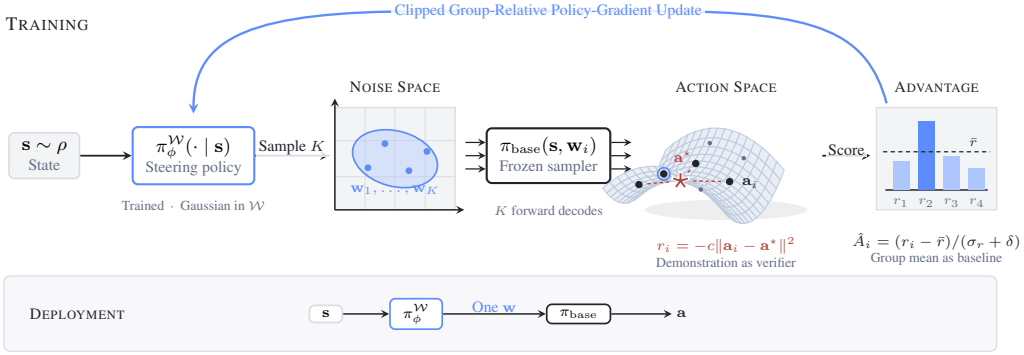


Figure 2: DS-GRPO optimizes against a demonstration action by sampling K noises, decoding them forward through the frozen π_{base} and scoring them based on distance to $\mathbf{a}^\star$ in action space. During deployment, we sample through the steering policy $\pi_\phi^{\mathcal{W}}$ and decode with the base policy π_{base}. See Algorithm 1.

3.1 Demonstrations as Action-Space Verifiers

Inversion-based methods use a demonstration as a *label*: they ask which noise would have produced $\mathbf{a}^\star$ and pull the steering policy toward that noise. We use the same demonstration as a *verifier*: it scores action chunks that the steered policy actually produces based on their distance to the expert action, and assigns a reward to each sample. The verifier is the negative squared distance to the demonstrated action,

$$r(\mathbf{a}; \mathbf{s}) \;=\; -c\,\big\|\mathbf{a} - \mathbf{a}^\star(\mathbf{s})\big\|^2, \tag{3.1}$$

taken in action space, with a fixed normalization factor $c > 0$ for each backend (Section A.1). We maximize the expected verifier score over demonstration states,

$$\max_\phi \; J(\phi) \;=\; \mathbb{E}_{\mathbf{s}\sim\rho}\, \mathbb{E}_{\mathbf{w}\sim\pi_\phi^{\mathcal{W}}(\cdot|\mathbf{s})}\Big[\, r\big(\pi_{\text{base}}(\mathbf{s}, \mathbf{w}); \mathbf{s}\big)\Big], \tag{3.2}$$

where ρ is a distribution over states that carry demonstrations, either dataset states offline, or states collected with a queryable expert online, held fixed during each policy update.

3.2 Group-Relative Policy Gradient Through a Frozen Sampler

We optimize Eq. (3.2) with its score-function gradient, obtained through the log-derivative trick:

$$\nabla_\phi J \;=\; \mathbb{E}\Big[\, r\big(\pi_{\text{base}}(\mathbf{s}, \mathbf{w}); \mathbf{s}\big)\, \nabla_\phi \log \pi_\phi^{\mathcal{W}}(\mathbf{w} \mid \mathbf{s})\Big]. \tag{3.3}$$

Crucially, this formulation treats the sampler as a black box, where candidates can be sampled, decoded, scored and re-weighted, and nothing is ever differentiated through π_{base}.

Following the same formulation as GRPO (Shao et al., 2024), we enable critic-free updates through sampling from a group of candidates, and optimize the policy directly using the relative rewards within that group. More concretely, at each sampled state $\mathbf{s}$, we draw a group of K noises $\mathbf{w}_1, \ldots, \mathbf{w}_K \sim \pi_{\phi_{\text{old}}}^{\mathcal{W}}(\cdot \mid \mathbf{s})$, decode each one into action space as $\mathbf{a}_i = \text{stopgrad}\big(\pi_{\text{base}}(\mathbf{s}, \mathbf{w}_i)\big)$, score them with the reward function $r_i = r(\mathbf{a}_i; \mathbf{s})$, and standardize within the group:

$$\hat{A}_i \;=\; \frac{r_i - \bar{r}}{\sigma_r + \delta}, \qquad \bar{r} = \tfrac{1}{K}\textstyle\sum_j r_j, \quad \sigma_r = \big(\tfrac{1}{K-1}\textstyle\sum_j (r_j - \bar{r})^2\big)^{1/2}. \tag{3.4}$$

Because all K candidates share the state and the demonstration, their scores are directly comparable: the group average serves as a state-specific baseline and avoids a learned value network at the cost of K forward decodes. The steering policy is updated by maximizing the clipped surrogate

$$\mathcal{L}(\phi) \;=\; \mathbb{E}_{\mathbf{s}\sim\rho}\, \frac{1}{K}\sum_{i=1}^{K} \min\Big(\rho_i(\phi)\,\hat{A}_i,\; \text{clip}\big(\rho_i(\phi),\, 1-\varepsilon,\, 1+\varepsilon\big)\,\hat{A}_i\Big), \rho_i(\phi) = \frac{\pi_\phi^{\mathcal{W}}(\mathbf{w}_i \mid \mathbf{s})}{\pi_{\phi_{\text{old}}}^{\mathcal{W}}(\mathbf{w}_i \mid \mathbf{s})}. \tag{3.5}$$

Following DAPO (Yu et al., 2026a), we add no KL penalty and keep no reference policy. To avoid variance collapse, we lower-bound each component of the learned standard deviation $\boldsymbol{\sigma}_\phi$ by σ_{min}.

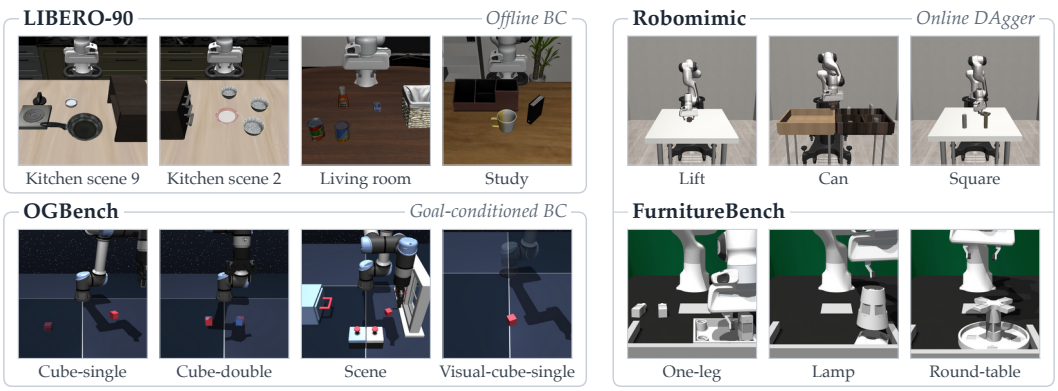


Figure 3: Representative Simulation Environments. We use LIBERO-90 for Offline BC, OGBench for Goal-conditioned BC, and Robomimic as well as FurnitureBench for Online DAgger experiments.

