Flow Policy Gradients for Robot Control

Brent Yi[1,2,†,*], Hongsuk Choi[1,2,†,*], Himanshu Gaurav Singh[1,2,†], Xiaoyu Huang[1,2,†]
Takara E. Truong[1,3,†], Carmelo Sferrazza[1], Yi Ma[2,4], Rocky Duan[1,‡]
Pieter Abbeel[1,2,‡], Guanya Shi[1,5,‡], Karen Liu[1,3,‡], Angjoo Kanazawa[1,2,‡]
[1]Amazon FAR [2]UC Berkeley [3]Stanford [4]HKU [5]CMU
*Equal Contribution †Work done as an intern at Amazon FAR ‡Amazon FAR team co-lead

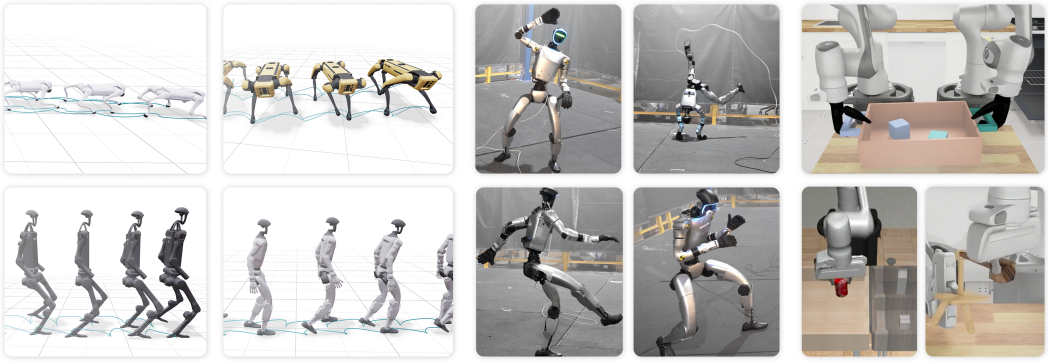


Figure 1: **Flow policies for robot control.** We show how robust control policies for quadrupeds, humanoids, and manipulators can be trained with reinforcement learning.

Abstract: Likelihood-based policy gradient methods are the dominant approach for training robot control policies from rewards. These methods rely on differentiable action likelihoods, which constrain policy outputs to simple distributions like Gaussians. In this work, we show how flow matching policy gradients, a recent framework that bypasses likelihood computation, can be made effective for training and finetuning more expressive policies in challenging robot control settings. We introduce an improved objective that enables success in legged locomotion, humanoid motion tracking, and manipulation tasks, as well as robust sim-to-real transfer on two humanoid robots. We then present ablations and analysis on training dynamics. Results show how policies can exploit the flow representation for exploration when training from scratch, as well as improved finetuning robustness over baselines. Project webpage: hongsukchoi.github.io/fpo-control.

Keywords: Flow Matching, Policy Gradients, Robot Control

1 Introduction

Likelihood-based policy gradient methods [1, 2] have driven a broad range of robot control results. This includes recent advances in legged locomotion [3], whole-body humanoid control [4, 5], and in-hand manipulation [6], where policies are trained from scratch, as well as for manipulation policies that are pretrained with demonstrations [7, 8].

Policy gradient algorithms are traditionally run by sampling actions from stochastic policies, followed by policy updates that backpropagate through the differentiable likelihoods of sampled actions [1]. This is effective for simple action distributions, but challenging for more expressive policy representations: computing likelihoods in the flow policies used for imitation learning [9, 10], for example, requires expensive sampling or integration to account for volume changes in the underlying flow field [11, 12]. Reinforcement learning (RL) algorithms that support these representations

10th Conference on Robot Learning (CoRL 2026), Austin TX, USA.

promise more expressive distributions for exploration when training from scratch [13], as well as post-training of more capable pretrained policies [7, 8]—ideally within a single algorithm and policy representation that spans both regimes.

We study robot control with flow matching policy gradients [13], a framework for flow RL that uses conditional flow matching to circumvent likelihoods entirely. Specifically, we show how this class of algorithm—previously only validated in simpler synthetic settings—can be made practical for training robot control policies in more challenging settings.

Our contributions are as follows. **(1)** We introduce an improved flow policy gradient algorithm that we call FPO++. FPO++ proposes two simple changes, per-sample ratio clipping and an asymmetric trust region, that enable more robust training in challenging robotics settings. **(2)** We use tasks in legged locomotion, humanoid motion tracking, and both single-arm and bimanual manipulation to demonstrate settings where FPO++ succeeds: in learning policies from scratch, in sim-to-real transfer, and in finetuning policies that are pretrained from demonstrations. **(3)** We ablate algorithmic choices, showing that per-sample ratios consistently improve performance while the asymmetric trust region is important for entropy preservation when training from scratch. **(4)** We analyze training dynamics and find that FPO++ learns correlated action distributions that diagonal Gaussian policies cannot represent, producing qualitatively different gaits from the same rewards and showing improved sample efficiency and robustness to base policy choice when finetuning.

2 Background and Related Work

Policy gradients and PPO. Policy gradient techniques are the dominant approach for continuous control with RL [14, 3, 6, 15, 16]. These algorithms are on-policy; rollouts in the form of observation, action, and reward tuples (o_t, a_t, r_t) for each environment timestep t are used to update a policy $\pi_\theta(a_t \mid o_t)$ to maximize expected return. In the robotics community, the standard approach for achieving this is the clipped Proximal Policy Optimization (PPO) [2] objective. For an action with likelihood ratio ρ_θ and advantage estimate [17] $\hat{A}_t$, this can be written as:

$$\psi_{\text{PPO}}\left(\rho_\theta, \hat{A}_t\right) = \min\left(\rho_\theta \hat{A}_t,\ \text{clip}(\rho_\theta, 1 \pm \varepsilon^{\text{clip}})\hat{A}_t\right). \tag{1}$$

The overall optimization problem is then

$$\max_\theta\ \mathbb{E}_{\pi_{\theta_{\text{old}}}}\left[\psi_{\text{PPO}}\left(\rho_\theta, \hat{A}_t\right)\right];\quad \rho_\theta = \frac{\pi_\theta(a_t \mid o_t)}{\pi_{\theta_{\text{old}}}(a_t \mid o_t)}. \tag{2}$$

PPO is popular because it is simple to implement and provides strong empirical performance. It also inherits advantages of general policy gradient algorithms: unlike methods that rely on learned Q functions [18] or transition models [19], the PPO objective requires differentiability only from action likelihoods and not from rewards or environment dynamics.

Flow and diffusion policies. Flow and diffusion models [20, 21], which we consider equivalent in practice [22], are at the current frontier for supervised policy learning with continuous action spaces. Systems relying on these models typically train robot manipulation policies from human demonstrations [9, 23, 24], and have been validated with language conditioning at larger scales [10, 25, 26]. Similar approaches have also been adopted in whole-body humanoid control, where iterative generative policies are supervised by Gaussian experts [27, 4]. All of these approaches need expert action labels. Instead, we study online RL where supervision is only provided in the form of environment rewards. This can be used to learn new behaviors or to finetune pretrained policies.

