PROXIMAL BEHAVIOR RESHAPING

Anonymous authors
Paper under double-blind review

ABSTRACT

Policy optimization in continuous control commonly uses the same parametric distribution both as the object of optimization and as the behavior distribution for environment interaction. This coupling can be restrictive when effective exploration requires richer behavioral structure, while directly using learned action values for policy improvement can make learning sensitive to critic error. We introduce **Proximal Behavior Reshaping (PBR)**, which separates the distribution directly optimized by PPO from the distribution executed for interaction. PBR retains a Gaussian policy as the optimization anchor and learns a residual flow that reshapes its sampled actions into a richer composed policy. Learned action values guide local, policy supported reshaping without directly determining the Gaussian policy update, and are trained using the current rollout together with a short buffer of recent policy data. On a goal reaching task with wall avoidance, PBR produces coherent obstacle bypassing behavior while making effective use of locally useful but imperfect action value estimates. Across six high dimensional Isaac Lab tasks spanning locomotion and manipulation, PBR exceeds PPO across the benchmark while providing strong and consistent performance relative to more direct Q guided and flow based alternatives.

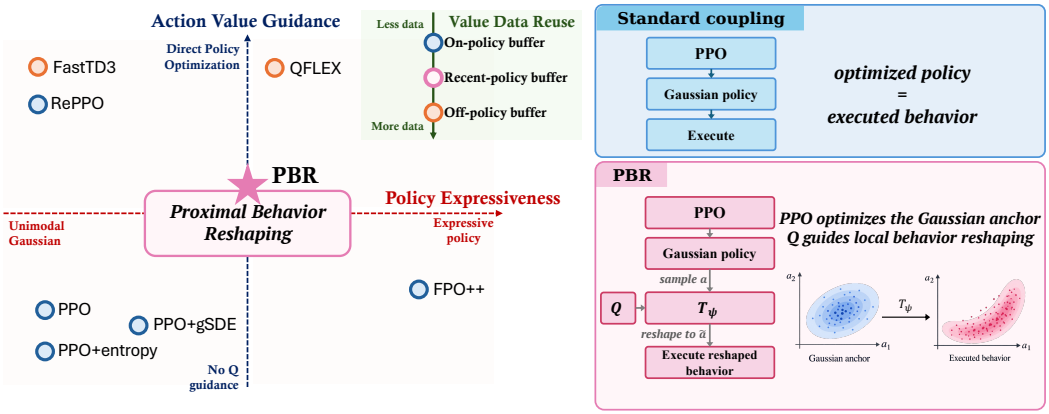


Figure 1: Overview of the continuous RL algorithm design space and PBR architecture. **Left:** Methods differ in policy expressiveness, action value guidance, and value data reuse, with PBR taking the center position. **Right:** Standard policy optimization directly optimizes the same distribution that is executed for interaction. PBR keeps PPO optimization on the Gaussian anchor while learned action values provide proximal guidance for local behavior reshaping, producing a richer executed policy.

1 INTRODUCTION

Continuous control policy optimization often uses the same parametric distribution both as the object directly optimized and as the distribution executed for environment interaction. This coupling can be limiting, since the policy should remain easy and stable to optimize, while the resulting behavior must be expressive enough to discover useful interaction strategies. Methods such as PPO Schulman et al. (2017) commonly use a Gaussian policy for both roles, providing a simple optimization target but restricting interaction to the structure represented by that distribution. Increasing exploration or using a more expressive policy can produce richer behavior, but often at the cost of a more difficult policy optimization problem. A complementary direction is to use learned action values to provide

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.

structure. They can direct exploration toward promising actions and support policy improvement beyond what is available from policy gradient information alone. However, giving a learned Q function greater influence over the policy also makes optimization more dependent on the quality of its value estimates. In addition, methods differ in how much past experience they reuse for value learning, ranging from current on-policy data to off-policy replay. These three choices define a broader design space spanning *policy expressiveness*, *action value guidance*, and *value estimation data reuse*, where moving farther along any one dimension is not uniformly preferable.

This view raises a simple question: *must policy optimization, interaction behavior, and value guidance be tied to the same mechanism?* We argue that separating these roles provides useful design freedom. We introduce **Proximal Behavior Reshaping (PBR)**, which keeps PPO optimization on a Gaussian anchor while reshaping its sampled actions through a learned transformation, with learned action values used proximally to guide local, policy supported modifications. The resulting behavior distribution can acquire additional structure without requiring the Gaussian policy itself to represent that structure. We call the value guidance *proximal* because the critic shapes local behavior without directly determining the Gaussian policy update, leaving PPO responsible for how the anchor policy evolves. Finally, the auxiliary action value function is learned from the current rollout together with a short window of recent policy data, providing limited data reuse without making the PPO update itself off policy.

We first study these choices in Wall2D, a goal reaching task that requires a coherent multi-step detour around an obstacle. Comparisons across methods with different levels of policy expressiveness and action value guidance show that neither richer exploration nor learned Q guidance alone guarantees the required behavior. PBR instead reshapes the Gaussian anchor into a structured behavior distribution that reliably completes the bypass. Our analysis then isolates the roles of its two components: behavior reshaping provides structure that the Gaussian anchor need not represent directly, while proximal guidance uses locally informative action values without requiring uniformly accurate Q estimates to drive the policy update. We then evaluate PBR on high-dimensional locomotion and manipulation problems. Across six Isaac Lab tasks, PBR remains consistently strong relative to the baselines, with additional MuJoCo Playground experiments showing similar gains. These results demonstrate that the separation exposed in Wall2D remains practical at scale, while confirming that the relative benefits of different design choices remain task dependent.

Overall, our contributions are: First, we organize continuous control methods along three design dimensions: policy expressiveness, action value guidance, and value estimation data reuse, highlighting the tradeoffs between different choices along each dimension (Section 3). Second, We introduce **Proximal Behavior Reshaping (PBR)**, which keeps PPO optimization on a Gaussian anchor while learning a richer behavior distribution through local, value guided action reshaping (Section 5). Third, through controlled analysis and high dimensional locomotion and manipulation benchmarks, we show why behavior reshaping and proximal value guidance are complementary and demonstrate strong, consistent performance across diverse continuous control tasks. (Section 7).

2 Related Work

Exploration and expressive policies. Gaussian policies remain a common choice in continuous control because they support simple and stable policy optimization, but their exploration is limited by the structure of the underlying distribution. Structured exploration methods such as parameter space noise Plappert et al. (2017) and gSDE Raffin & Stulp (2020) introduce temporally coherent or state dependent exploration while retaining a tractable policy class. Other approaches instead increase the expressiveness of the policy itself. Recent diffusion and flow policies can represent substantially richer action distributions and have been adapted to reinforcement learning through methods such as DPPO Ren et al. (2025), ReinFlow Zhang et al. (2026), and FPO++ Yi et al. (2026), and PolicyFlow Yang et al. (2026). These methods improve the structure available for exploration by optimizing a more expressive policy class. PBR takes a different route. It keeps PPO optimization on a Gaussian anchor and places additional expressiveness in a learned transformation of the behavior.

Action value guided policy improvement. Action value functions provide a direct signal for improving continuous actions and form the basis of off-policy actor critic methods such as DDPG Lillicrap et al. (2015), TD3 Fujimoto et al. (2018), and SAC Haarnoja et al. (2018). Other methods combine learned values with policy optimization through control variates, constrained policy improvement, or weighted regression Gu et al. (2016); Abdolmaleki et al. (2018); Peng et al. (2019). Recent

methods extend this idea to large scale continuous control. FastTD3 Seo et al. (2025) combines off-policy value learning with actor optimization and highly parallel simulation, while RePPO Voelcker et al. (2026) learns an action value function from on-policy data and uses pathwise Q gradients to update the policy. QFLEX Wei et al. (2026) combines these directions in an off policy setting, using learned Q values both to improve a Gaussian source policy and to construct a Q guided flow for interaction. PBR also uses learned action values to direct behavior, but restricts their role to local behavior reshaping while leaving the Gaussian anchor under PPO optimization.

Separating exploration and policy learning. Prior work has also decoupled the behavior used for exploration from the policy being learned. ADAC Liu et al. (2020) uses separate target and expressive behavior policies within an off policy actor critic framework, while DEEP Whitney et al. (2021) learns distinct exploration and exploitation policies for sample efficient continuous control. These approaches use a separate exploration policy to collect data for an off policy target policy. PBR instead retains PPO optimization on a Gaussian anchor while learning a transformation whose composition with that anchor is itself the policy used for interaction and evaluation.