4 Experiments

In this section, we aim to answer the following three questions through our experiments:

- **(Q1)** How does DS-GRPO perform compared to noise-space regression for offline adaptation?
- **(Q2)** Does this method transfer to better performance or efficiency for online adaptation?
- **(Q3)** Is DS-GRPO a practical and effective approach for real-world tasks?

4.1 Experimental Setup

Simulation Environments and Task Setups. We showcase representative simulation environments used for evaluation in Figure 3. For Offline BC, we adapt a pretrained policy with per-task successful trajectories in `LIBERO-90` (Liu et al., 2023). Following Tang et al. (2026), we use generalist control policies fine-tuned on the remaining sets of LIBERO to serve as base policies, including the VLA models $\pi_{0.5}$ (Intelligence, 2025) and StarVLA (Community, 2026), and the World-Action Model Fast-WAM (Yuan et al., 2026). For Goal-conditioned BC, we select four manipulation tasks in `OGBench` (Park et al., 2025): `cube-single`, `cube-double`, `scene` and pixel-based `visual-cube-single`. The base policy is a chunked flow prior trained on data without goals, and the aim is to learn a goal-conditioned steering policy that adapts the goal-agnostic base. For Online DAgger, we use 3 tasks in *Robomimic* (Mandlekar et al., 2021) (`lift`, `can`, `square`) and 3 tasks in *FurnitureBench* (Heo et al., 2023) (`one_leg`, `lamp`, `round_table`). The base policies and expert policies are taken from DPPO (Ren et al., 2025), where the base policy is trained via offline supervised learning, and the expert policy is fine-tuned from the base policy via DPPO.

Real-World Experimental Setup. For real-world experiments, we focus on improving a policy's performance via human-intervention data. We attach a pair of dexterous Wuji hands onto a bimanual arm as our hardware setup, and experiment on two tasks: `Pick-and-Place` and `Peg-Insertion`, as shown in Fig. 6. The base policy is acquired by supervised fine-tuning of $\pi_{0.5}$ on the task's teleoperated demonstrations, and we collect human intervention data by rolling out the base policy and letting human experts take over when early signs of failure appear. We then learn through diffusion steering via offline supervised learning on the human intervention data and test the success rate of the adapted policy following the same protocol as base policy evaluation.

Baselines. For offline BC, goal-conditioned BC and real-world tasks, we compare against the noise-space regression method: DSBC (Tang et al., 2026), which inverts demonstrations into noise labels by approximating the reversal with the end-point velocity and regresses a Gaussian noise policy onto them. In the online DAgger settings the baseline is FlowDAgger (Murray et al., 2026), which inverts expert corrections via per-step fixed-point iteration and regresses a deterministic noise policy by MSE; there we additionally compare against DSRL (Wagenmaker et al., 2025), a critic-based alternative that learns through reinforcement learning from environment rewards.

Table 1: Average Success Rate on LIBERO-90. Mean ± s.d. of success rate averaged across 89 tasks, each evaluated on 50 trials, over two training seeds.

Base policy	Base	DSBC	DS-GRPO
$\pi_{0.5}$	28.2	43.5±0.1	**53.4**±0.3
StarVLA	18.8	18.1±0.3	**52.9**±0.6
Fast-WAM	13.0	13.7±0.0	**17.7**±0.4
Average	20.0	25.1	**41.3**

Table 2: Success Rate on OGBench. Mean ± s.d. over three seeds; `GA-BC` is the goal-agnostic prior.

Environment	GA-BC	DSBC	DS-GRPO
Cube-single	10.4	84.1±3.7	**93.4**±1.9
Cube-double	1.6	8.8±1.2	**28.2**±2.4
Scene	2.4	38.8±2.4	**42.6**±2.0
Visual cube-single	4.0	74.0±3.0	**91.4**±1.1
Average	4.6	51.4	**63.9**

4.2 Offline Adaptation: Single-Task Behavior Cloning

To evaluate DS-GRPO's effectiveness in single-task offline adaptation, we utilize LIBERO-90 (Liu et al., 2023) as our testbed. Following the evaluation protocol of Tang et al. (2026), we use generalist embodied policies trained on the remaining LIBERO suites and adapt each policy separately to each LIBERO-90 task using the task's successful demonstrations. For our evaluation, we use two representative VLA baselines: $\pi_{0.5}$ and StarVLA, as well as one representative WAM baseline: Fast-WAM. These baselines are acquired by training on the remaining datasets for LIBERO, either via supervised fine-tuning or via training from scratch. Both steering networks are trained from scratch with two random training seeds; DSBC follows the exact recipe of Tang et al. (2026) (12,500 updates of batch 128), and DS-GRPO runs 2,000 training iterations of 16 states with $K = 8$ candidates each. Since one task contains no successful episode, we evaluate on the remaining 89 tasks. As shown in Table 1, DS-GRPO improves every base policy and consistently outperforms DSBC on all evaluated base policies, demonstrating its effectiveness for offline single-task adaptation.

4.3 Offline Adaptation: Multi-Task Goal-Conditioned Steering

Furthermore, we investigate whether we can learn a general steering policy applicable to multiple tasks. Here, we focus on offline learning with an external goal condition, using manipulation tasks in `OGBench` (Park et al., 2025), in which different goals share the same environment, but success is measured based on whether the corresponding goal is satisfied. The base prior is trained on the task's play data without ever seeing a goal; one goal-conditioned steering network per environment then reweights this frozen distribution conditioned on the goal variable via offline behavior cloning on all demonstrations in this environment. Importantly, in order to perform well in this evaluation setup, the algorithm must learn a general mapping that can be effective across multiple tasks based on the conditioned goal. DS-GRPO is trained under exactly DSBC's budget (100k updates $\times$ 512 states per update), with $K = 32$ candidates per state as the estimator's group size. Each result averages success over 50 evaluation episodes per goal, the last three checkpoints, and three seeds. As shown in Table 2, by steering a goal-agnostic base policy, DS-GRPO substantially improves the prior's near-zero success rate while outperforming DSBC in every evaluated environment.