RL for flow and diffusion policies. Beyond imitation, recent works have also begun to integrate flow and diffusion policies into reinforcement learning settings. Many of these works study offline RL from static datasets (Appendix A). We instead consider online RL using policy gradient-style training. Existing work in this space is primarily focused on mechanisms for likelihood computation. Similar to image diffusion RL techniques that formulate denoising as an MDP [28, 29], DPPO [7] and ReinFlow [8] optimize likelihoods computed from stochastic sampling noises. NCDPO [30] optimizes likelihoods computed from both the initial noise and stochastic sampler noises, while

backpropagating through unrolled denoising steps. GenPO [31] also unrolls denoising steps, while incorporating an invertible architecture inspired by normalizing flows [32]. In contrast, we study a flow policy gradient [13] approach that bypasses likelihoods entirely. It does not rely on noise likelihoods from specific stochastic sampling trajectories, which (i) inflate the credit assignment horizon and (ii) are not equivalent to action likelihoods marginalized over initial noises and sampling trajectories. It does not require specific network architectures or unrolling, which is expensive and risks vanishing or exploding gradients. We describe how this is achieved in the next section.

3 FPO++

We introduce FPO++, an updated version of the FPO (Flow Policy Optimization) algorithm that succeeds in real-world robotics tasks. We summarize FPO, then discuss FPO++.

3.1 Preliminaries

Flow matching policy gradients. FPO [13] aims to enable policy gradient-style training of policies parameterized as flow models [20], without explicit likelihoods. Although action likelihoods under flow policies can be computed by accounting for changes in volume, the direct or indirect divergence integration required for this [11] is computationally prohibitive in RL settings.

FPO addresses this by proposing a surrogate for ρ_θ,

$$\hat{\rho}_{\text{FPO}}(\theta) = \exp\Big(\hat{\mathcal{L}}_{\text{CFM},\theta_{\text{old}}}(a_t; o_t) - \hat{\mathcal{L}}_{\text{CFM},\theta}(a_t; o_t)\Big), \tag{3}$$

where $\hat{\mathcal{L}}_{\text{CFM},\theta}(a_t; o_t)$ is a Monte Carlo estimate of the conditional flow matching (CFM) loss.

This formulation enables PPO-style training of flow-based policies, which can express more complex distributions than the diagonal Gaussians that are most common in online reinforcement learning for robotics [3, 6, 5]. FPO mirrors PPO's clipped objective (Equation 2), maximizing:

$$\max_\theta \quad \mathbb{E}_{\pi_{\theta_{\text{old}}}}\left[\psi_{\text{PPO}}\Big(\hat{\rho}_{\text{FPO}}(\theta), \hat{A}_t\Big)\right]. \tag{4}$$

Intuitively, FPO's ratio approximation uses CFM loss differences to approximate action log-likelihood differences. The final objective (Equation 4) then uses advantage estimates to shift probability flow toward higher-reward actions.

Conditional flow matching loss. To estimate CFM losses, FPO first draws N_{mc} noise $\epsilon_i \sim \mathcal{N}(0, I)$ and flow step $\tau_i \in [0, 1]$ pairs for each action a_t and $i \in \{1 \dots N_{\text{mc}}\}$. Noised actions are then computed using an interpolation schedule. Linear interpolation is common in flow models:

$$a_t^{\tau_i} = \tau_i a_t + (1 - \tau_i)\epsilon_i. \tag{5}$$

Squared errors are computed and averaged for the policy's velocity predictions $\hat{v}_\theta$,

$$\hat{\mathcal{L}}_{\text{CFM},\theta}(a_t; o_t) = \frac{1}{N_{\text{mc}}} \sum_i^{N_{\text{mc}}} \ell_\theta^{(i,t)} \tag{6}$$

$$\ell_\theta^{(i,t)} = \ell_\theta(a_t, \tau_i, \epsilon_i; o_t) = \|\hat{v}_\theta(a_t^{\tau_i}, \tau_i; o_t) - (a_t - \epsilon_i)\|_2^2. \tag{7}$$

These losses can then be used in policy updates, which aim to increase or decrease CFM losses based on the sign of corresponding action advantages. Without loss of generality, the network can also be trained to predict the clean action a_t, noise ϵ_i, or a different combination of the pair [22].

3.2 FPO++

While the vanilla FPO formulation succeeds in synthetic benchmarks [13], we found that it required refinements to achieve reliable performance in more difficult tasks. FPO++ proposes two changes to the FPO objective: (1) per-sample ratios and (2) an asymmetric trust region.

Per-sample ratio. FPO estimates CFM losses by averaging over multiple (τ_i, ϵ_i) samples for each action. In the vanilla FPO algorithm [13], this produces a single ratio per action:

$$\hat{\rho}_{\text{FPO}}(\theta) = \exp\left(\frac{1}{N_{\text{mc}}} \sum_{i=1}^{N_{\text{mc}}} \left(\ell_{\theta_{\text{old}}}^{(i,t)} - \ell_{\theta}^{(i,t)}\right)\right). \tag{8}$$

An important characteristic of this formulation is that ratios are clipped *after* averaging across samples. For a given action, this means that either all or no samples are clipped. In FPO++, we instead calculate a separate ratio for each sample i,

$$\hat{\rho}_{\text{FPO++}}^{(i)}(\theta) = \exp\left(\ell_{\theta_{\text{old}}}^{(i,t)} - \ell_{\theta}^{(i,t)}\right). \tag{9}$$

The same advantage $\hat{A}_t$ is shared across samples. Equations 8 and 9 produce identical gradients for on-policy data, where all ratios evaluate to 1. When taking multiple gradient steps, however, the per-sample ratio provides a finer-grained trust region than the original per-action formulation. It allows each (τ_i, ϵ_i) pair to be clipped independently.

Asymmetric trust region (ASPO). We found that the stability of FPO when training policies from scratch can be improved by adjusting its trust region implementation. For solving these tasks in FPO++, we introduce an asymmetric trust region that we refer to as *Asymmetric SPO (ASPO)*. We use PPO clipping (Equation 1) for positive-advantage actions where gradients push to decrease CFM losses; for negative-advantage actions where gradients push to increase the CFM loss, we adopt the more constrained Simple Policy Optimization (SPO) objective proposed by [33]:

$$\psi_{\text{SPO}}\left(\rho_\theta, \hat{A}_t\right) = \rho_\theta \, \hat{A}_t \; - \; \frac{|\hat{A}_t|}{2\,\varepsilon^{\text{clip}}} \left(\rho_\theta - 1\right)^2. \tag{10}$$

Instead of zeroing out gradients for samples with ratios that surpass the trust region, the SPO objective provides a gradient signal that pulls ratios back.

Applying SPO to negative advantages disincentivizes large CFM loss increases during FPO++ updates. When interpreting the CFM loss as a variational bound [13, 34], this penalizes (i) aggressive decreases in action likelihoods and (ii) aggressive increases in the KL divergence between the sampled and learned denoising posteriors. The first property is attractive for preserving entropy [35], while the second property stabilizes the variational gap.

3.3 FPO++ Objective

We now summarize the FPO++ objective by combining the modifications described above. For each action a_t with advantage $\hat{A}_t$, we draw N_{mc} Monte Carlo pairs (τ_i, ϵ_i). The ASPO trust region combines Equations 1 and 10, defined piecewise based on the sign of the advantage:

$$\psi_{\text{ASPO}}\left(\rho_\theta, \hat{A}_t\right) = \begin{cases} \psi_{\text{PPO}}\left(\rho_\theta, \hat{A}_t\right), & \hat{A}_t \geq 0, \\ \psi_{\text{SPO}}\left(\rho_\theta, \hat{A}_t\right), & \hat{A}_t < 0. \end{cases} \tag{11}$$

The FPO++ objective is then

$$\max_{\theta} \;\; \mathbb{E}_{\pi_{\theta_{\text{old}}}} \left[\sum_{i=1}^{N_{\text{mc}}} \psi_{\text{ASPO}}\left(\hat{\rho}_{\text{FPO++}}^{(i)}(\theta), \hat{A}_t\right)\right]. \tag{12}$$

4 Experiments

This section summarizes our main findings and discussion, with additional supporting figures and analysis provided in the appendix and referenced inline where relevant.