3 Design Space of Exploration and Value Guidance

Rather than viewing the methods discussed above as separate algorithmic families, we organize them along three largely independent design spectra. **Policy expressiveness** ranges from unimodal Gaussian policies, through structured exploration and Gaussian policies with learned transformations, to directly expressive policies. **Action-value guidance** ranges from no action-value guidance, to behavior-level guidance that uses learned Q values to reshape sampled actions, to direct policy optimization using learned Q values. Finally, **value-estimation data reuse** ranges from current on-policy data, through a short recent-policy buffer, to off-policy replay.

Representative methods occupy different points along these spectra. PPO and PPO+entropy use unimodal Gaussian policies, while gSDE introduces structured exploration and FPO++ directly parameterizes an expressive flow policy. QFLEX instead augments a Gaussian source policy with a learned transformation. Along the value guidance spectrum, PPO, gSDE, and FPO++ do not use learned action values; FastTD3 and RePPO use Q for direct policy optimization; and QFLEX uses Q for both action transport and Gaussian actor optimization. These methods also span the on-policy and off-policy ends of the data reuse spectrum, as indicated by color in the left panel of Figure 1.

These dimensions represent tradeoffs rather than monotonic improvements. Greater policy expressiveness can support richer behavior, but increases the complexity of the policy or behavior model being optimized. Stronger action-value guidance can provide more directed improvement, but gives the learned critic greater influence over learning. Greater data reuse provides more experience for value estimation, but increasingly historical data may be less relevant to the current policy. Algorithm design therefore requires balancing behavioral expressiveness, dependence on learned value estimates, and the amount of historical experience used for value learning.

PBR deliberately occupies an intermediate point along all three spectra. It augments a Gaussian policy with a learned behavior transformation, uses Q_ϕ to guide interaction without directly determining the Gaussian policy update, and reuses a short window of recent-policy data for value learning. We next use a controlled navigation task to examine how these tradeoffs appear in practice.

4 Toy Example: Goal Reaching with Wall avoidance

To make these design tradeoffs concrete, we begin with a two-dimensional goal-reaching task, Wall2D, with a central wall obstacle. The agent must navigate from a fixed start region to a goal on the opposite side of the wall, requiring a sustained detour around either boundary of the obstacle. We use a dense progress reward based on the shortest collision-free distance to the goal, together with a terminal success bonus and penalties for collision or leaving the workspace. Despite its simple geometry, the task requires the policy to discover and maintain a coherent multi-step bypass rather than greedily progressing toward the goal. We consider methods spanning two complementary dimensions of the design space under the same interaction budget:

- **Policy expressiveness and exploration.** We include standard PPO, PPO with increased entropy (5x), PPO with generalized State-Dependent Exploration (gSDE), and FPO++. These methods progressively vary how exploratory behavior is generated: PPO and PPO+entropy retain a unimodal Gaussian policy, gSDE introduces state-dependent and temporally persistent exploration, and FPO++ directly parameterizes an expressive flow policy.

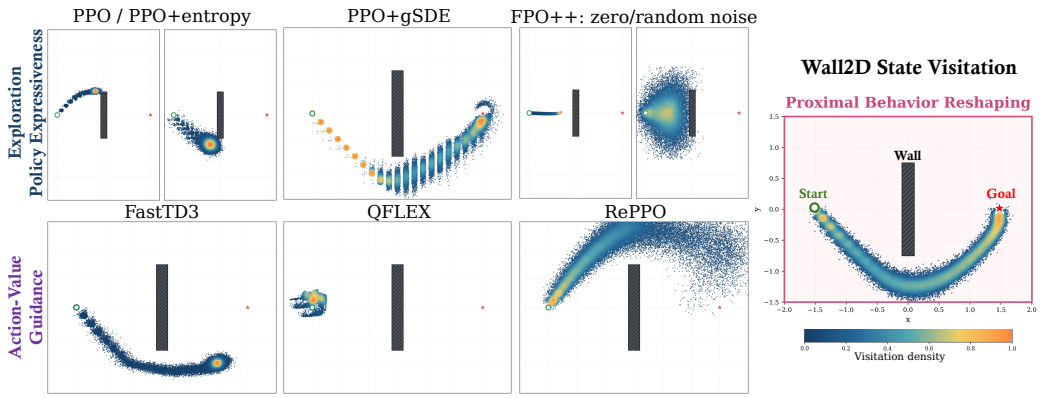


Figure 2: State-visitation density on Wall2D after 409.6k environment steps. **Top: Exploration and Policy expressiveness.** PPO and PPO+entropy fail to establish a coherent bypass; gSDE succeeds through structured exploration, while FPO++ produces richer behavior that remains confined before the wall. **Bottom: Action-value guidance.** FastTD3, Qflex, and RePPO exhibit distinct directed behaviors but do not reliably reach the goal at the matched budget. **Right:** PBR combines a Gaussian policy with value-guided behavior reshaping to produce a coherent goal-reaching trajectory.

- **Action-value guidance.** We include FastTD3, QFLEX, and RePPO, which use learned action values through different interfaces and value estimation data regimes. FastTD3 performs off-policy deterministic policy improvement using learned value functions; QFLEX combines an off-policy learned Gaussian source policy with Q-guided action transport; and RePPO learns an on-policy Q function and directly optimizes a stochastic policy through KL-constrained pathwise Q gradients. Thus, these methods also span the value estimation data-reuse spectrum introduced in Section 3, from on-policy estimation to off-policy replay.

The left panels of Figure 2 visualize the state-visitation distributions induced by the representative methods. Despite the dense reward providing continuous guidance, most methods fail to reliably reach the goal within the matched interaction budget.

The top row examines **policy expressiveness and exploration**. Standard PPO concentrates its visitation before the wall, while increasing entropy broadens exploration without establishing a coherent bypass. gSDE, in contrast, reliably reaches the goal through temporally persistent, state-dependent exploration. FPO++ further increases policy expressiveness, where without additional flow noise its behavior remains concentrated before the wall, while random flow noise produces substantially broader visitation, yet neither forms a successful detour. Together, these results suggest that neither exploration magnitude nor policy expressiveness alone is sufficient. The induced behavior must contain structure that supports the sustained multi-step bypass required by the task.

The bottom row examines **action-value guidance**. FastTD3, QFLEX, and RePPO leverage learned action values through different policy interfaces and value estimation data regimes, producing distinct directed behaviors. Nevertheless, none reliably reaches the goal at the matched budget. Thus, introducing action-value guidance, whether learned from on-policy or off-policy data and incorporated through different mechanisms, does not by itself guarantee successful policy improvement.

Together, these comparisons reveal two complementary requirements. First, successful exploration requires behavior with sufficient structure to form the multi-step detour. Second, the availability of learned action values alone is insufficient to ensure such behavior is acquired. This motivates considering behavior shaping and value guidance jointly, rather than treating either axis in isolation.

Guided by these observations, **Proximal Behavior Reshaping (PBR)** combines two components: ***behavior reshaping***, which constructs a structured behavior distribution from a Gaussian anchor, and ***proximal*** *value guidance*, which uses learned action values locally around the current policy.

5 Proximal Behavior Reshaping

Let $\pi_\theta(a \mid s) = \mathcal{N}\big(\mu_\theta(s), \mathrm{diag}(\sigma_\theta^2(s))\big)$. denote the Gaussian anchor policy optimized by PPO. Rather than executing samples from π_θ directly, PBR learns a residual transformation T_ψ that re-

shapes them using local value information. Given $a \sim \pi_\theta(\cdot \mid s),\ \tilde{a} = T_\psi(s, a)$, the transformation induces the behavior distribution

$$\tilde{\pi}_{\theta,\psi} = T_\psi \# \pi_\theta, \tag{1}$$

where $\#$ denotes the pushforward operator. PPO updates the anchor policy π_θ, while value guidance acts through T_ψ to shape the behavior used for interaction. This decoupling allows PBR to retain a Gaussian optimization anchor while expressing richer behavior through the learned transformation.