4.4 Online Adaptation: Continual Improvement Through DAgger

To evaluate whether DS-GRPO's superior performance in offline adaptation transfers to the online setting, we evaluate using two benchmarks: `Robomimic` (Mandlekar et al., 2021) (`lift`, `can`, `square`), a widely adopted robot manipulation benchmark, and `FurnitureBench` (Heo et al., 2023) (`one_leg`, `lamp`, `round_table`), a highly challenging benchmark for long-horizon high-precision assembly tasks. For the DAgger-based methods, we assume access to an expert policy. During online rollouts of the current steered policy, we query the expert for actions at visited states and update the steering network using the aggregated state–action buffer, while keeping the base policy frozen. For each task, the base policy is the frozen diffusion policy released with DPPO (Ren et al., 2025), and the teacher is the same checkpoint fine-tuned with DPPO's recipe. Every arm runs to the same environment-step budget per task (200K / 1.2M / 2.5M on robomimic, 10M on FurnitureBench) over three training seeds, and we report the final evaluation at the budget end (200 episodes on robomimic, 512 on FurnitureBench), as well as learning curves. Because the samplers in DPPO clamp their predictions to ± 1 at each denoising step, the map from noise to action is many-to-one and its exact inverse is not unique. We therefore seed FlowDAgger's fixed-point solve at the

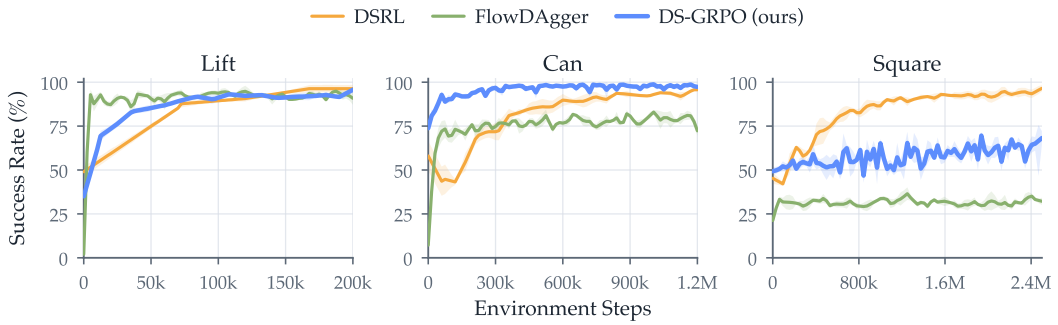


Figure 4: Success Rate versus Environment Steps on Robomimic. Mean over three seeds (shaded: ±1 s.d.).

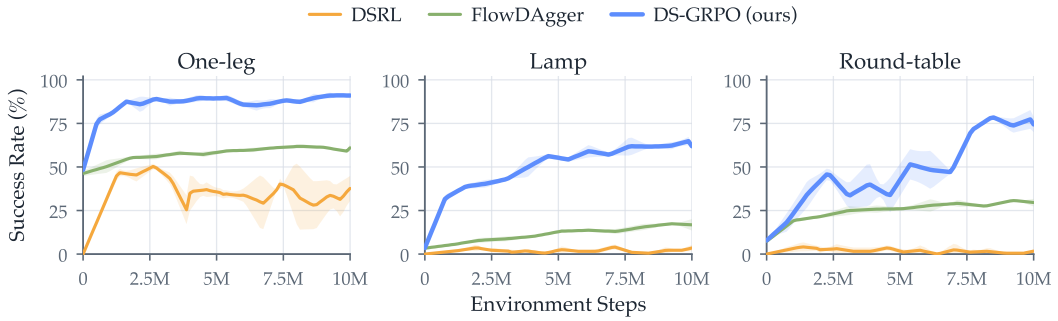


Figure 5: Success Rate versus Environment Steps on FurnitureBench. Same arms and protocol as Fig. 4.

actor's current prediction and filter inversion labels, discarding those whose decoded-action MSE against the expert chunk exceeds 10^{-3}. DS-GRPO and DSRL only decode and thus need no such adjustment.

We report the success rates at the end of training in Table 3, and the learning curves in Figs. 4 and 5. On `Robomimic`, DS-GRPO outperforms FlowDAgger on all three tasks, while being superior to or on par with DSRL on all tasks except `square`. On `FurnitureBench`, a long-horizon sparse-reward task, DSRL significantly underperforms DAgger-based methods. Among DAgger-based methods, DS-GRPO outperforms FlowDAgger by a large margin in every evaluated task across different checkpoints during training, demonstrating effective online adaptation.

Table 3: Success Rate on Online DAgger. Mean ± s.d. over three seeds at each task's shared budget. Bold: within one s.d. of the best entry in each row.

Task	DSRL	FlowDAgger	DS-GRPO
	Robomimic		
`lift`	**96.3**±2.1	90.8±3.7	**95.7**±1.5
`can`	95.7±0.6	72.3±2.9	**97.3**±0.8
`square`	**96.5**±1.8	32.2±2.1	68.2±4.0
	FurnitureBench		
`one_leg`	37.7±7.2	61.0±2.0	**91.0**±1.2
`lamp`	3.5±1.8	16.8±3.3	**61.9**±1.0
`round_table`	1.7±0.6	29.6±2.2	**74.4**±4.4

4.5 Real-World Adaptation from Human Interventions

We then examine whether DS-GRPO is a practical and effective approach for real-world robot adaptation. Following the real-world setup in Section 4.1, we obtain the base policy by fully fine-tuning $\pi_{0.5}$ (Intelligence, 2025) using 200 episodes for the pick and place task, and 100 episodes for the peg insertion task. We then adapt each base policy offline using 40 human-intervention episodes per task via diffusion steering. For the `Pick-and-Place` task, we evaluate performance across different object and basket placement configurations, randomized across 5 object positions and 3 basket positions, with each combination evaluated twice, yielding a total of 30 evaluation episodes per method. For the `Peg-Insertion` task, we evaluate precise insertion from a fixed configuration, averaging the success rate over 10 independent evaluation episodes per method with the same reset position. Further training and evaluation details are provided in Section A.4.

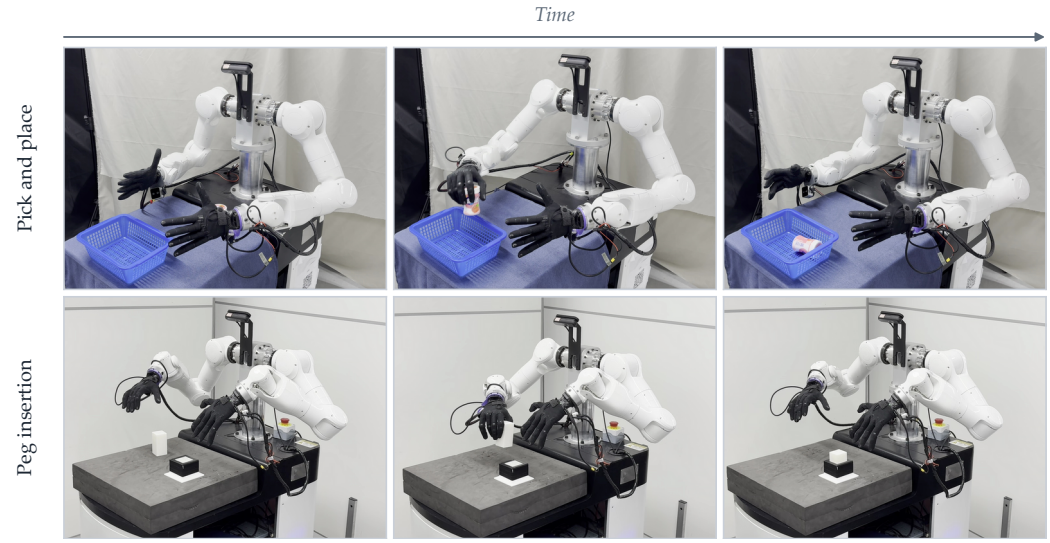


Figure 6: Real-world Policy Rollouts of DS-GRPO. Rollouts of policies adapted with DS-GRPO using human intervention data for two real-world tasks: *Pick and place* (top), which transfers an object into a basket, and *Peg insertion* (bottom), which requires precisely aligning and inserting a peg into a peg base.