4.1 Locomotion Benchmarks

Experiment setup. In our first set of experiments, we train policies using the standard IsaacLab [36] velocity-conditioned robot locomotion environments. We include results on four different robots:

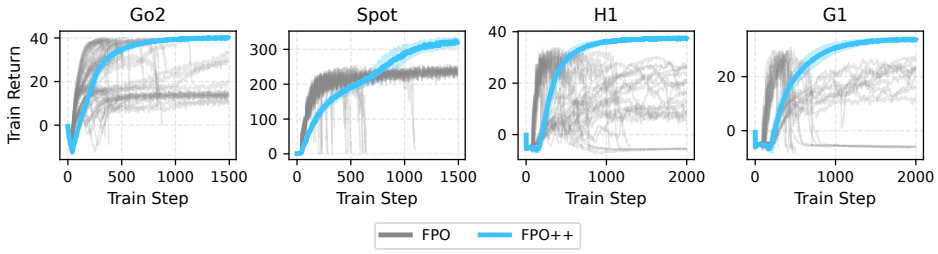
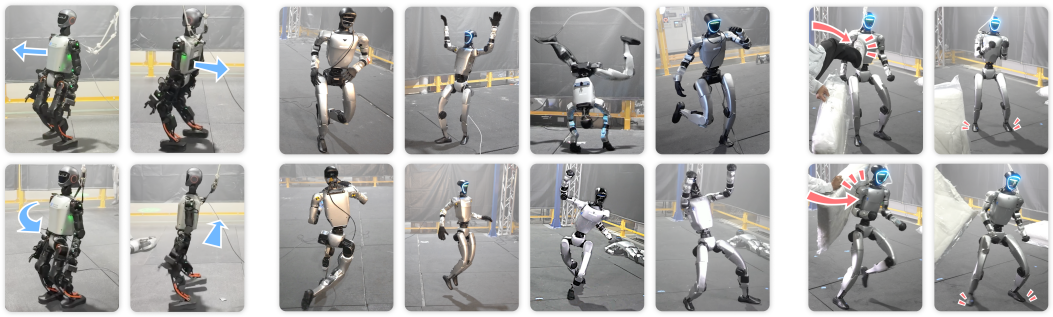


Figure 2: **Improved stability in IsaacLab locomotion environments.** We compare episode returns from FPO++ training with FPO; FPO++ curves show the mean over 5 random seeds. For thoroughness and to ensure we are not comparing against a bad set of hyperparameters for the FPO baseline, we include FPO runs for many different hyperparameter choices: all combinations of learning rate $\in \{10^{-5}, 10^{-4}, 3 \times 10^{-4}\}$, clip parameter $\in \{0.04, 0.05, 0.06\}$, and Monte Carlo samples $\in \{8, 16, 32\}$. FPO++ addresses stability problems that we were unable to solve by tuning FPO hyperparameters.



(a) T1 locomotion (b) G1 motion tracking (c) G1 robustness test

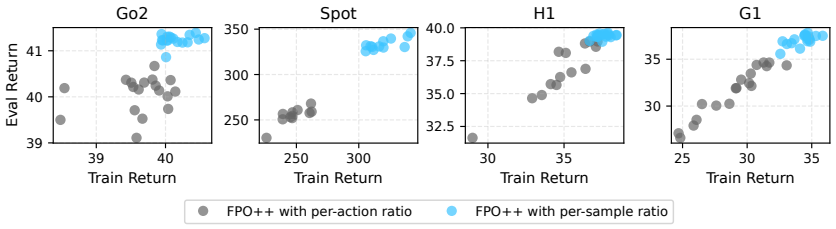
Figure 3: **Sim-to-real transfer.** We deploy flow policies for locomotion to a Booster T1 and motion tracking to a Unitree G1. Policies are directly deployed to real robots with reduced sampling step counts, demonstrating stable gaits, tracking for long sequences, and robustness to external forces. The arrows in T1 locomotion (Figure 3a) indicate velocity commands, while the arrows in the robustness tests (Figure 3c) highlight external forces.

two quadrupeds (Unitree Go2, Boston Dynamics Spot) and two humanoids (Unitree H1 and G1). Policies in these environments take proprioceptive state, linear velocity target, and angular velocity target as input, and are rewarded for matching the given velocity targets.

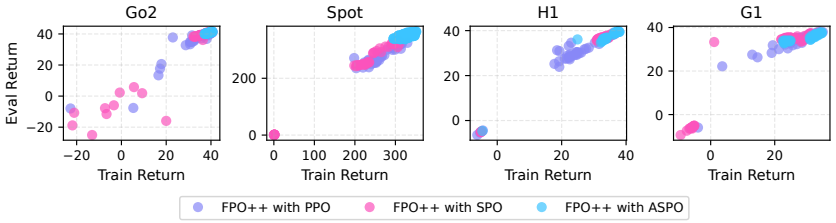
FPO++ dramatically improves stability. Figure 2 reports FPO and FPO++ training curves on each of the four robots. We found that vanilla FPO was more prone to local minima and catastrophic failures in these environments than the DeepMind Control Suite [37, 38] or PHC [39] tasks evaluated by prior work [13]. We attribute this to a combination of factors: high-dimensional action spaces, realistic joint position and torque limits, and coarser reward functions; instability persisted even with hyperparameter tuning, gradient clipping, and observation normalization. In contrast, FPO++ achieves and then maintains high episode returns.

4.2 Humanoid Sim-to-real

Experiment setup. Next, we consider sim-to-real transfer of flow policies using two humanoid robots: the Booster T1 and Unitree G1. We modify HumanoidVerse [40, 41] to train and deploy flow policies for velocity-conditioned T1 locomotion, and adapt the IsaacLab-based BeyondMimic [4] codebase to train and deploy flow policies for whole-body G1 motion tracking. Locomotion policies are conditioned on and trained to match linear and angular velocity targets; motion tracking policies



(a) Per-sample ratios improve locomotion policies.



(b) ASPO produces higher, more consistent returns.

Figure 4: **FPO++ ablations on locomotion.** We plot final training and evaluation returns across many runs; each point is a training run with clipping parameter $\in \{0.04, 0.05, 0.06\}$ and random seed $\in \{0, 1, 2, 3, 4\}$. **(a)** compares FPO++ with per-sample ratios (Equation 9) against per-action ratios from prior work (Equation 8); per-sample ratios produce higher and more consistent returns. **(b)** compares trust region implementations: standard PPO clipping, SPO [33], and our asymmetric ASPO objective.

are conditioned on and trained to match retargeted reference motions from the LAFAN dataset [42]. We use six reference motions: *dance1_subject2*, *dance1_subject1*, *walk1_subject1*, *run1_subject2*, *fight1_subject2*, and *jumps1_subject1*, which cover a wide variety of dynamic whole-body control challenges and last around 2.5 minutes each on average. We train one policy per motion and deploy each on the real robot, following the RL-based motion tracking protocol of BeyondMimic [4].