To construct this reshaped behavior distribution, we seek actions $\tilde{a}$ with higher estimated action value while remaining close to the support of the current policy. A natural choice is the KL regularized policy improvement objective $p_Q^\star(\cdot \mid s) = \arg\max_p \left[\mathbb{E}_{\tilde{a}\sim p}\left[Q_\phi(s, \tilde{a})\right] - \frac{1}{\beta}\mathrm{KL}(p(\cdot \mid s) \,\|\, \pi_\theta(\cdot \mid s))\right]$, with its solution $p_Q^\star(\tilde{a} \mid s) \propto \pi_\theta(\tilde{a} \mid s) \exp(\beta Q_\phi(s, \tilde{a}))$, where β controls the strength of value guidance. This distribution captures the tradeoff between favoring actions with higher estimated value and remaining close to the current policy. PBR further localizes this improvement around each sampled source action a, seeking a proximal target a^+ that increases estimated value without moving far from the current policy support. To formalize this local reshaping, we measure the value improvement of a target a^+ relative to its source action a as

$$\Delta Q_\phi(s, a, a^+) = Q_\phi(s, a^+) - Q_\phi(s, a). \tag{2}$$

We additionally introduce a transport cost $c(a, a^+)$ that penalizes large changes in action space. The proximal target distribution is then defined by

$$p^\star(\cdot \mid s, a) = \arg\max_p \left[\mathbb{E}_{a^+\sim p}\left[\beta_Q \Delta Q_\phi(s, a, a^+) - \beta_c c(a, a^+)\right] - \mathrm{KL}\left(p(\cdot \mid s, a) \,\|\, \pi_\theta(\cdot \mid s)\right)\right]. \tag{3}$$

The corresponding solution is

$$p^\star(a^+ \mid s, a) \propto \pi_\theta(a^+ \mid s) \exp\big(\beta_Q \Delta Q_\phi(s, a, a^+) - \beta_c c(a, a^+)\big). \tag{4}$$

This distribution favors targets that improve on the source action while staying close to the policy. Direct sampling from Eq. 4 is generally intractable, so for each state we independently draw source actions and candidate actions from the current policy, $a_i, a'_j \sim \pi_\theta(\cdot \mid s), i, j = 1, \ldots, K$. For each source action a_i, we measure the relative value improvement of candidate a'_j as

$$\Delta Q_{ij} = Q_\phi(s, a'_j) - Q_\phi(s, a_i). \tag{5}$$

We include a_i as an identity candidate and normalize ΔQ_{ij} to $\overline{\Delta Q}_{ij}$ using the candidate set standard deviation. To define the coupling between source actions and candidate targets, we measure their policy normalized transport cost as

$$c(a_i, a'_j) = \frac{1}{D_a} \left\| \frac{a'_j - a_i}{\sigma_\theta(s)} \right\|_2^2, \tag{6}$$

where D_a is the action dimension. Each source–target pair is scored by balancing relative value improvement against the transport cost,

$$S_{ij} = \beta_Q \overline{\Delta Q}_{ij} - \beta_c\, c(a_i, a'_j). \tag{7}$$

where β_Q controls the strength of value guidance and β_c controls the penalty on transport distance. We augment the candidate set with the identity $a'_0 = a_i$, for which $S_{i0} = 0$, and sample according to

$$p_{ij} = \frac{\exp(S_{ij})}{\sum_{k=0}^{K} \exp(S_{ik})}, \qquad j = 0, \ldots, K. \tag{8}$$

The sampled targets define desirable local modifications of the Gaussian policy, but constructing them independently at every interaction would not provide a reusable behavior model. PBR therefore amortizes these source–target relationships into a shared transformation T_ψ. We build T_ψ through a residual flow over the action space, parameterized by a time dependent velocity field G_ψ. Given a sampled pair (a_i, a_i^+), we train G_ψ to transport the source action toward its proximal target. For $t \sim \mathcal{U}[0, 1]$, we define the interpolation $a_t = (1 - t)a_i + t a_i^+$, with target velocity $a_i^+ - a_i$. The velocity field is trained with the conditional flow matching objective

$$\mathcal{L}_{\text{flow}}(\psi) = \mathbb{E}_{s, a_i, a_i^+, t}\left[\left\| G_\psi(s, a_t, t) - (a_i^+ - a_i)\right\|_2^2\right]. \tag{9}$$

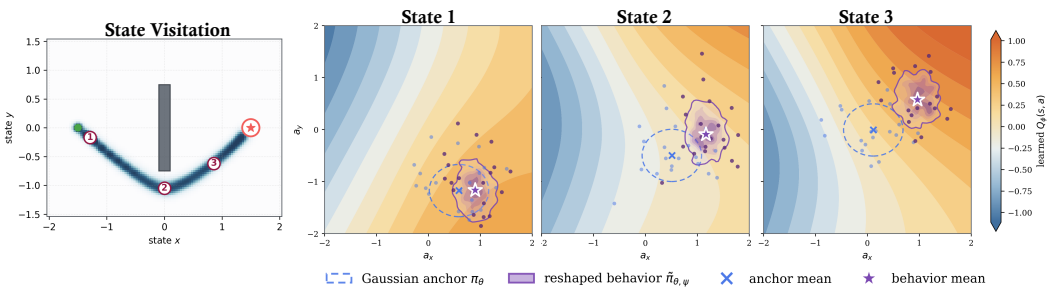


Figure 3: Behavior reshaping along a successful Wall2D trajectory. **Left:** PBR state visitation with three representative states marked along the bypass. **Right:** Gaussian anchor and reshaped action distributions over the learned $Q_\phi(s, a)$ landscape at the corresponding states. The learned transformation reshapes the local Gaussian source differently across states, producing structured interaction behavior while preserving the Gaussian policy as the optimization anchor.

At execution time, we sample a source action $a \sim \pi_\theta(\cdot \mid s)$ and integrate the learned velocity field from $t = 0$ to $t = 1$,

$$\frac{da_t}{dt} = G_\psi(s, a_t, t), \qquad a_{t=0} = a, \tag{10}$$

yielding the reshaped action

$$\tilde{a} = T_\psi(s, a) = a_{t=1}. \tag{11}$$

The environment executes $\tilde{a}$, while PPO evaluates and updates the Gaussian policy using the original source action a. Thus, PBR reshapes the behavior used for interaction without intervening the Gaussian optimized by PPO. We call this guidance *proximal* because Q_ϕ compares and reshapes policy-supported actions locally around the current Gaussian anchor, rather than searching freely over action space or directly optimizing the anchor policy. The transport cost further limits how far the reshaped action can move from its source.

We train Q_ϕ on transitions from the reshaped behavior using the current rollout and a short buffer of recent policy data. For $(s, \tilde{a}, r, s')$, with $a' \sim \pi_\theta(\cdot \mid s')$ and $\tilde{a}' = T_\psi(s', a')$, the Bellman target is $y = r + \gamma(1 - d)Q_{\bar{\phi}}(s', \tilde{a}')$. , where r, γ, d denote reward, discount factor, and termination. Notably, PBR uses the recent policy buffer only to train Q_ϕ, while PPO optimizes the Gaussian anchor from the current on-policy data. Since Q_ϕ only guides behavior reshaping, value learning can reuse more data than the current rollout without changing the PPO update. Restricting this reuse to a short recent window provides additional data for value estimation while limiting stale one that may be less relevant to the current policy.

Overall, PBR follows two central principles. First, *behavior reshaping* keeps the Gaussian policy as the optimization anchor while T_ψ induces a richer behavior distribution for interaction. This allows the behavior distribution to acquire additional structure without requiring the Gaussian policy itself to represent that structure directly. Second, *proximal value guidance* uses the action value function to compare policy supported candidate actions and construct local targets for T_ψ, rather than to directly determine the update direction of π_θ. The policy supported candidates and transport cost keep the reshaping local in action space, while Q_ϕ is learned from the current rollout together with a short buffer of recent policy data. In this way, Q_ϕ determines how behavior is reshaped, while PPO determines how the anchor policy itself evolves. Algorithm 1 in Appendix A.1 summarizes the complete training procedure.

6 Understanding PBR on Wall2D

We now return to Wall2D to examine the two design choices underlying PBR: separating the optimized policy from the behavior distribution, and using action value guidance proximally.