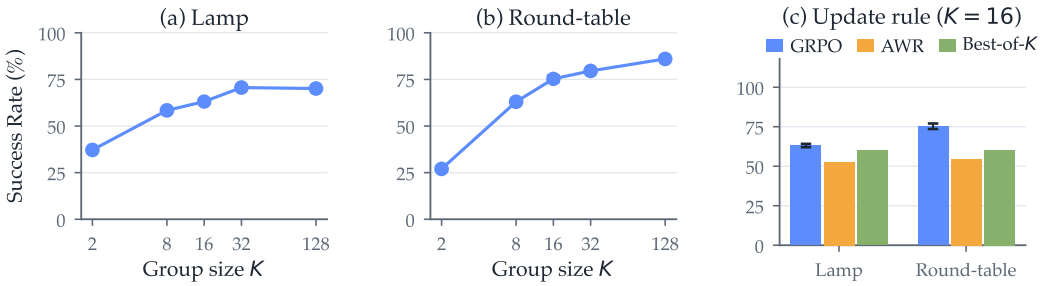


Figure 7: Ablations of DS-GRPO on FurnitureBench. *(a, b)* Effect of group size $K \in \{2, 8, 16, 32, 128\}$ on success rate. *(c)* Comparison of optimization objectives at $K = 16$: DS-GRPO uses signed group-relative advantages, whereas AWR and best-of-K use non-negative weights. All three methods use the same number of sampled candidates per state, isolating the effect of different optimization objectives.

As shown in Table 4, DS-GRPO achieves the highest observed success rate on both tasks. It increases the base policy's pick-and-place success rate by a factor of approximately $1.5\times$, while boosting the success rate on peg insertion by a factor of $4\times$. Furthermore, it exceeds DSBC by 10 percentage points on both tasks. Taken together, these results support the effectiveness of DS-GRPO for adapting real-world policies from a small set of human corrections. Representative rollouts of the adapted DS-GRPO policies are shown in Fig. 6.

Table 4: Real-world Task Success. Success rate (%): 30 pick-and-place and 10 fixed-placement insertion trials per method.

Task	Base	DSBC	DS-GRPO
Pick and place	36.7	43.3	**53.3**
Peg insertion	20.0	70.0	**80.0**

4.6 Analysis and Ablations

Group size K. To understand the impact of group size, we ablate the group size K on two tasks in FurnitureBench: `lamp` and `round_table`. Specifically, we sweep $K \in \{2, 8, 32, 128\}$ while keeping the other hyperparameters the same, and compare against the reported results in Fig. 5, obtained at $K = 16$. As shown in Fig. 7, with larger group sizes, the success rate of the resulting policies generally improves, although at the cost of more forward passes. This is well within expectations, since more samples can result in better within-group baseline estimation and better exploration. Full transition versus success rate curves are provided in Section B.1.

Optimization Objective: versus Best-of-K and AWR. We next ablate whether the gains of DS-GRPO come solely from larger sample counts, rather than the group-relative policy gradient optimization objective. Here we compare against two alternative objectives: *best-of-K*, which assigns all weight to the highest-scoring candidate sampled, and *advantage-weighted regression (AWR)* (Peng et al., 2019), which assigns non-negative weights proportional to exponentiated advantages. Importantly, DS-GRPO uses signed advantages to reinforce above-average candidates and *explicitly penalize below-average ones*, whereas the two alternatives use only non-negative weights. As shown in Fig. 7, DS-GRPO outperforms both alternatives on both tasks under matched sample counts, supporting the effectiveness of the objective. Further details are provided in Section B.1.

5 Related Work

Adaptation of Generative Control Policies. Given a pretrained generative policy, adapting the policy to new tasks or domains involves two design axes: problem setting and modulation space. Regarding problem setting, there exist three prevailing setups. *Offline Behavior Cloning* adapts from a fixed set of expert demonstrations (Torabi et al., 2018) offline; *Online DAgger* (Ross et al., 2011) queries an expert online at the states the learner actually visits; *Online RL* is also learning online, but admits only a reward function and no expert policy (Ren et al., 2025; Xu et al., 2026a).

On the second axis, adaptation of a generative control policy largely falls into three categories: full fine-tuning, residual or noise-space steering. Full fine-tuning is the most expressive as well as most compute-heavy option, though backpropagation through the diffusion chain makes reinforcement learning relatively challenging (Ren et al., 2025; Patil et al., 2026b). Residual policies learn a residual correction of the base policy's output while keeping the base policy frozen (Ankile et al., 2025a;b; Dong et al., 2026; Xu et al., 2026b; Sun & Song, 2026), being relatively compute-efficient but unconstrained by the base policy's support and blind to its multimodal structure. Noise-space steering acts on the input noise of the base policy (Wagenmaker et al., 2025; Su et al., 2026), reweighting the base policy distribution and thus remaining support-constrained (Yu et al., 2026b).

Policy Optimization with Verifiable Reward. RLVR (Lambert et al., 2024) optimizes a policy against a computed, verifiable reward, usually instantiated as updating the policy based on critic-free group-relative policy gradients (Shao et al., 2024; Guo et al., 2025). The objective and its popular instantiation, GRPO, have since moved from language models to diffusion and flow generators (Xue et al., 2025; Liu et al., 2026) and to the post-training of VLA models (Li et al., 2026; Sharma et al., 2026). These methods update all parameters of the generative model itself. In contrast, DS-GRPO updates only a lightweight noise-space steering policy, using demonstration agreement in action space as the reward, without backpropagating through the frozen generator.

6 Conclusion

We present DS-GRPO, a diffusion-steering method that combines learning from demonstrations with direct verification in action space. By treating demonstrations as action-space verifiers, DS-GRPO evaluates actions decoded from sampled noises and updates a lightweight steering policy through group-relative policy gradients. This directly ties steering policy updates to demonstration agreement in action space, avoiding noise-space regression without requiring a critic, sampler inversion, or backpropagation through the frozen generator. Experiments across offline behavior cloning, goal-conditioned behavior cloning, and online DAgger demonstrate the effectiveness of our approach. Moreover, DS-GRPO successfully transfers to improved performance on real-world pick-and-place and peg-insertion tasks. Taken together, these results show that action-space verification can provide an effective learning signal for adapting frozen generative policies from demonstrations, providing a concrete step towards efficient yet effective robot policy adaptation.