FPO++ policies succeed on physical robots. FPO++ learns robust gaits for T1 locomotion and reliably tracks dynamic motion sequences on the G1, deployed as shown in Figure 3. We consider this a significant result: to our knowledge, it is the first demonstration of humanoid sim-to-real using either (i) a flow policy trained without expert distillation or (ii) a policy gradient technique without explicit likelihoods.

4.3 Manipulation Finetuning

Experiment setup. One application not studied in prior flow policy gradient work [13] is reward-based finetuning for policies that are first trained from demonstrations. The success of flow policies in imitation learning [10] makes RL of flow policies uniquely important in this setting. To validate FPO++ for finetuning, we begin by pretraining image-based manipulation policies for tasks from RoboMimic [24] and DexMimicGen [43]. We follow the data processing of ResFiT [44] for five tasks, which cover diverse embodiments and control regimes (Figure A.2). All policies use a 3-layer MLP backbone with a ViT [45] encoder for image observations. We then finetune flow policies using FPO++, FPO, and two baselines adapted from the DPPO [7] implementation in [13]: one with a fixed noise scale for exploration and one with a predicted noise inspired by ReinFlow [8] (shown as *DPPO – fixed noise* and *DPPO – learned noise* in Figure 5). Implementation and baseline details are in Appendix C.4; notably, we disable ASPO for manipulation experiments (Section 4.4).

Flow policy gradients succeed in finetuning. We plot policy success rates using each finetuning algorithm in Figure 5. We find that FPO++ consistently achieves high success rates, converging more rapidly than baselines. Vanilla FPO also performs well on simpler manipulation tasks, suggesting

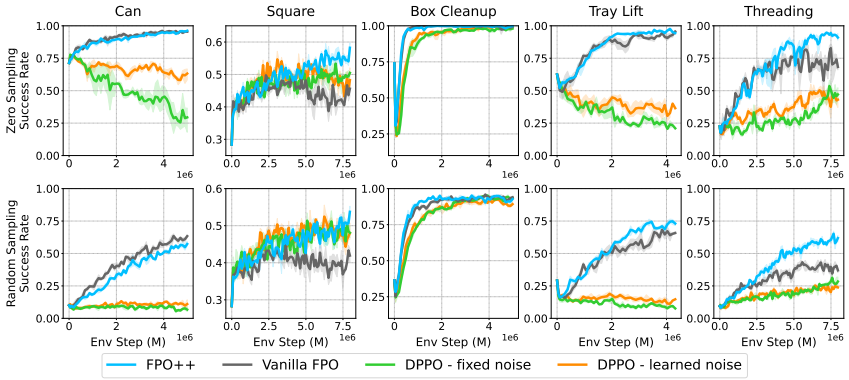


Figure 5: **Manipulation finetuning results.** We compare the evaluation success rates for FPO++, FPO, and our two DPPO implementations. We show two rows: the first row contains policy success rates with zero-sampling ($\epsilon = \vec{0}$), while the second row contains policy success rates using standard random sampling ($\epsilon \sim \mathcal{N}(0, I)$). For each task, all algorithms are initialized from the same base policy, which is an image-based flow matching policy trained to predict action chunks.

that behavior cloning initialization can provide sufficient regularization for mitigating the instabilities we observed when training from scratch. Both DPPO variants underperform FPO-based methods, which we attribute to the longer effective MDP horizon introduced by treating diffusion steps as decision points [30]. These results are notable given the emphasis on likelihood computation in prior flow and diffusion finetuning methods (e.g., two-layer MDPs [7] and learned noise predictors [8]); FPO++ suggests the explicit density estimates motivating these changes are not necessary for effective flow RL training.

FPO++ is more robust to base-policy quality. We found that our DPPO runs often failed when base policies had low random sampling success rates, like the Can example in Figure 5, which starts at $\sim 10\%$ success rate. The same task succeeds when initialized with higher-quality base policies; examples are provided in Appendix C.5.

4.4 Training Ablations

The experiments above demonstrate stable flow policy learning across locomotion, sim-to-real, and manipulation tasks. To better understand what makes this possible, we ablate FPO++'s per-sample ratio and ASPO trust region.

FPO++ changes are critical for all locomotion embodiments. Figure 4a shows final training and evaluation returns for locomotion environments when we ablate the per-sample ratio; Figure 4b shows returns when we ablate the ASPO trust region. We include multiple hyperparameter configurations for thoroughness: each point is a run trained using a unique clipping parameter and random seed combination. We find that FPO++ achieves consistent policy performance across hyperparameters and random seeds. When either the per-sample ratio or ASPO trust region is disabled, average returns drop significantly across robot embodiments. The variance of returns also increases, suggesting that the ratio and ASPO trust region combination stabilizes learning.

ASPO successfully preserves entropy. We compare flow field visualizations for policies trained using PPO and ASPO trust regions in Appendix D.2 (Figure A.13). Policies trained using ASPO successfully solve tasks without entropy collapse, explaining the improved robustness in Figure 4b.

FPO++ reduces empirical gradient variance. A motivation for FPO++'s changes is reduced gradient variance, which we verify using a cosine similarity metric inspired by [46] (Appendix D.1).

ASPO's benefit depends on the reward regime. ASPO improves locomotion but underperforms PPO in manipulation finetuning (Appendix C.6). To test whether this stems from pretraining, we compare both trust regions in FPO++ on RoboMimic Lift with randomly initialized action networks

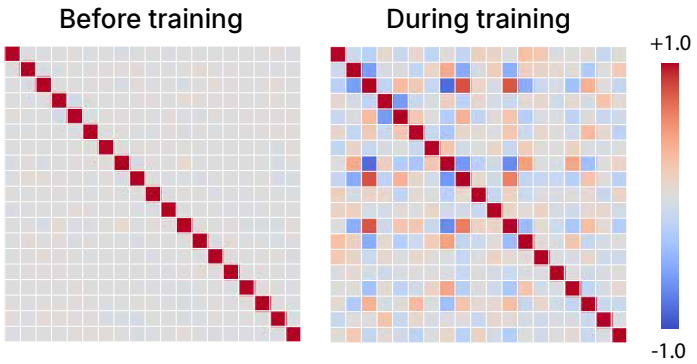


Figure 6: **Correlations before and during training.** We sample 10,000 actions conditioned on the same observation, and visualize the correlations between action dimensions before and during FPO++ training for an H1 locomotion policy.

(Appendix C.7, Figure A.11). With sparse rewards, ASPO barely learns, reaching 0.07 final success versus PPO's 0.73. This suggests that limiting likelihood decreases for negative-advantage actions keeps the policy near its random initialization and hinders bootstrapping from rare successes. With dense rewards, ASPO initially learns but later loses success despite relatively high shaped return (0.28 final success versus PPO's 0.81). Rollouts reveal grasp-and-hover behavior that earns shaping reward without completing the lift. These results suggest that ASPO's effectiveness depends on the reward regime, not just policy initialization. We recommend ASPO for dense rewards closely aligned with task completion, as in locomotion and motion tracking, PPO for sparse rewards, and caution with ASPO when shaping rewards can be exploited.

4.5 Comparison with Gaussian PPO

While some FPO++ use cases (e.g., the finetuning in Section 4.3) require flow policies, other tasks are designed for Gaussian PPO—a well-understood baseline for validating FPO++ training, analyzing dynamics, and assessing sample efficiency and expressiveness. We investigate these characteristics by first comparing FPO++ against IsaacLab's default Gaussian PPO implementation [47] across locomotion tasks with varying amounts of parallelization (Figure A.5). Details and hyperparameter tuning information can be found in Appendix B.1.