6.1 The Role of Behavior Reshaping

The policy expressiveness comparison in Section 4 shows that neither increased Gaussian stochasticity nor greater policy expressiveness alone guarantees a successful bypass. While gSDE succeeds by introducing temporally structured exploration, FPO++ can produce a much broader state-visitation distribution without organizing it into a coherent goal-reaching trajectory. PBR takes a different

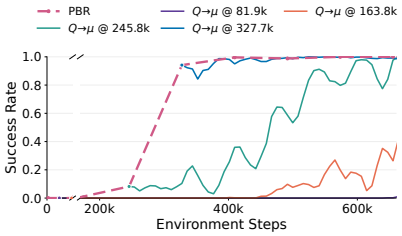


Figure 4: Effect of using Q_ϕ to update the Gaussian mean at different Wall2D training stages. Early handoffs slow learning, while later preserve high success.

Method	Steps	Start	Approach	Wall-edge	Post-wall	Success
FastTD3	409.6k	0.354	0.404	0.145	0.103	0.000
QFLEX	409.6k	0.113	–	–	–	0.000
RePPO	409.6k	-0.099	0.357	0.517	0.357	0.005
	819.2k	0.776	0.689	0.677	0.749	0.606
PBR	409.6k	0.757	0.797	0.902	0.018	0.996

Table 1: Regional local value alignment on Wall2D. All methods use 409.6k environment steps, with one later RePPO checkpoint with partial success. Values are mean state-wise cosine alignment over 32 matched action pairs.

approach: rather than replacing the optimization policy with an expressive policy class, it retains a Gaussian anchor optimized by PPO and learns a separate transformation that reshapes its sampled actions for interaction.

To examine the resulting behavior, we compare the Gaussian anchor and the reshaped behavior at representative states along a successful Wall2D trajectory. Figure 3 visualizes the Gaussian anchor π_θ, the reshaped behavior $\tilde{\pi}_{\theta,\psi}$, and the learned action-value landscape $Q_\phi(s,a)$ at three states along the bypass trajectory. As shown, the transformation modifies the anchor distribution differently across states, redirecting probability mass toward actions favored by Q critic while remaining tied to the local Gaussian source. The resulting behavior distribution therefore acquires state-dependent structure without requiring that structure to be represented directly by the policy optimized by PPO. This illustrates the role of behavior reshaping in PBR: the Gaussian policy provides the optimization anchor, while the learned transformation provides the additional structure used as actual behavior.

6.2 The Role of Proximal Value Guidance

In PBR, *proximal* refers to restricting value guidance both in where it is applied and in how it affects learning. The learned Q_ϕ guides local, policy supported action changes at the behavior level, while PPO retains control of the Gaussian anchor. This design contrasts with methods that give learned action values a more direct role in policy optimization. We examine the resulting tradeoff from two perspectives: first, how well local value differences predicted by Q_ϕ align with Monte Carlo returns along the regions encountered during learning; and second, what happens when the same Q_ϕ is instead used to update the Gaussian mean.

Action value estimates versus Monte Carlo returns. We first ask whether the learned critic and Monte Carlo returns agree on the relative value of local action changes. At each sampled state s_i, we evaluate M action pairs $\{(a^0_{im}, a^1_{im})\}_{m=1}^M$, where a^0 is a reference action and a^1 is the corresponding method-specific modification. We estimate a Monte Carlo reference value Q_{ref} for each action by executing it for one transition and rolling out the frozen policy thereafter. For each state, we construct vectors $\Delta\mathbf{q}_{\phi,i}$ and $\Delta\mathbf{q}_{\text{ref},i}$ from the M value changes $\Delta\mathbf{q} = Q(s_i,a^1) - Q(s_i,a^0)$ under the learned critic and Monte Carlo reference, respectively. We measure their agreement by

$$C_Q(s_i) = \frac{\Delta\mathbf{q}^\mathsf{T}_{\phi,i}\Delta\mathbf{q}_{\text{ref},i}}{\|\Delta\mathbf{q}_{\phi,i}\|_2\|\Delta\mathbf{q}_{\text{ref},i}\|_2}. \tag{12}$$

We report the mean C_Q within each Wall2D region.

For PBR, a^0 is sampled from the Gaussian anchor and $a^1 = T_\psi(s,a^0)$ is the corresponding reshaped action. For QFLEX, the pair consists of a Gaussian source action and its deployed flow endpoint. For RePPO and FastTD3, a^1 is obtained by a normalized local ascent step along the learned Q gradient. We use $M = 32$ pairs per state. Full evaluation details are provided in Appendix A.5.

As shown in Table 1, FastTD3 visits all four regions, yet its local value alignment remains relatively weak throughout and fails. QFLEX gives Q a broader role, using it both to improve the Gaussian policy and construct the refining flow, yet remains confined near the start where its measured alignment is also weak. RePPO at 409.6k steps its alignment is mixed and success is rare, whereas by 819.2k steps both success and alignment across regions improve substantially. Thus, direct Q guidance can succeed once reliable value information is available along the relevant behavior. PBR instead

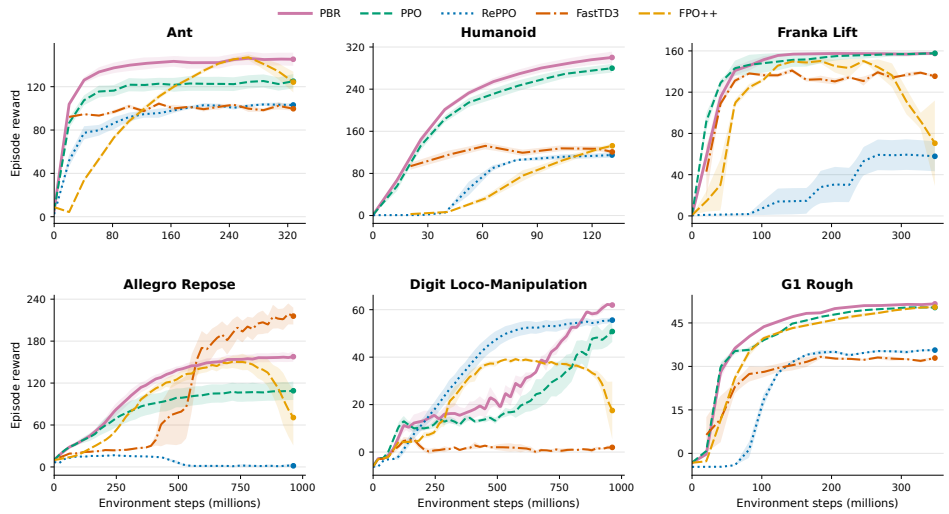


Figure 5: Performance of PBR and baseline methods across six high-dimensional Isaac Lab tasks.

reliably succeeds while showing strong local value alignment in the start, approach, and wall-edge regions, where the bypass must be acquired, despite weak alignment post-wall. This suggests that sufficiently reliable guidance in the critical regions can be enough for successful behavior.

This pattern reflects a key architectural distinction in PBR. The Gaussian anchor is updated by PPO, while Q_ϕ guides a separate transformation of the executed actions. The critic can therefore shape behavior using locally reliable value estimates without directly driving the Gaussian policy update.

Behavior level guidance versus direct policy optimization. To examine this separation more directly, we replace PPO's update of the Gaussian mean with a direct Q_ϕ gradient update, while retaining the same Q guided behavior reshaping and rollout procedure. PPO continues to update the Gaussian scale. We apply this intervention at multiple stages of PBR training, with each branch initialized from the corresponding PBR checkpoint. Figure 4 shows a strong dependence on the handoff stage. Transferring the mean update to Q_ϕ early, at 81.9k or 163.8k steps, substantially slows learning and fails in acquiring the bypass within the shown training horizon. At 245.8k steps, the intervention eventually recovers high success but much slower than PBR. By 327.7k steps, after successful behavior has already been established, the $Q \to \mu$ update largely preserves performance.

Together with the regional value analysis, the results support separating behavior level value guidance from policy optimization. Early in learning, using Q_ϕ to reshape behavior while PPO updates the Gaussian anchor is more robust than allowing it to directly determine the mean update. Once useful behavior and more reliable value estimates have formed, this distinction becomes less critical.

Across the Wall2D analyses, the two components of PBR play complementary roles: behavior reshaping provides the structured interaction needed to acquire the bypass, while proximal value guidance exploits locally useful action values without intervening the Gaussian policy update. We next test whether these design choices extend to high dimensional locomotion and manipulation tasks.

7 PBR in High-Dimensional Continuous Control

We first evaluate PBR on high-dimensional continuous-control problems in Isaac Lab to test how well the method scales beyond the controlled Wall2D setting. We compare PBR with PPO, RePPO, FastTD3, and FPO++ across six Isaac Lab tasks. The benchmark spans locomotion and manipulation settings and tests whether PBR can retain the stability of its PPO anchor while benefiting from behavior reshaping and proximal value guidance at scale.