Limitations. There are several limitations of DS-GRPO. First, like all other diffusion-steering methods, DS-GRPO can only re-weight behaviors the frozen policy already exhibits, which bounds performance due to the limited support of the base policy. Second, the training cost of DS-GRPO scales approximately linearly with the group size K through forward decodes of the base policy, making it more expensive than DSBC or FlowDAgger in terms of FLOPs per optimization step.

AI USE STATEMENT

We use large language models to assist with code development and writing refinement, including improving sentence clarity, correcting typos, and checking for grammatical mistakes. We did not use generative AI tools to develop the research ideas or methodology, to design or interpret the experiments, or to outline the paper. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This study does not involve human subjects. All simulation datasets, base models, and benchmarks are obtained through publicly available channels and were used in compliance with their licenses and terms. Real-robot experiments were carried out in a controlled laboratory environment under human supervision, following standard safety protocols. The authors declare no conflicts of interest.

REPRODUCIBILITY STATEMENT

We have taken concrete steps to ensure that our results are reproducible. We have included detailed descriptions of our method in Section A.1. Training hyperparameters and implementation details are provided in Sections A.2 and A.3. We also provide detailed descriptions of how our real-world experiments are conducted in Section A.4. We will release the full source code and trained model checkpoints upon publication to ensure full reproducibility and facilitate future extensions.

REFERENCES

Lars Ankile, Zhenyu Jiang, Rocky Duan, Guanya Shi, Pieter Abbeel, and Anusha Nagabandi. Residual off-policy rl for finetuning behavior cloning policies. *arXiv preprint arXiv:2509.19301*, 2025a.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement-residual rl for precise assembly. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 01–08. IEEE, 2025b.

Jose Barreiros, Andrew Beaulieu, Aditya Bhat, et al. A careful examination of large behavior models for multitask dexterous manipulation. *Science Robotics*, 11(113):eaea6201, 2026.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *International conference on machine learning*, pp. 1042–1051. PMLR, 2019.

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

StarVLA Community. Starvla: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.

Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. Expo: Stable reinforcement learning with expressive policies. In *International Conference on Learning Representations*, volume 2026, pp. 131379–131394, 2026.

Simon S Du, Sham M Kakade, Ruosong Wang, and Lin F Yang. Is a good representation sufficient for sample efficient reinforcement learning? *arXiv preprint arXiv:1910.03016*, 2019.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Minho Heo, Youngwoon Lee, Doohyun Lee, and Joseph J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. In *Robotics: Science and Systems*, 2023.

Physical Intelligence. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv: 2504.16054*, 2025.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Yang Zhaohui, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. Simplevla-rl: Scaling vla training via reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 125846–125869, 2026.

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

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *Advances in neural information processing systems*, 38:40783–40818, 2026.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Junjie Lu, Xinyao Qin, Yuhua Jiang, Kaixin Wang, Chuheng Zhang, Bin Liang, Jun Yang, Min Xu, and Li Zhao. Unified noise steering for efficient human-guided vla adaptation. *arXiv preprint arXiv:2605.10821*, 2026.

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 *Conference on Robot Learning (CoRL)*, 2021.

Michael Murray, Daphne Chen, Simran Bagaria, Dean Fortier, Tess Hellebrekers, Galen Mullins, Harshavardhan Gajarla, Oier Mees, Maya Cakmak, and Andrey Kolobov. Flowdagger: Human-in-the-loop adaptation of generative robot policies in latent space. *arXiv preprint arXiv:2607.08877*, 2026.

Chaoyi Pan, Giridharan Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, et al. Much ado about noising: Dispelling the myths of generative robotic control. In *International Conference on Learning Representations*, volume 2026, pp. 90575–90614, 2026.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations (ICLR)*, 2025.

Omkar Patil, Ondrej Biza, Thomas Weng, Karl Schmeckpeper, Wil Thomason, Xiaohan Zhang, Kausik Sivakumar, Robin Walters, Nakul Gopalan, Sebastian Castro, et al. You've got a golden ticket: Improving generative robot policies with a single noise vector. *arXiv preprint arXiv:2603.15757*, 2026a.

Sarvesh Patil, Mitsuhiko Nakamoto, Manan Agarwal, Shashwat Saxena, Jesse Zhang, Giri Anantharaman, Cleah Winston, Chaoyi Pan, Douglas Chen, Nai-Chieh Huang, et al. Ogpo: Sample efficient full-finetuning of generative control policies. *arXiv preprint arXiv:2605.03065*, 2026b.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Allen Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, volume 2025, pp. 77288–77329, 2025.

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

Junwon Seo, Sushant Veer, Ran Tian, Wenhao Ding, Apoorva Sharma, Karen Leung, Edward Schmerling, Marco Pavone, and Andrea Bajcsy. Stressdream: Steering video world models for robust policy evaluation and improvement. *arXiv preprint arXiv:2606.00267*, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Ansh Kumar Sharma, Yixiang Sun, Ninghao Lu, Yunzhe Zhang, Jiarao Liu, and Sherry Yang. World-gymnast: Training robots with reinforcement learning in a world model. *arXiv preprint arXiv:2602.02454*, 2026.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Entong Su, Tyler Westenbroek, Anusha Nagabandi, and Abhishek Gupta. Rfs: Reinforcement learning with residual flow steering for dexterous manipulation. In *The Fourteenth International Conference on Learning Representations*, 2026.

Zhanyi Sun and Shuran Song. From prior to pro: Efficient skill mastery via distribution contractive rl finetuning. *arXiv preprint arXiv:2603.10263*, 2026.

Andy Tang, William Chen, Andrew Wagenmaker, Chelsea Finn, and Sergey Levine. Improving robotic generalist policies via flow reversal steering. *arXiv preprint arXiv:2606.13675*, 2026.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

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. *arXiv preprint arXiv:2506.15799*, 2025.

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

Xiaomeng Xu, Yifan Hou, Zeyi Liu, and Shuran Song. Compliant residual dagger: Improving real-world contact-rich manipulation with human corrections. *Advances in Neural Information Processing Systems*, 38:139559–139581, 2026b.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, et al. Dancegrpo: Unleashing grpo on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026a.

Raymond Yu, William Huey, Mustafa Mukadam, Anusha Nagabandi, and Abhishek Gupta. Support-constrained rl enables real-world policy improvement without real-world experience. *arXiv preprint arXiv:2606.27475*, 2026b.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

A Experimental and Implementation Details

A.1 DS-GRPO Implementation

In this subsection, we present the pseudocode and implementation details of DS-GRPO for simulation tasks. In Algorithm 1, we summarize the algorithm procedure.