FPO++ locomotion policies show sample efficiency advantages. We observe in Figure A.5 that FPO++ locomotion policies for each environment count almost always converge to higher returns than the Gaussian PPO configurations we compare against, with reduced variance between random seeds. In particular, we see improved robustness to very small batch sizes in Go2, H1, and G1, and better exploitation of increased parallelism for Spot.

FPO++ policies explore with more expressive action distributions. Consistent with [48], we found that the same rewards often produced different gaits with different algorithms. FPO++ policies trained with default Spot locomotion rewards, for example, produced more consistent "trot" gaits than Gaussian PPO policies, which tended toward more symmetric "pronk" gaits (Figure A.6). To examine dependencies between action dimensions, Figure 6 visualizes correlations for an H1 locomotion policy. Off-diagonal correlations are near zero before training but develop structured positive and negative patterns during training, including negative correlations between the left and right hip joints. Because actions are sampled at a fixed observation, these patterns reflect dependencies within the policy's conditional action distribution rather than changes in robot state along a rollout. They show that FPO++ learns to vary joint commands together or in opposition during stochastic exploration, unlike diagonal-Gaussian PPO policies whose action dimensions are conditionally independent.

Limitations and future work. FPO++ provides a single algorithm and policy representation that succeeds across embodiments and tasks, for both training from scratch and finetuning, but it is not fully turnkey and some challenges remain. The benefit of ASPO depends on the reward regime rather than initialization alone (Section 4.4); choosing the right configuration still requires task-specific knowledge. Gaussian PPO with entropy regularization and an adaptive learning rate also achieves higher motion-tracking returns (Appendix C.3). FPO++ also takes more wall-clock time than Gaussian PPO, limiting its appeal where Gaussian policies already suffice: for G1 locomotion on an L40S GPU, our Gaussian PPO baseline reaches an evaluation return of 25 in 19 minutes, versus 23 minutes for FPO++, and motion tracking can take up to $2\times$ longer than a tuned Gaussian PPO baseline. Reducing this gap, for example through few-step distillation [49, 50] for more efficient training and inference, is promising future work. Other directions include applications where Gaussian policies do not apply, such as tasks requiring more expressive exploration or diffusion-based sequence modeling [27, 51].

5 Conclusion

We presented a study of robot control with flow matching policy gradients. The proposed FPO++ algorithm is more stable than prior flow policy gradient implementations and succeeds on practical locomotion, motion tracking, and manipulation finetuning tasks. Our analysis highlights the effect of per-sample ratios and an asymmetric trust region, and shows how FPO++ exploits the flow representation: learning correlated action distributions and distinct gaits, with improved sample efficiency and finetuning robustness over baselines.

Beyond validating a specific algorithm on specific tasks, FPO++ aims to serve as an existence proof that we are excited about for several reasons. FPO++ validates several possibilities, including policy gradient-style training of flow policies for real-world continuous control and sim-to-real transfer of these policies after training using only RL. Importantly, these results challenge the common assumption that explicit likelihoods are needed for policy gradient methods in robot control. By demonstrating that this constraint can be bypassed, FPO++ suggests new directions for expanding the design space of RL algorithms for future robot learning systems.

Acknowledgments

We thank David McAllister, Arthur Allshire, and Chungmin Kim for helpful discussions.

References

[1] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 1999.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[3] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 91–100. PMLR, 2022. URL https://proceedings.mlr.press/v164/rudin22a.html.

[4] Q. Liao, T. E. Truong, X. Huang, Y. Gao, G. Tevet, K. Sreenath, and C. K. Liu. BeyondMimic: From motion tracking to versatile humanoid control via guided diffusion. *Science Robotics*, 11(117):eadx8924, 2026.

[5] A. Allshire, H. Choi, J. Zhang, D. McAllister, A. Zhang, C. M. Kim, T. Darrell, P. Abbeel, J. Malik, and A. Kanazawa. Visual imitation enables contextual humanoid control. In *Conference on Robot Learning (CoRL)*, 2025.

[6] H. Qi, B. Yi, S. Suresh, M. Lambeta, Y. Ma, R. Calandra, and J. Malik. General in-hand object rotation with vision and touch. In *Conference on Robot Learning*, pages 2549–2564. PMLR, 2023.

[7] A. Z. Ren, J. Lidard, L. L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.

[8] T. Zhang, C. Yu, S. Su, and Y. Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *arXiv preprint arXiv:2505.22094*, 2025.

[9] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2025.

[10] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

[11] M. Skreta, L. Atanackovic, A. J. Bose, A. Tong, and K. Neklyudov. The superposition of diffusion models using the itô density estimator, 2025. URL https://arxiv.org/abs/2412.17762.

[12] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[13] D. McAllister, S. Ge, B. Yi, C. M. Kim, E. Weber, H. Choi, H. Feng, and A. Kanazawa. Flow matching policy gradients. *arXiv preprint arXiv:2507.21053*, 2025.

[14] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020. doi:10.1126/scirobotics.abc5986.

[15] H. Qi, B. Yi, M. Lambeta, Y. Ma, R. Calandra, and J. Malik. From simple to complex skills: The case of in-hand object reorientation. *arXiv preprint arXiv:2501.05439*, 2025.

[16] D. Hafner, W. Yan, and T. Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.

[17] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[18] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[19] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[20] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

[21] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 2020.

[22] R. Gao, E. Hoogeboom, J. Heek, V. D. Bortoli, K. P. Murphy, and T. Salimans. Diffusion meets flow matching: Two sides of the same coin. 2024. URL https://diffusionflow.github.io/.

[23] L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal. From imitation to refinement–residual rl for precise visual assembly. *arXiv preprint arXiv:2407.16677*, 2024.

[24] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[25] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[26] S. Jiang, X. Fang, N. Roy, T. Lozano-Pérez, L. P. Kaelbling, and S. Ancha. Streaming flow policy: Simplifying diffusion / flow-matching policies by treating action trajectories as flow trajectories. *arXiv preprint arXiv:2505.21851*, 2025.

[27] X. Huang, T. Truong, Y. Zhang, F. Yu, J. P. Sleiman, J. Hodgins, K. Sreenath, and F. Farshidian. Diffuse-cloc: Guided diffusion for physics-based character look-ahead control. *ACM Transactions on Graphics (TOG)*, 44(4):1–12, 2025.

[28] K. Black, M. Janner, Y. Du, I. Kostrikov, and S. Levine. Training diffusion models with reinforcement learning. *arXiv preprint arXiv:2305.13301*, 2023.

[29] Y. Fan, O. Watkins, Y. Du, H. Liu, M. Ryu, C. Boutilier, P. Abbeel, M. Ghavamzadeh, K. Lee, and K. Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36:79858–79885, 2023.

[30] N. Yang, J. Gao, F. Gao, Y. Wu, and C. Yu. Fine-tuning diffusion policies with backpropagation through diffusion timesteps. *arXiv preprint arXiv:2505.10482*, 2025.

[31] S. Ding, K. Hu, S. Zhong, H. Luo, W. Zhang, J. Wang, J. Wang, and Y. Shi. Genpo: Generative diffusion models meet on-policy reinforcement learning. *arXiv preprint arXiv:2505.18763*, 2025.