Figure 5 shows consistently strong performance across six tasks. PBR achieves the highest final return on Ant, Humanoid, Digit Loco-Manipulation, and G1 Rough, and matches PPO near the Franka Lift task ceiling. On Allegro Repose, FastTD3 eventually overtakes PBR, while the other baselines show more task-dependent performance.

Taken together, these results support the tradeoff view introduced in Section 3. Although PBR does not dominate in every task, it is competitive or leading across different high-dimensional control

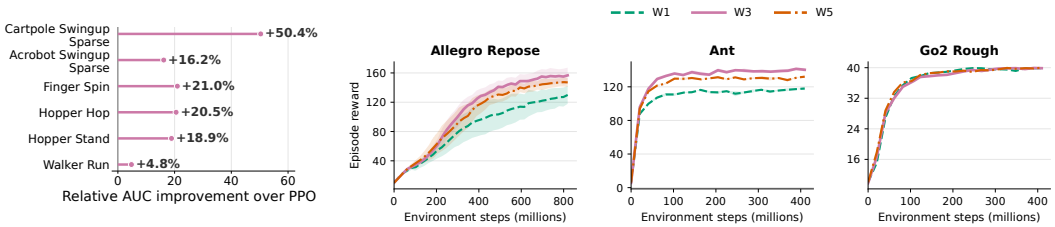


Figure 6: Additional evidence for PBR across data reuse and task suites. **Left:** Relative learning-curve AUC improvement of PBR over PPO on six MuJoCo Playground tasks. PBR improves over PPO on most tasks while remaining comparable on the others, showing that the gains are not specific to Isaac Lab. **Right:** Effect of the recent-policy data window. A short amount of reuse improves performance on Allegro Repose and Ant, while Go2 Rough is largely insensitive to the window size; increasing the window beyond $W = 3$ provides no consistent additional benefit.

regimes. Methods that make more extreme choices in the design space can achieve stronger performance in particular settings, as FastTD3 does on Allegro Repose, but these advantages are less consistent across the benchmark. PBR instead combines richer interaction behavior and value guidance while retaining PPO as the optimization anchor, providing a consistent operating point across tasks.

We additionally evaluate QFLEX on two Isaac Lab tasks in Appendix B.2. The original method was evaluated on a different benchmark stack, and transferring it to Isaac Lab requires additional action-space Q-gradient computation and implementation changes. Our adaptation did not recover comparable performance on the tested tasks, so we report these results separately rather than include QFLEX in the full six-task comparison.

The methods also differ in computational cost. PBR is slower than PPO because it additionally learns Q_ϕ and the behavior transformation, but remains substantially cheaper than the other value-guided and flow-based alternatives in our implementation. On Ant, PBR requires about $3.4\times$ the wall-clock time of PPO, while RePPO, FPO++, and FastTD3 require roughly 2.0–$2.6\times$ the wall-clock time of PBR, and our QFLEX adaptation about $9.4\times$. Full timing results are reported in Appendix B.5.

Effect of recent-policy data reuse. PBR learns Q_ϕ from the current rollout together with a short window of recent-policy data. In the right panel of Figure 6, we vary this window while keeping all other components fixed. Reusing recent data improves performance over the shortest window on Allegro Repose and Ant, with $W = 3$ performing strongest overall, while extending the window to $W = 5$ provides no consistent additional benefit. On Go2 Rough, all three choices perform similarly. These results support limited recent-policy reuse as a useful middle ground: additional data can improve value learning, while increasingly historical experience does not provide uniform gains.

MuJoCo Playground. We further compare PBR with PPO on six MuJoCo Playground control tasks in the left panel of Figure 6. PBR achieves higher learning-curve AUC on all six tasks, with relative gains up to 50.4%. These results provide additional evidence that the improvements observed in Isaac Lab extend to a different simulation suite. Full learning curves for all six MuJoCo Playground tasks are provided in Appendix C.

8 Conclusion

We introduced Proximal Behavior Reshaping (PBR), which retains a Gaussian policy as the PPO optimization anchor while learning a separate, value-guided transformation for interaction. This design balances richer interaction behavior with limited reliance on learned value estimates. Our analysis shows that behavior reshaping can provide structured interaction without requiring the Gaussian policy itself to represent that structure, while proximal value guidance exploits useful local action values without letting imperfect Q estimates directly govern the policy update. Across high-dimensional tasks, PBR achieves strong and consistent performance while incurring substantially less computational overhead than more complex value-guided and flow-based alternatives. More broadly, our results show that the policy optimized for learning, the behavior used for interaction, and learned value guidance need not be collapsed into the same mechanism. Separating these roles creates useful design freedom for continuous control.

AI USE STATEMENT

We used generative AI tools to assist with language editing and LaTeX formatting. We did not use generative AI to generate synthetic data; process or analyze data; interpret results; develop theoretical models or conceptual frameworks. All AI-assisted edits were reviewed and revised by the authors to ensure their accuracy and consistency with the underlying research. We take responsibility for the final content of this work, including all text and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, sensitive personal data, or foreseeable ethical risks.

REPRODUCIBILITY STATEMENT

We provide the methodological and experimental details needed to reproduce our results in the main paper and appendix. Code will be released upon publication.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. Pmlr, 2018.

Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Anji Liu, Yitao Liang, and Guy Van den Broeck. Off-policy deep reinforcement learning with analogous disentangled exploration. *arXiv preprint arXiv:2002.10738*, 2020.

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.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

Antonin Raffin and Freek Stulp. Generalized state-dependent exploration for deep reinforcement learning in robotics. *Arxiv*, 2020.

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.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Younggyo Seo, Carmelo Sferrazza, Haoran Geng, Michal Nauman, Zhao-Heng Yin, and Pieter Abbeel. Fasttd3: Simple, fast, and capable reinforcement learning for humanoid control. *arXiv preprint arXiv:2505.22642*, 2025.

Claas Voelcker, Axel Brunnbauer, Marcel Hussing, Michal Nauman, Pieter Abbeel, Radu Grosu, Eric Eaton, Amir-massoud Farahmand, and Igor Gilitschenski. Relative entropy pathwise policy optimization. In *International Conference on Learning Representations*, volume 2026, pp. 35761–35790, 2026.

Yunyue Wei, Chenhui Zuo, and Yanan Sui. Scalable exploration for high-dimensional continuous control via value-guided flow. *arXiv preprint arXiv:2601.19707*, 2026.

William F Whitney, Michael Bloesch, Jost Tobias Springenberg, Abbas Abdolmaleki, Kyunghyun Cho, and Martin Riedmiller. Decoupled exploration and exploitation policies for sample-efficient reinforcement learning. *arXiv preprint arXiv:2101.09458*, 2021.

Shunpeng Yang, Ben Liu, and Hua Chen. Policyflow: Policy optimization with continuous normalizing flow in reinforcement learning. *arXiv preprint arXiv:2602.01156*, 2026.

Brent Yi, Hongsuk Choi, Himanshu Gaurav Singh, Xiaoyu Huang, Takara E Truong, Carmelo Sferrazza, Yi Ma, Rocky Duan, Pieter Abbeel, Guanya Shi, et al. Flow policy gradients for robot control. *arXiv preprint arXiv:2602.02481*, 2026.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *Advances in Neural Information Processing Systems*, 38: 106282–106319, 2026.