Algorithm 1 DS-GRPO

function $\pi_{\text{base}}(\mathbf{s}, \mathbf{w})$ ▷ frozen sampler, forward only
  return FrozenDecode$(\mathbf{s}, \mathbf{w})$ ▷ backend's released sampler

initialize steering policy $\pi_\phi^{\mathcal{W}}$ with $\mu_\phi \approx 0$, $\boldsymbol{\sigma}_\phi = \sigma_0 \mathbf{1}$
while not converged **do**
  $\phi_{\text{old}} \leftarrow \phi$; sample states $\mathbf{s} \sim \rho$
  ▷ Propose and decode K candidates per state
  $\mathbf{w}_1, \ldots, \mathbf{w}_K \sim \pi_{\phi_{\text{old}}}^{\mathcal{W}}(\cdot \mid \mathbf{s})$
  $\mathbf{a}_i \leftarrow \pi_{\text{base}}(\mathbf{s}, \mathbf{w}_i)$ ▷ no gradient; K forward decodes
  ▷ Verify in action space
  $r_i \leftarrow r(\mathbf{a}_i; \mathbf{s})$ ▷ the demonstration is the verifier, Eq. (3.1)
  $\hat{A}_i \leftarrow (r_i - \bar{r})/(\sigma_r + \delta)$ ▷ group mean as baseline, Eq. (3.4)
  ▷ Update the steering policy only
  Update ϕ on the clipped surrogate Eq. (3.5), with $\boldsymbol{\sigma}_\phi$ floored componentwise at $\sigma_{\min}$ in the forward pass
return $\pi_\phi^{\mathcal{W}}$ ▷ deploy: $\mathbf{w} \sim \pi_\phi^{\mathcal{W}}(\cdot \mid \mathbf{s})$, one decode $\mathbf{a} = \pi_{\text{base}}(\mathbf{s}, \mathbf{w})$

Steering policy. The network predicts the mean of a diagonal Gaussian. Its standard deviations are learned separately, initialized at 0.5 and clipped to be at least 0.25 during training and deployment. Network architectures are listed in Section A.3.

Candidate Scoring. Each sampled noise is decoded into an action chunk and scored against the expert chunk using Eq. (3.1). For H action steps and A action dimensions, we set $c = 1/(HA)$ on LIBERO and OGBench and $c = 1/H$ on robomimic and FurnitureBench. LIBERO compares actions after conversion to environment coordinates. We standardize scores within each group using Eq. (3.4) with $\delta = 10^{-6}$. Groups with reward standard deviation below 10^{-4} contribute no gradient.

Policy Update. We use Adam as DS-GRPO's optimizer. After a batch of sampled noises is passed through the frozen sampler, we update the steering policy for two steps on LIBERO and one on the other benchmarks. Noises, advantages, and old log-probabilities stay fixed until the next batch. We use PPO clip 0.2, clamp log-ratios to $[-20, 20]$, and clip gradient norms at 1.0. We use no reference policy or KL penalty. Batch sizes and learning rates are detailed in Table 6.

A.2 Baseline and Ablation Implementations

DSBC. DSBC (Tang et al., 2026) first inverts expert actions into noise labels, then fits the steering policy to these labels by Gaussian negative log-likelihood. Labels are computed and stored before training, using the end-point velocity to reverse each sampler step. The encoder and mean network match DS-GRPO's; DSBC also predicts the standard deviation from the state. On LIBERO, labels are extracted every 10th frame, followed by $12{,}500$ Adam updates with batch size 128 and learning rate 10^{-4}. OGBench uses the training settings in Table 6.

FlowDAgger. FlowDAgger (Murray et al., 2026) finds noise labels with 5 fixed-point iterations per sampler step, then fits a deterministic steering policy by MSE. For the clipped DPPO sampler, we start inversion from the policy's current noise prediction and refine the result with 20 Adam steps. Refinement uses learning rate 0.01, gradient-norm clip 1.0, and an ℓ_2 penalty of weight 0.01 that keeps the noise close to its starting point. We discard labels whose decoded actions have MSE above 10^{-3} against the expert. The policy is a three-layer MLP of width $2{,}048$ with output $3\tanh(\cdot)$. After 250 collection steps, it takes 500 updates with batch size 256 and learning rate 3×10^{-4}. Each collection step executes one action chunk in every parallel environment.

DSRL. DSRL (Wagenmaker et al., 2025) learns from environment success rewards without teacher labels. We use DSRL-NA on robomimic and DSRL-SAC on FurnitureBench. For DSRL-SAC, action magnitude is 1.0, learning rate is 3×10^{-4}, and the update-to-data ratio is 5 per environment step. Training starts with 1,250 vectorized warm-up steps.

Best-of-K and AWR. Both alternatives sample and score candidates as DS-GRPO does, but change their training weights. Best-of-K trains only on the highest-scoring candidate; AWR gives every candidate a positive weight, with larger weights for higher scores. In place of $\hat{A}_k$ in Eq. (3.5), we use

$$\omega_k^{\text{Best}} = K \cdot \mathbf{1}[k = \arg\max_j \hat{A}_j], \qquad \omega_k^{\text{AWR}} = K \frac{\exp(\hat{A}_k/\tau)}{\sum_j \exp(\hat{A}_j/\tau)}, \qquad \tau = 1. \tag{A.1}$$

Ablation Protocol. We use FurnitureBench `lamp` and `round_table` for ablation. The group-size ablation varies $K \in \{2, 8, 32, 128\}$; the update-rule ablation fixes $K = 16$ and changes only the optimization objective. Other training settings and hyperparameters are unchanged.

A.3 Simulation Experiment Details

Base policies and datasets. We list configurations for simulation base policies, adaptation data and samplers in Table 5. For Online DAgger tasks, teachers are DPPO-fine-tuned policies: FurnitureBench uses 250 iterations for `one_leg` and 350 for `lamp` and `round_table`.

Table 5: Simulation base policies, adaptation data, and samplers.

Benchmark	Base policy and adaptation data	Sampler
LIBERO-90	Released $\pi_{0.5}$ (Intelligence, 2025), StarVLA (Community, 2026), and Fast-WAM (Yuan et al., 2026), trained on the other LIBERO suites; adapted on successful demonstrations of each LIBERO-90 task.	$\pi_{0.5}$: Euler, 10 steps StarVLA: Euler, 4 steps Fast-WAM: Euler, 20 steps
OGBench	Flow prior trained on the environment's `play` dataset without goals; steering uses the same data with goals added.	Euler, 10 steps
Robomimic	Released DPPO BC checkpoints; labels from the corresponding DPPO fine-tuned teachers.	DDIM, 8 steps
FurnitureBench	Released DPPO BC checkpoints; labels from the corresponding DPPO fine-tuned teachers.	DDIM, 100 steps

Steering architectures. LIBERO uses separate encoders for the base and wrist cameras, each producing 50 features through a Linear–LayerNorm–$\tanh$ layer. A three-layer, width-128 MLP outputs the mean through $5\tanh(\cdot)$. For $\pi_{0.5}$ and StarVLA, one 7-dimensional noise vector is repeated across the action chunk; Fast-WAM uses eight such vectors for eight temporal blocks. Unsteered noise dimensions remain standard Gaussian. OGBench uses a three-layer, width-512 GELU MLP on the observation and goal, with output $3\tanh(\cdot)$; the visual task first encodes each image into 128 features. Robomimic and FurnitureBench use a three-layer, width-2,048 MLP with $\tanh$ and layer normalization, mapping the state to a mean for every noise dimension.