[32] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[33] Z. Xie, Q. Zhang, F. Yang, M. Hutter, and R. Xu. Simple policy optimization. *arXiv preprint arXiv:2401.16025*, 2024. URL https://arxiv.org/abs/2401.16025. v9, revised 26 Jul 2025.

[34] D. P. Kingma and R. Gao. Understanding diffusion objectives as the elbo with simple data augmentation, 2023. URL https://arxiv.org/abs/2303.00848.

[35] Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, T. Fan, G. Liu, L. Liu, X. Liu, H. Lin, Z. Lin, B. Ma, G. Sheng, Y. Tong, C. Zhang, M. Zhang, W. Zhang, H. Zhu, J. Zhu, J. Chen, J. Chen, C. Wang, H. Yu, W. Dai, Y. Song, X. Wei, H. Zhou, J. Liu, W.-Y. Ma, Y.-Q. Zhang, L. Yan, M. Qiao, Y. Wu, and M. Wang. Dapo: An open-source llm reinforcement learning system at scale. 2025. URL https://arxiv.org/abs/2503.14476. version v2, submitted March 2025.

[36] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi:10.1109/LRA.2023.3270034.

[37] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[38] K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs, et al. Mujoco playground. *arXiv preprint arXiv:2502.08844*, 2025.

[39] Z. Luo, J. Cao, K. Kitani, W. Xu, et al. Perpetual humanoid control for real-time simulated avatars. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10895–10904, 2023.

[40] LeCAR-Lab. Humanoidverse: A versatile and extendable reinforcement learning framework for humanoid robots. https://github.com/LeCAR-Lab/HumanoidVerse, 2023. Accessed: 2025-09-22.

[41] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[42] F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal. Robust motion in-betweening. *ACM Transactions on Graphics (TOG)*, 39(4):60–1, 2020.

[43] Z. Jiang, Y. Xie, K. Lin, Z. Xu, W. Wan, A. Mandlekar, L. J. Fan, and Y. Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16923–16930. IEEE, 2025.

[44] L. Ankile, Z. Jiang, R. Duan, G. Shi, P. Abbeel, and A. Nagabandi. Residual off-policy rl for finetuning behavior cloning policies. *arXiv preprint arXiv:2509.19301*, 2025.

[45] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[46] A. Ilyas, L. Engstrom, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry. A closer look at deep policy gradients. *arXiv preprint arXiv:1811.02553*, 2018.

[47] C. Schwarke, M. Mittal, N. Rudin, D. Hoeller, and M. Hutter. Rsl-rl: A learning library for robotics research. *arXiv preprint arXiv:2509.10771*, 2025.

[48] Y. Seo, C. Sferrazza, H. Geng, M. Nauman, Z.-H. Yin, and P. Abbeel. Fasttd3: Simple, fast, and capable reinforcement learning for humanoid control. *arXiv preprint arXiv:2505.22642*, 2025.

[49] T. Salimans and J. Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[50] X. Liu, C. Gong, and Q. Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[51] B. Chen, D. M. Monso, Y. Du, M. Simchowitz, R. Tedrake, and V. Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *arXiv preprint arXiv:2407.01392*, 2024.

[52] X. B. Peng, A. Kumar, G. Zhang, and S. Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[53] B. Kang, X. Ma, C. Du, T. Pang, and S. Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 2024.

[54] S. Ding, K. Hu, Z. Zhang, K. Ren, W. Zhang, J. Yu, J. Wang, and Y. Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *Advances in Neural Information Processing Systems*, 37:53945–53968, 2024.

[55] S. Zhang, W. Zhang, and Q. Gu. Energy-weighted flow matching for offline reinforcement learning. *arXiv preprint arXiv:2503.04975*, 2025.

[56] Z. Wang, J. J. Hunt, and M. Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

[57] C. Lu, H. Chen, J. Chen, H. Su, C. Li, and J. Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning*, pages 22825–22855. PMLR, 2023.

[58] L. He, L. Shen, L. Zhang, J. Tan, and X. Wang. Diffcps: Diffusion model based constrained policy search for offline reinforcement learning. *arXiv preprint arXiv:2310.05333*, 2023.

[59] Z. Ding and C. Jin. Consistency models as a rich and efficient policy class for reinforcement learning. *arXiv preprint arXiv:2309.16984*, 2023.

[60] R. Zhang, Z. Luo, J. Sjölund, T. Schön, and P. Mattsson. Entropy-regularized diffusion policy with q-ensembles for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 37:98871–98897, 2024.

[61] S. E. Ada, E. Oztop, and E. Ugur. Diffusion policies for out-of-distribution generalization in offline reinforcement learning. *IEEE Robotics and Automation Letters*, 9(4):3116–3123, 2024.

[62] O. Celik, Z. Li, D. Blessing, G. Li, D. Palenicek, J. Peters, G. Chalvatzaki, and G. Neumann. DIME: Diffusion-based maximum entropy reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2025. URL https://openreview.net/forum?id=Aw6dBR7Vxj.

[63] S. Park, Q. Li, and S. Levine. Flow q-learning. *arXiv preprint arXiv:2502.02538*, 2025.

[64] M. Psenka, A. Escontrela, P. Abbeel, and Y. Ma. Learning a diffusion model policy from rewards via q-score matching. *arXiv preprint arXiv:2312.11752*, 2023.

[65] Q. Li and S. Levine. Q-learning with adjoint matching. *arXiv preprint arXiv:2601.14234*, 2026.

[66] A. Havens, B. K. Miller, B. Yan, C. Domingo-Enrich, A. Sriram, B. Wood, D. Levine, B. Hu, B. Amos, B. Karrer, et al. Adjoint sampling: Highly scalable diffusion samplers via adjoint matching. *arXiv preprint arXiv:2504.11713*, 2025.

[67] B. Yi, C. M. Kim, J. Kerr, G. Wu, R. Feng, A. Zhang, J. Kulhanek, H. Choi, Y. Ma, M. Tancik, and A. Kanazawa. Viser: Imperative, web-based 3d visualization in python. *arXiv preprint arXiv:2507.22885*, 2025.

[68] C. Pan, G. Anantharaman, N.-C. Huang, C. Jin, D. Pfrommer, C. Yuan, F. Permenter, G. Qu, N. Boffi, G. Shi, et al. Much ado about noising: Dispelling the myths of generative robotic control. *arXiv preprint arXiv:2512.01809*, 2025.