A Appendix

A.1 Algorithm: Proximal Behavior Reshaping

Algorithm 1 Proximal Behavior Reshaping (PBR)

Require: Gaussian anchor π_θ, critics $V, Q_\phi, Q_{\bar{\phi}}$, flow G_ψ, candidates K, coefficients β_Q, β_c

Initialize recent policy buffer $\mathcal{B}$ for executed transitions
for each training iteration **do**
  1. Environment Interaction
  for each environment step **do**
    Sample $a \sim \pi_\theta(\cdot \mid s)$ and compute $\tilde{a} = T_\psi(s, a)$ by integrating G_ψ
    Execute $\tilde{a}$; store (s, a, r, d) for PPO and $(s, \tilde{a}, r, s', d)$ in $\mathcal{B}$
  2. Action value update
  for minibatches from $\mathcal{B}$ **do**
    Sample $a' \sim \pi_\theta(\cdot \mid s')$, $\tilde{a}' = T_\psi(s', a')$
    $y \leftarrow r + \gamma(1-d)Q_{\bar{\phi}}(s', \tilde{a}')$
    Update Q_ϕ with $\big(Q_\phi(s, \tilde{a}) - y\big)^2$; soft update $Q_{\bar{\phi}}$
  3. Proximal target and flow update
  for each sampled state s **do**
    Sample $a_i, a'_j \sim \pi_\theta(\cdot \mid s)$, $i, j = 1, \ldots, K$
    for each source a_i **do**
      Augment the candidates with the identity $a'_{i0} = a_i$
      Compute, for $j = 1, \ldots, K$,

$$\Delta Q_{ij} = Q_\phi(s, a'_{ij}) - Q_\phi(s, a_i), \qquad \overline{\Delta Q}_{ij} = \frac{\Delta Q_{ij}}{\mathrm{Std}_k[\Delta Q_{ik}] + \epsilon}$$

      Compute

$$c_{ij} = \frac{1}{D_a} \left\| \frac{a'_{ij} - a_i}{\sigma_\theta(s)} \right\|_2^2, \qquad S_{ij} = \beta_Q \overline{\Delta Q}_{ij} - \beta_c c_{ij}$$

      Set
      $(p_{i0}, \ldots, p_{iK}) = \mathrm{Softmax}(S_{i0}, \ldots, S_{iK})$
      Sample a_i^+ from $\{a_i, a'_1, \ldots, a'_K\}$ according to p_i
      Sample $t \sim \mathcal{U}[0, 1]$, $a_t = (1-t)a_i + t a_i^+$
      Update G_ψ by minimizing

$$\big\| G_\psi(s, a_t, t) - (a_i^+ - a_i) \big\|_2^2$$

  4. PPO update
  Update π_θ and V with PPO using source actions a and rewards generated by $\tilde{a}$

A.2 Hyperparameters

PBR uses the same PPO architecture and PPO settings as the corresponding baseline for each task. The actor, PPO critic, Q_ϕ, and flow network share the same hidden widths within each task and use ELU activations. Table 2 summarizes the remaining shared and task-specific settings.

A.3 Wall2D Environment Details

Wall2D is a two-dimensional continuous-control navigation task on $[-2, 2] \times [-1.5, 1.5]$. A vertical wall occupies $[-0.1, 0.1] \times [-0.75, 0.75]$, leaving symmetric passages above and below. Episodes start uniformly within a radius-0.05 disk centered at $(-1.5, 0)$, and the goal is centered at $(1.5, 0)$ with success radius 0.15. The state is the position $p_t = (x_t, y_t)$ with the observation as the normalized position and goal-relative displacement, $o_t = \left[\frac{x_t}{2}, \frac{y_t}{1.5}, \frac{x_g - x_t}{4}, \frac{y_g - y_t}{3}\right]$, $(x_g, y_g) = (1.5, 0)$. Actions lie in $[-1, 1]^2$ and follow deterministic single-integrator dynamics, $p_{t+1} = p_t + 0.15\ \mathrm{clip}(a_t, -1, 1)$. Collision detection checks the full motion segment against the wall, preventing the agent from crossing it in a single step. Let $D(p)$ denote the shortest collision-free path

Table 2: Hyperparameters for the Isaac Lab benchmarks. PBR uses the same PPO architecture and PPO settings as the corresponding PPO baseline. Within each task, the actor, PPO critic, Q_ϕ, and flow network use the same hidden widths. K and M denote the numbers of source and proposal actions; β_Q and β_c control normalized value improvement and transport cost.

Shared settings

PPO		*Action value learning*		*Behavior reshaping*	
Activation	ELU	Target update τ	0.005	Source actions K	16
GAE λ	0.95	Recent policy window W	3	Proposal actions M	16
PPO clip ϵ	0.2	Q target samples	1	Flow integration steps	5
PPO epochs	5	Q/flow epochs	1	Flow integration range	$[0, 1]$
		Q loss	MSE	Flow time embedding	16
				Normalization ϵ	10^{-6}

Task-specific settings

Task	Hidden widths	γ	$\mathrm{lr}_{\mathrm{PPO}}$	$\mathrm{lr}_Q = \mathrm{lr}_G$	β_Q	β_c
Wall2D	PPO:$(64, 64)$, Q/G : $(128, 128)$	0.99	3×10^{-4}	3×10^{-4}	2	1
Ant	$(400, 200, 100)$	0.99	5×10^{-4}	5×10^{-4}	1	1
Humanoid	$(400, 200, 100)$	0.99	5×10^{-4}	5×10^{-4}	1	0.5
Franka Lift	$(256, 128, 64)$	0.98	10^{-4}	10^{-4}	1	0.5
Allegro Repose	$(512, 256, 128)$	0.998	10^{-3}	10^{-3}	1	1
Digit Loco-Manip.	$(256, 128, 128)$	0.99	10^{-3}	10^{-3}	1	0.5
G1 Rough	$(512, 256, 128)$	0.99	10^{-3}	10^{-3}	1	0.5

distance from p to the goal. The dense reward is

$$r_t = D(p_t) - D(p_{t+1}) - 0.01.$$

Reaching the goal adds $+10$. Wall collisions or leaving the workspace incur -2 and terminate the episode. Success is defined by a valid proposed next position within distance 0.15 of the goal. Episodes are truncated after 64 transitions.

A.4 Wall2D State Visitation over Training

Figure 7 shows the evolution of state visitation for all Wall2D methods across matched training checkpoints. It complements the main-text comparison by showing when each method develops, or fails to develop, a coherent route around the wall.

A.5 Monte Carlo Evaluation of Local Action Values

We evaluate whether each learned action value function correctly predicts the relative value of local action changes relevant to its method. All policies, critics, and behavior transformations are frozen during evaluation, and states are collected from fresh rollouts of each method's native behavior.

State collection and matched action pairs. We partition Wall2D into four fixed regions based on the agent position $s = (x, y)$:

$$\mathcal{R}_{\mathrm{start}} = \{(x, y) : x < -1.10\}, \tag{13}$$

$$\mathcal{R}_{\mathrm{approach}} = \{(x, y) : -1.10 \le x < -0.25\}, \tag{14}$$

$$\mathcal{R}_{\mathrm{wall}} = \{(x, y) : -0.25 \le x \le 0.25,\ |y| \ge 0.72\}, \tag{15}$$

$$\mathcal{R}_{\mathrm{post}} = \{(x, y) : x > 0.25\}. \tag{16}$$

We refer to these as the start, approach, wall-edge, and post-wall regions. For each available method-region pair, we evaluate 256 states sampled from that method's own visitation distribution. Regions not sufficiently visited by a method are left unevaluated; states outside these four regions are excluded.

At each state s_i, we construct $M = 32$ matched action pairs $\{(a^0_{im}, a^1_{im})\}_{m=1}^{M}$, where a^0 is a source action and a^1 is the corresponding method-specific modification.