Training Hyperparameters. Table 6 lists hyperparameters used for DS-GRPO training. Each batch contains the listed number of states and K candidates per state. For online DAgger, one collection step executes an action chunk in every parallel environment; each collection phase is followed by several policy updates. Baseline settings are in Section A.2.

Evaluation. Table 7 lists the evaluation episodes, seeds, and checkpoints used for the main results. Base policies are evaluated once per task; adapted policies are evaluated for each training seed.

Table 6: DS-GRPO simulation training configurations. Shared numerical settings are given in Section A.1.

	LIBERO-90	OGBench	robomimic	FurnitureBench
Group size K	8	32	16	16
States per batch	16	512	64	64
Adam steps per batch	2	1	1	1
Batches per collection	—	—	4	64/91/91
Chunk-steps per collection	—	—	75	88/125/125
Learning rate	3×10^{-5}	3×10^{-4}	10^{-5}	3×10^{-5}
Training budget	2,000 iterations	100k updates	0.2/1.2/2.5M env steps	10M env steps
Training seeds	0, 1	1, 2, 3	42, 43, 44	42, 43, 44

Task order: robomimic `lift`/`can`/`square`; FurnitureBench `one_leg`/`lamp`/`round_table`.

Table 7: Simulation evaluation protocols for the main results.

Benchmark	Trials and eval seed	Checkpoint selection and aggregation
LIBERO-90	50 per task; seed 7	Final policy; average success over 89 tasks, then over two training seeds. Fixed benchmark initial states.
OGBench	50 per goal; seed 7	Five evaluation goals; average over the 80k, 90k, and 100k checkpoints and three training seeds.
robomimic	200; seed 10007	Final evaluation within each task's budget, over three training seeds; held-out initial states.
FurnitureBench	512; seed 10007	10M-step results, averaged over three training seeds.

A.4 Details of Real-World Experiments

Platform and tasks. We use a bimanual robot with dexterous Wuji hands for real-world experiments. We evaluate on two tasks: *Pick-and-Place* and *Peg-Insertion*. `Pick-and-Place` transfers a gum bottle into a basket with variable placement positions. For `Peg-Insertion`, the goal of the task is to transport the peg, initialized at a fixed position, into the peg base. The first task requires generalization across initial object positions, whereas the second demands high precision.

Base Policy and Human Intervention Data Collection. Across two tasks, we fine-tune $\pi_{0.5}$ (Intelligence, 2025) with teleoperation data to form our base policy. For `Pick-and-Place`, we collect 200 teleoperation demonstrations, and for `Peg-Insertion`, we train on 100 demonstrations. We then apply human interventions during base policy rollout to acquire adaptation data. For each task, a human intervenes when the base policy fails or is clearly about to take an erroneous action. In both tasks, we collect 40 episodes of this intervention data, which is later used for diffusion steering training. The data was recorded at 30 Hz, including visual observations of 640×480 RGB images from the overhead scene camera and two wrist cameras, 54-dimensional proprioceptive states, and 54-dimensional actions (7-DoF left arm+20-DoF left hand+7-DoF right arm+20-DoF right hand).

Training Details. The human intervention data is then used to train DSBC and DS-GRPO offline. The diffusion steering policy outputs a 54-dimensional noise vector, which is repeated across the 32-step action chunking horizon and denoised using 10 denoising steps. Training is done using a global batch size of 8 for 4 epochs. For DS-GRPO, we used 8 samples per observation and the Adam optimizer with a learning rate of 3×10^{-5}. Training was performed on four NVIDIA A800 GPUs and took approximately 20 minutes to complete. Unlike DS-GRPO, DSBC did not sample action groups or apply PPO updates; all other training settings were held fixed.

Evaluation protocol. We measure the steered policy's performance through success rate during policy rollout. Before every rollout, the robot was reset to the same initial joint configuration, which is the element-wise mean of the first-frame robot states across the training dataset. During evaluation, the policy executes the full action chunk horizon (32 steps) before running policy inference again to replan the next action chunk.

For *Pick-and-Place*, we randomly initialize the scene at five object locations and three basket locations within the training distribution, as illustrated in Fig. 8. Each of the 15 object–basket con-

figurations was evaluated twice, yielding 30 trials per method. A pick-and-place trial is considered successful if and only if the object is placed inside the target basket and released by the gripper.

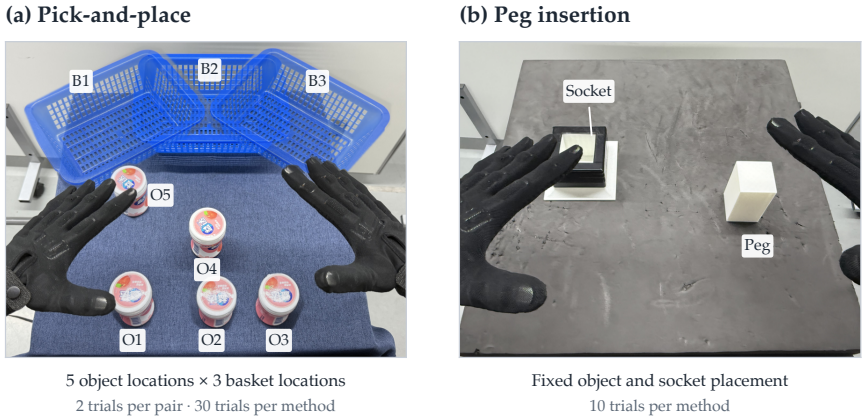


Figure 8: Real-world evaluation setups. Left: Overlay of five object locations (O1–O5) and three basket locations (B1–B3) for Pick-and-Place, with two trials per combination and 30 trials per method. Right: the fixed Peg-Insertion configuration, evaluated over 10 trials per method.

For *Peg-Insertion*, we reset from the same configuration across all trials. We evaluate each method over 10 episodes. A peg-insertion trial is successful only if the peg is fully inserted into the peg base, leaving no room in between the peg and the peg base.

B Additional Experimental Results

B.1 Ablation Learning Curves

We report success rates against environment steps for the experiments in Fig. 7 in Figure 9. For the ablation on group size K, increasing group size K generally improves learning efficiency and final performance. For the ablation on update rules, using DS-GRPO's signed-advantage optimization objective leads to better performance over most of training and at the final budget compared to the two objectives using only non-negative weights: Best-of-K and AWR.