Appendix of "Flow Policy Gradients for Robot Control"

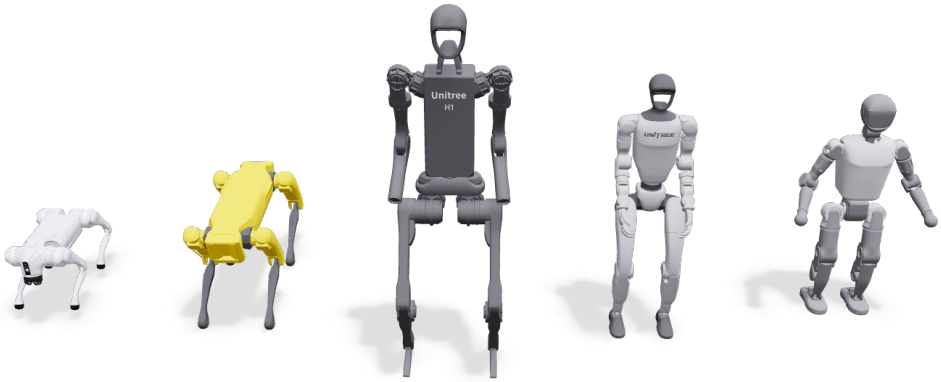
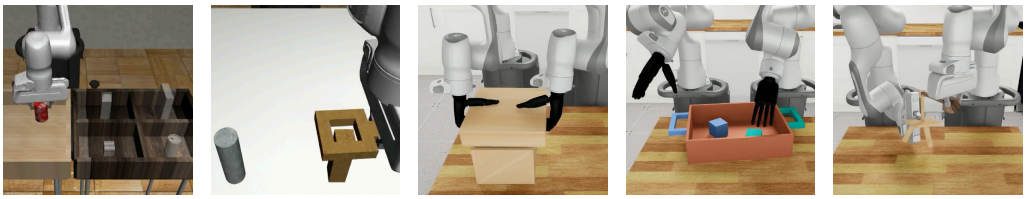


Figure A.1: **Legged robots used for experiments.** We train policies for the Go2, Spot, H1, and G1 robots in simulation. We deploy policies to physical G1 and Booster T1 robots.



(a) Can (b) Square (c) Box Cleanup (d) Tray Lift (e) Threading

RoboMimic DexMimicGen

Figure A.2: **Manipulation tasks.** We compare FPO++ to baseline methods on manipulation tasks from RoboMimic [24] and DexMimicGen [43]. We choose tasks that cover diverse embodiments and control regimes: from single-arm to bimanual manipulation, parallel-jaw grippers and dexterous hands, and both short- and long-horizon tasks.

A Further related work

The main body of our paper discusses online RL, where policies are updated from their own experience. A related body of work has explored using flow and diffusion-based policies in offline RL. One common strategy is advantage weighted regression (AWR) [52, 53, 54, 55]. Another line of work optimizes a Q-learning objective jointly with a generative-model loss [56, 57, 58, 59, 60, 61], enabling value-based training while regularizing the policy through diffusion or flow matching. Maximum-entropy approaches such as DIME [62] extend this direction by integrating diffusion policies with entropy-regularized RL, further improving robustness and sample quality in offline settings.

A key challenge in training diffusion or flow policies with Q-learning is that backpropagation through the multi-step denoising process is numerically unstable. FQL [63] addresses this by training a one-step flow policy without backpropagation through time (BPTT). Q-score matching [64] links the score of the diffusion policy to the action gradient of the Q-function. Finally, QAM [65] leverages adjoint matching [66] to transform the critic's action gradient into a step-wise objective for the policy.

B Experiment Details

In this section, we detail the experimental setup for the IsaacLab velocity-conditioned locomotion benchmarks, motion tracking benchmarks, and manipulation finetuning benchmarks used throughout the paper. The robots we used in the paper, including quadrupeds, humanoids, and manipulators, are illustrated in Figures A.1 and A.2.

B.1 Locomotion Benchmarking

B.1.1 Hyperparameters

All policies for the locomotion benchmarking experiments utilize 3-layer Multi-Layer Perceptrons (MLPs). Specifically, the actor network employs 256 hidden units, and the critic network employs 768 hidden units per layer. Following the IsaacLab defaults, we train with 4096 parallel environments and take 24 environment steps between policy updates, and we use 64 Euler steps for all rollouts. Lower step counts (8, 16) reduced training stability without significantly impacting runtime: policy inference is fast, requires no gradients (even for training rollouts), and is not a training bottleneck. Additional hyperparameters can be found in Table A.1.

Quadruped policies are trained for 1500 steps, and humanoid policies are trained for 2000 steps. For the Gaussian PPO results, we adopt the default hyperparameters provided by the `rsl_rl` library, with additional sweeps performed over the clipping parameters in the set $\{0.1, 0.15, 0.2, 0.25\}$.

B.1.2 Environments and Rewards

All locomotion benchmarking experiments use the standard IsaacLab velocity-conditioned locomotion environments with default reward weights. The following reward terms are shared across all robot configurations:

- **Linear velocity tracking**: Exponential reward for matching the commanded linear velocity in the horizontal plane.
- **Angular velocity tracking**: Exponential reward for matching the commanded yaw rate.
- **Feet air time**: Reward for keeping feet in the air for a sufficient duration during each step.
- **Angular velocity (xy)**: Penalty for roll and pitch angular velocities.
- **Joint accelerations**: Penalty for large joint accelerations.
- **Action rate**: Penalty for rapid changes in actions between consecutive timesteps.

The default quadruped environments (Go2, Spot) use less reward shaping, relying on velocity tracking and a weak air time reward (weight: 0.125) to allow gaits to emerge naturally. These environments also include penalties for vertical base velocity, joint torques, and undesired contacts on non-foot body parts (e.g., thighs).

The Spot quadruped environment introduces several additional terms: a gait synchronization reward that encourages diagonal foot pair coordination for trotting, a foot clearance reward for achieving target swing height, an air time variance penalty for consistent swing timing across legs, and explicit foot slip and body orientation penalties.

The humanoid environments (H1, G1) include biped-specific adaptations: a modified air time reward for two-legged locomotion, joint deviation penalties that regularize hip, arm, and torso positions toward default poses, feet slide penalties, and a large termination penalty to discourage falling. The G1 configuration uses higher angular velocity tracking weight and enables joint torque penalties, while H1 disables both torque penalties and vertical velocity penalties. Neither humanoid environment includes gait synchronization rewards.

For more details, we refer to the open-source IsaacLab [36] repository.

Hyperparameter category	Used value	Sweep range and notes
FPO++		
Flow integration steps	64	$\{8, 16, 32, 64\}$
Network output	u	$\{u, x_0\}$, x_0 denotes data
Samples per action	16	$\{8, 16, 32\}$
Training		
Learning rate	1×10^{-4}	1×10^{-5}, 1×10^{-4}, 3×10^{-4}
Weight decay / Adam betas	1×10^{-4} / $(0.9, 0.95)$	AdamW parameters
Clip parameter	0.05	$\{0.03, 0.04, 0.05, 0.06\}$
Discount factor (γ)	0.99	
GAE lambda (λ)	0.95	
Learning epochs	16 for Go2, others 32	
Minibatches per update	4	
Running observation normalization	Yes	

Table A.1: **FPO++ and training hyperparameters used for locomotion.** We report both the final values we use for experiments and the values we performed sweeps over.

B.2 Motion Tracking

Our motion tracking experiments train a policy for the Unitree G1 robot, which has 29 degrees of freedom (DoF). The control frequency is set to 50 Hz (with a simulation timestep $\Delta t = 0.005$ s and decimation of 4). The reward design, observation space, and termination conditions follow BeyondMimic [4]. The following sections detail the specific hyperparameters used.

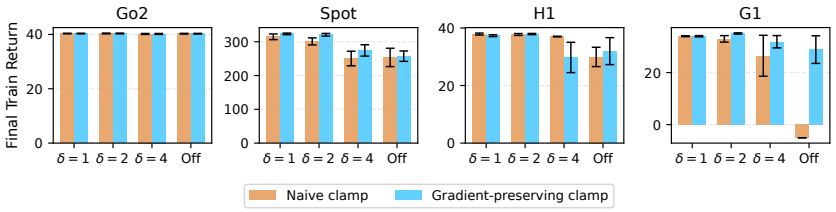
Hyperparameter category	Used value	Sweep range and notes
FPO++		
Flow integration steps	50	$\{10, 50\}$
Network output	u	$\{u, x_0\}$, x_0 denotes data
Samples per action	16	$\{8, 16, 32\}$
Training		
Learning rate	3×10^{-4}	
Weight decay / Adam betas	1×10^{-4} / $(0.9, 0.95)$	AdamW parameters
Clip parameter	0.01	$\{0.01, 0.05, 0.1\}$
Discount factor (γ)	0.99	
GAE lambda (λ)	0.95	
Learning epochs	5	
Minibatches per update	4	
Running observation normalization	Yes	

Table A.2: **FPO++ and training hyperparameters used for motion tracking.** We report both the final values we use for experiments and the values we performed sweeps over.