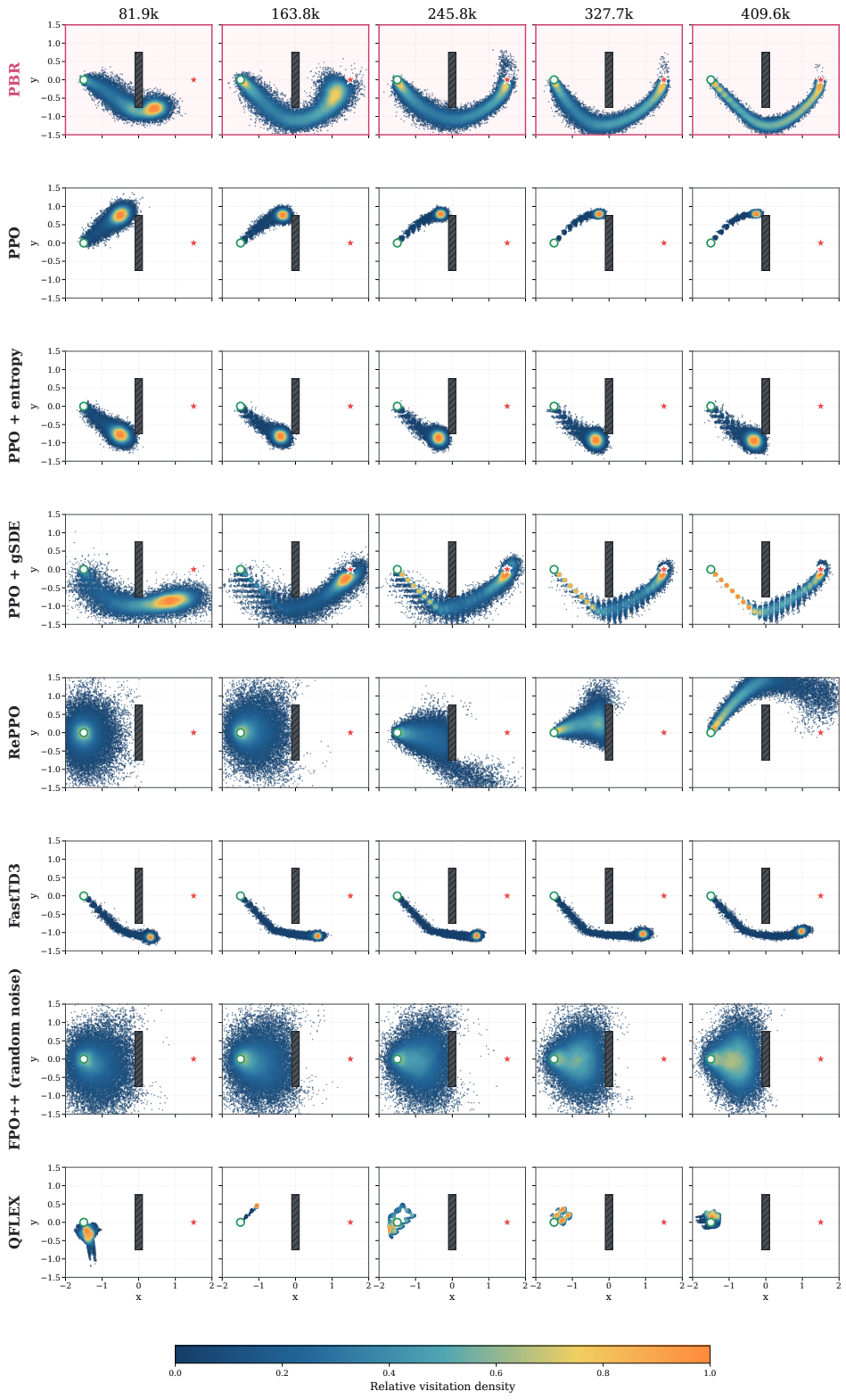


Figure 7: Wall2D state visitation across training. Columns show matched checkpoints from 81.9k to 409.6k environment interactions, and rows show PBR and the comparison methods. PBR progressively develops a coherent bypass around the wall, while the baselines exhibit distinct exploration patterns and learning dynamics.

Method-specific action pairs. For PBR,

$$a^0_{im} \sim \pi_\theta(\cdot \mid s_i), \qquad a^1_{im} = T_\psi(s_i, a^0_{im}),$$

so the pair directly evaluates the behavior change induced by the learned transformation. For QFLEX, a^0_{im} is sampled from its learned Gaussian source policy and a^1_{im} is the corresponding deployed flow endpoint.

For FastTD3 and RePPO, we probe the local action direction preferred by the learned critic. Given source action a^0_{im}, we construct

$$a^1_{im} = a^0_{im} + \rho \frac{\nabla_a Q_\phi(s_i, a^0_{im})}{\max(\|\nabla_a Q_\phi(s_i, a^0_{im})\|_2, \epsilon)}, \qquad \rho = 0.2. \tag{17}$$

For FastTD3, a^0_{im} follows its deterministic actor with native exploration noise and $Q_\phi = \min(Q_{\phi_1}, Q_{\phi_2})$; for RePPO, a^0_{im} is sampled from its frozen stochastic policy. These probes isolate the local critic direction and are not reconstructions of the full parameter update.

Monte Carlo reference. For each queried action, we restore the saved simulator state, force that action for one transition, and then roll out the frozen method-specific policy for the remainder of the episode. For rollout r,

$$G_r(s,a) = \sum_{j=0}^{H-1} \gamma^j r^r_j, \qquad Q_{\mathrm{ref}}(s,a) = \frac{1}{R} \sum_{r=1}^{R} G_r(s,a), \tag{18}$$

where H is the remaining episode horizon. We use $R = 32$ rollouts per queried action for PBR, QFLEX, and RePPO. The two actions in each matched pair use the same rollout randomness. FastTD3 is deterministic, so one complete rollout provides the exact reference return.

After the forced action, PBR executes its frozen Gaussian anchor followed by the learned transformation; QFLEX uses its frozen Gaussian reference policy; FastTD3 uses its frozen deterministic actor; and RePPO uses its frozen stochastic policy. For RePPO, G_r is the discounted soft return matching its learned action value objective. Absolute return-difference magnitudes are therefore not compared between RePPO and the other methods.

Local value alignment. For each matched pair, we compute

$$\Delta Q_{\phi,im} = Q_\phi(s_i, a^1_{im}) - Q_\phi(s_i, a^0_{im}), \qquad \Delta Q_{\mathrm{ref},im} = Q_{\mathrm{ref}}(s_i, a^1_{im}) - Q_{\mathrm{ref}}(s_i, a^0_{im}). \tag{19}$$

For each state, $\Delta \mathbf{q}_{\phi,i}$ and $\Delta \mathbf{q}_{\mathrm{ref},i}$ contain the M value changes under the learned critic and Monte Carlo reference, respectively. Their alignment is

$$C_Q(s_i) = \frac{\Delta \mathbf{q}^\top_{\phi,i} \Delta \mathbf{q}_{\mathrm{ref},i}}{\|\Delta \mathbf{q}_{\phi,i}\|_2 \|\Delta \mathbf{q}_{\mathrm{ref},i}\|_2}. \tag{20}$$

Positive values indicate agreement between critic-predicted and Monte Carlo return changes, while negative values indicate opposing changes. We compute C_Q at each state and report its mean within each Wall2D region, providing a regional measure of local value alignment.

B Additional Isaac Lab Experiments

B.1 Experimental Setup

All Isaac Lab experiments are run on NVIDIA RTX 4090 GPUs with results over five random seeds. Training progress is measured in environment interactions, and all methods are evaluated under the same interaction budget for each task.

We evaluate six Isaac Lab tasks spanning locomotion and manipulation: Ant, Humanoid, Franka Lift, Allegro Repose, Digit Loco-Manipulation, and G1 Rough. We use the standard task definitions and environment settings from Isaac Lab unless otherwise noted. PBR uses the same PPO architecture, observations, rewards, environment count, and PPO optimization settings as the corresponding PPO baseline. PBR-specific hyperparameters are summarized in Table 2.

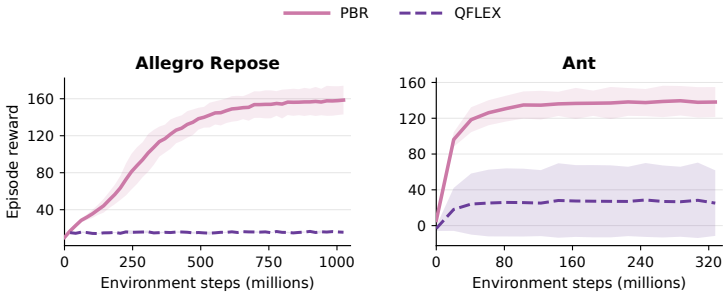


Figure 8: PBR and the released QFLEX learner adapted to two Isaac Lab tasks. The QFLEX core learning algorithm is retained, while a custom adapter connects it to the Isaac Lab environment and evaluation pipeline.

Baseline implementations. PPO and PBR use RSL-RL[1], with PBR implemented directly in the same framework. RePPO uses its released RSL-RL implementation[2], FastTD3 uses its released codebase[3], FPO++ uses `fpo-control`[4], and QFLEX uses its released implementation[5]. PPO uses the native Isaac Lab/RSL-RL integration; for RePPO, FastTD3, and FPO++ we adapt the environment interface and task configuration, with additional evaluation-script changes for FastTD3 and FPO++. QFLEX uses a custom Isaac Lab environment adapter and evaluation pipeline. The core baseline algorithms and their default optimization logic are left unchanged.

B.2 QFLEX on Isaac Lab

QFLEX was not originally evaluated on Isaac Lab. We therefore adapt its released JAX implementation to the same Isaac Lab task interface used by the other methods. We retain the original QFLEX learner and core update rules, while adding an environment adapter that transfers observations and actions between the PyTorch-based simulator and the JAX learner. Our adaptation uses 256 parallel environments, a replay buffer of 10^6 transitions, batch size 256, and 30k random-action warm-up transitions. We perform four learner updates per vectorized environment step. Actions produced by QFLEX are stored in replay before environment clipping, while the environment receives the bounded action. Evaluation is deterministic and uses 256 parallel environments.