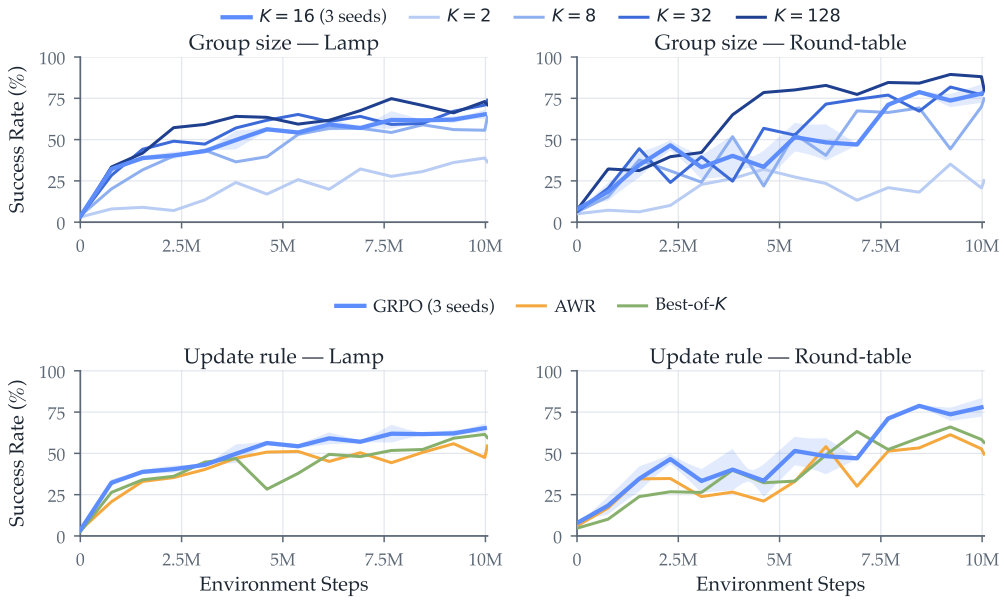


Figure 9: Success Rate against Environment Steps for DS-GRPO Ablation Experiments. *Top:* Ablation on group size K; *Bottom:* Ablation on the update rule at $K = 16$.

B.2 Real-World Rollouts

We provide additional real-world rollouts on the Wuji-hand platform in Figs. 10 and 11. Each row shows four chronologically ordered frames from one episode, with non-uniform time intervals. These selected qualitative examples complement the aggregate success rates in Table 4; training and evaluation details are provided in Section A.4.

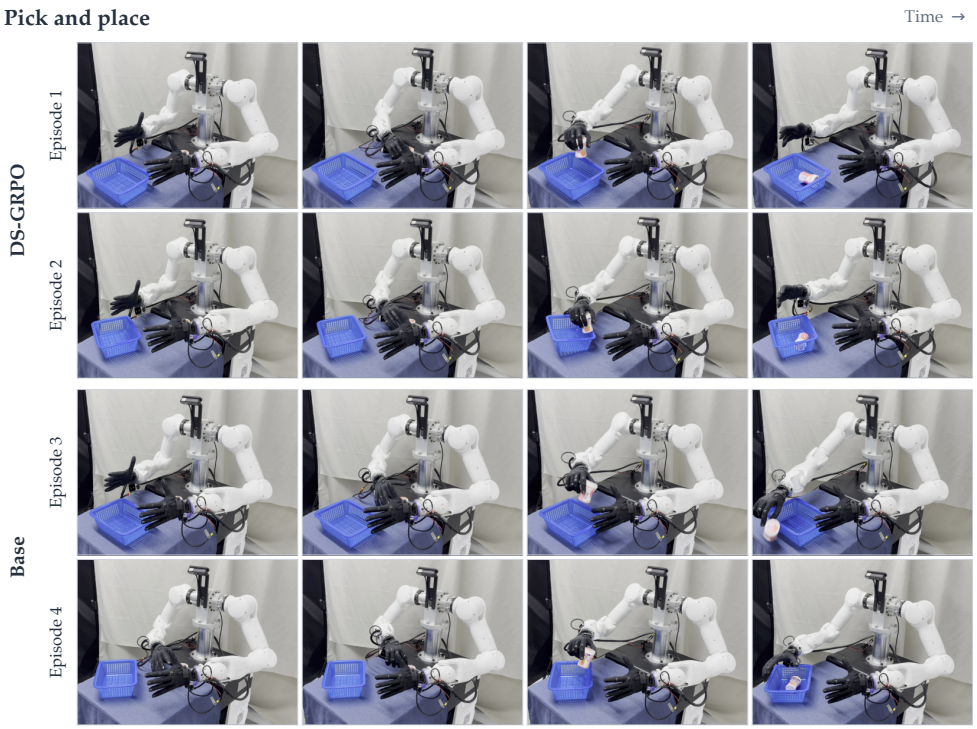


Figure 10: Additional Pick-and-Place Rollouts. Episodes 1–2 show successful transfers by DS-GRPO. Episodes 3–4 show a failed transfer and a successful transfer by the base policy, respectively.

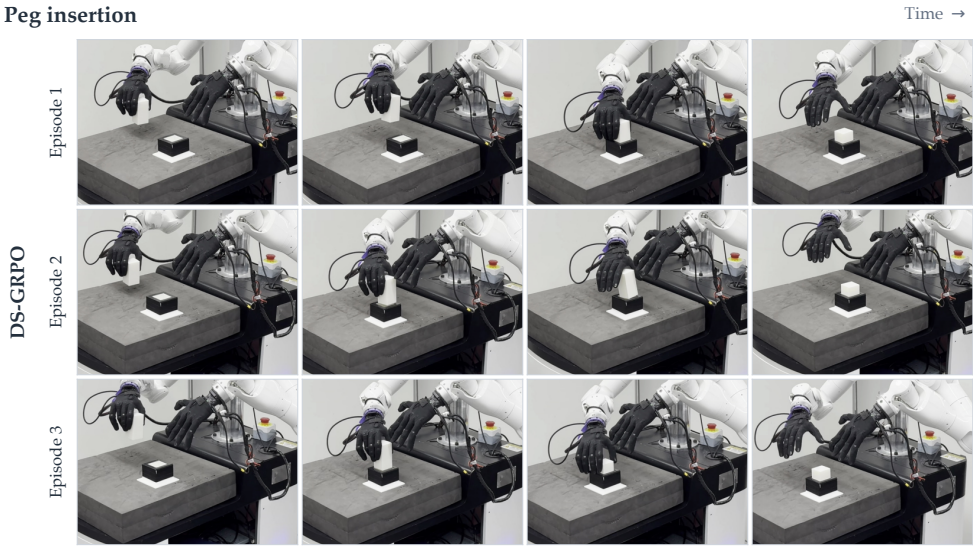


Figure 11: Additional Peg-Insertion Rollouts. Three rollout episodes of a policy steered via DS-GRPO, all of which ended successfully with the peg inside the peg base.