Figure 8 reports QFLEX on Ant and Allegro Repose. Under this adaptation, QFLEX remains substantially below PBR on both tasks. Because the original method was developed and evaluated with a different benchmark stack, we report these results separately as a best-effort Isaac Lab transfer rather than including QFLEX in the full six-task comparison.

B.3 Direct Q Guided Action Selection

To isolate the role of the learned behavior transformation on the **Digit Loco-Manipulation** task, we compare PBR with a direct Q guided sampling baseline, denoted **PPO + Q-Sample**. The baseline retains the same Gaussian PPO anchor and learned action value function, but removes the residual flow. For each source action sampled from the Gaussian policy, it generates a local set of candidate actions, scores them by their predicted value improvement,

$$\Delta Q_\phi(s, a) = Q_\phi(s, a) - Q_\phi(s, a_0),$$

and samples an action according to the corresponding Boltzmann weights. The selected action is then executed directly, while PPO continues to optimize the Gaussian anchor. Thus, PPO + Q-Sample uses the same local value information as PBR, but applies it through direct candidate selection rather than a learned behavior transformation. Figure 9 shows that this direct sampling variant

[1] `https://github.com/leggedrobotics/rsl_rl`
[2] `https://github.com/cvoelcker/rsl_rl`
[3] `https://github.com/younggyoseo/FastTD3`
[4] `https://github.com/amazon-far/fpo-control`
[5] `https://github.com/LNSGroup/Qflex`

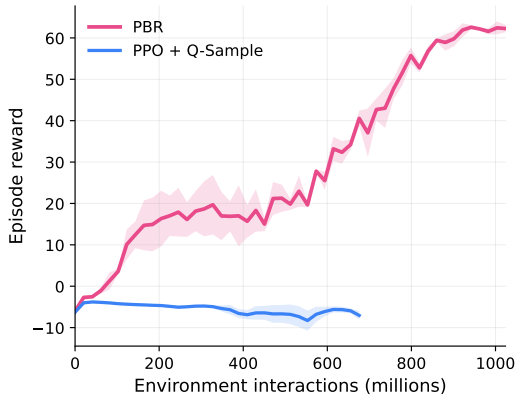


Figure 9: PBR compared with PPO + Q-Sample on Digit Loco-Manipulation. PPO + Q-Sample removes the learned behavior transformation and instead directly selects among locally sampled actions using the learned action value function. The variant collapses and is stopped early after sustained failure, while PBR continues to improve.

quickly collapses and remains near the minimum return throughout training, whereas PBR continues to improve. We therefore terminate PPO + Q-Sample early after its failure persists. This result shows that directly selecting among critic-preferred local candidates is not sufficient to reproduce the structured behavior learned by PBR; amortizing these local preferences into a state-dependent transformation is an important component of the method.

B.4 Value and Transport Guidance Ablation

We ablate the two coefficients used in proximal target construction on Ant. Here, β_Q controls the contribution of normalized value improvement, while β_c controls the transport penalty. We compare the full setting $(\beta_Q, \beta_c) = (1, 1)$ with variants that remove either value guidance $(0, 1)$ or transport regularization $(1, 0)$. As shown in Figure 10, the full setting achieves the highest mean final return. Removing either term reduces performance, suggesting that value guidance and transport regularization play complementary roles in constructing proximal targets.

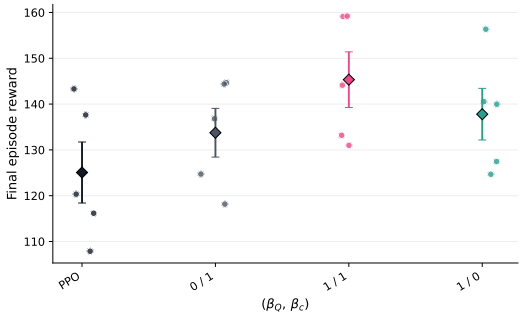


Figure 10: Ablation of value guidance and transport regularization on Ant. Diamonds show mean final return and points show individual seeds. The full setting $(\beta_Q, \beta_c) = (1, 1)$ performs best on average, while removing either term reduces performance.

B.5 Wall Clock Training Cost

The methods differ substantially in computational cost under the same environment interaction budget. Figure 11 reports the wall clock time required to reach 327.68M environment steps on Ant. PBR is approximately $3.4\times$ slower than PPO due to the additional action value and behavior transformation updates, but remains cheaper than the other comparison methods in our implementation.

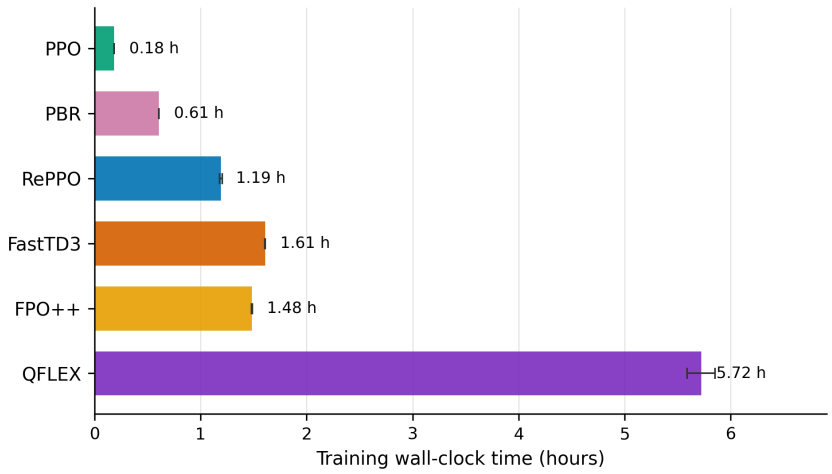


Figure 11: Wall clock training time to reach 327.68M environment interactions on Ant. All methods use the same interaction budget. Error bars show variation across runs.

RePPO, FPO++, and FastTD3 require approximately 2.0–$2.6\times$ the wall clock time of PBR, while our QFLEX adaptation is substantially more expensive. These measurements reflect our implementations and hardware configuration and complement the environment interaction comparison in the main text.

C MuJoCo Playground Results

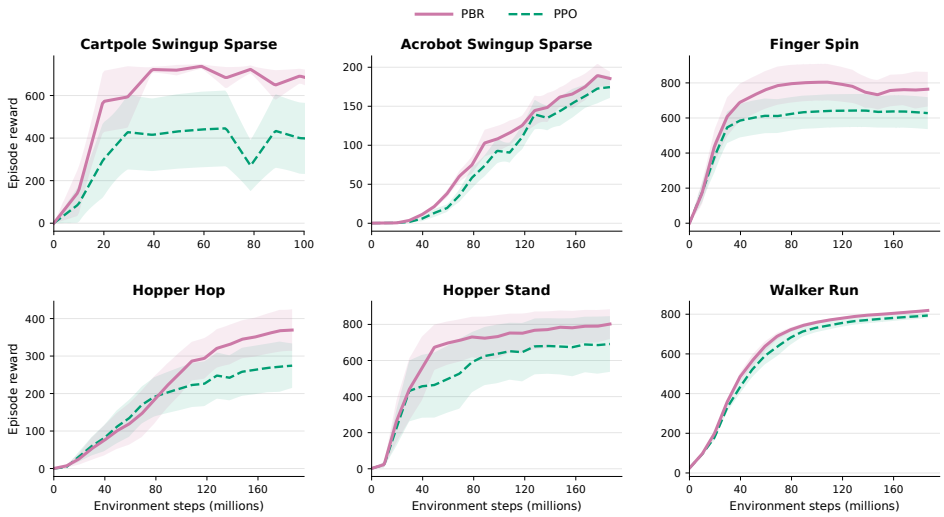


Figure 12: Full learning curves for PBR and PPO across six MuJoCo Playground control tasks. These curves correspond to the relative AUC comparison reported in the main text.

Figure 12 shows the full learning curves underlying the MuJoCo Playground AUC comparison in the main text. We compare PBR with PPO on six control tasks using matched interaction budgets. The curves show how the relative AUC improvements reported in Figure 6 arise throughout training rather than only at the final checkpoint.