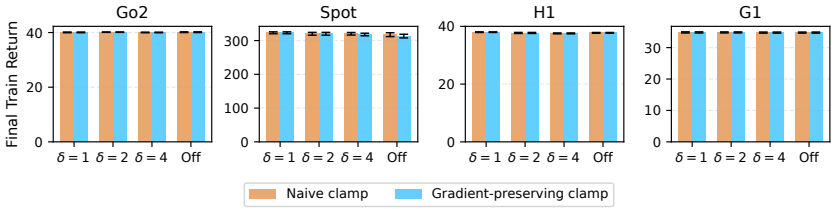
B.2.1 Domain Randomization

Domain randomization is applied to improve the policy's sim-to-real transfer ability:

- **Physics Material**: Static friction (Uniform $[0.3, 1.6]$), Dynamic friction (Uniform $[0.3, 1.2]$), and Restitution (Uniform $[0.0, 0.5]$) are sampled at startup.
- **Joint Defaults**: Default joint angles are uniformly offset by $\mathcal{U}(-0.01, 0.01)$ rad at startup.
- **Center of Mass (COM)**: The torso COM is offset uniformly in (x, y, z) at startup.



(a) Without CFM loss clamp.



(b) With CFM loss clamp.

Figure A.3: **Clamping CFM losses improves stability.** The x-axis varies the Huber threshold δ; "Off" uses squared CFM loss without Huber. Loss-difference clamping is enabled throughout: orange uses naive clamping, and blue preserves gradients through the clamp. Panels (a) and (b) respectively disable and enable clamping of the old- and new-policy CFM losses before subtraction. Bars show mean final training return with standard error over five seeds.

- **External Forces**: Pushes are applied at regular intervals of 2.0 to 3.0 seconds. Each push applies a random linear velocity between -0.5 and 0.5 meters per second along the forward and sideways directions, and between -0.2 and 0.2 meters per second in the vertical direction. Angular velocity is randomized between -0.52 and 0.52 radians per second for pitch and roll, and between -0.78 and 0.78 radians per second for yaw.
- **Actuator Command Delay**: Actuator latency is simulated by applying a random delay of 0 to 2 simulation steps (corresponding to 0 ms to 10 ms at $\Delta t = 0.005$ s) to the control commands (joint positions, velocities, efforts) at each environment reset to improve robustness and sim-to-real transfer.
- **Motion Initialization**: Root position, orientation, and joint angle offsets are applied uniformly at episode reset.

B.2.2 Training Hyperparameters

Training is conducted across 4096 parallel environments, with a rollout length of 96 steps per environment. Both the actor and critic networks use 3-layer MLPs with hidden unit sizes (1024, 512, 256). We will release code for further details. The Booster T1 locomotion policy uses the same hyperparameters as the IsaacLab locomotion experiments. We train motion tracking policies with 50 flow integration steps; at deployment, we use zero-sampling with 5 flow integration steps to reduce latency. Additional hyperparameters can be found in Table A.2.

B.2.3 CFM loss implementation details

The FPO and FPO++ ratios (Equation 3) both exponentiate a difference between squared CFM losses. An early concern we had was that exponentiating a value computed from squares could cause numerical instabilities for outlier epsilon values. We ran experiments with details like Huber instead of squared CFM losses, as well as by applying both naive and gradient-preserving clamp operations to CFM loss differences (Figure A.3). We found that a simple approach of (i) clamping

CFM losses before taking differences and (ii) then clamping the difference before exponentiation was sufficient for stabilizing training.

B.3 Manipulation finetuning

We use actor learning rates of 1×10^{-5} and critic learning rates of 1×10^{-4} in the final manipulation results. The generalized advantage estimation (GAE) parameter is fixed to $\lambda = 0.99$ across all five manipulation tasks. The discount factor γ varies by task: $\gamma = 0.99$ for *RoboMimic Can*, $\gamma = 0.995$ for *RoboMimic Square* and *DexMimicGen Box Cleanup*, and $\gamma = 0.999$ for *DexMimicGen Tray Lift* and *Threading*. These hyperparameters are shared across vanilla FPO and DPPO variants in Figure 5. For both FPO++ and vanilla FPO, we use 10 flow sampling steps and fix the number of Monte Carlo samples per action chunk to 8 across all manipulation tasks. Policies sample action chunks with horizon length 16, and during finetuning we compute chunk-level ratios by summing CFM losses across all chunk timesteps.

For our comparisons between FPO++, vanilla FPO, and two DPPO [7] variants in Figure 5, we adapt DPPO implementations from [13]. Some implementation details differ from the original DPPO implementation, for example, adopting velocity prediction instead of epsilon prediction [21]. This allowed comparisons between methods to be initialized from the same base policy and evaluated under more similar conditions. We extensively tuned baselines for fairness, sweeping over hyperparameters including PPO clipping thresholds, gradient norm clipping, and exploration noise scales. We present additional baselines in Appendix C.4.

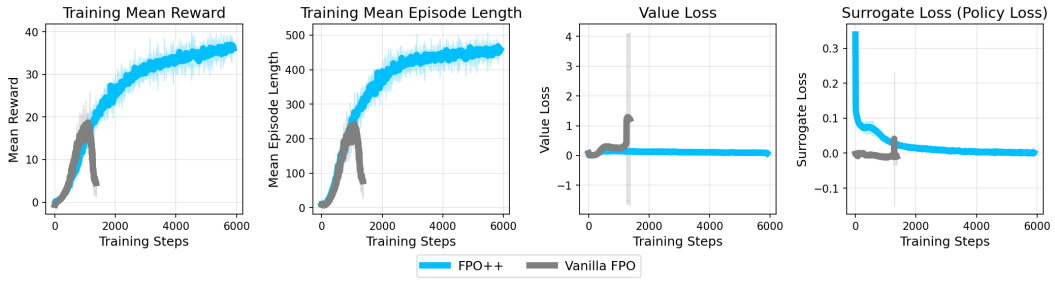


Figure A.4: **Motion tracking training curves.** Vanilla FPO fails in learning to track the motion, while FPO++ trains stably. Hyperparameters are shared between both runs.

C More experiments

C.1 Sample efficiency and emergent gaits versus Gaussian PPO

Figure A.5 compares policy returns between Gaussian PPO and FPO++ across locomotion tasks as we vary the number of parallel environments. Figure A.6 shows the qualitatively different gaits that the two algorithms converge to under identical rewards.

C.2 Comparison with vanilla FPO for motion tracking

Figure A.4 shows the training curves for the motion-tracking policy that we deploy on the real G1 robot. As the plots illustrate, vanilla FPO initially learns but quickly collapses: the mean reward and episode length peak early and then deteriorate, while both the value loss and surrogate loss exhibit large spikes, indicating numerical instability. In contrast, FPO++ maintains stable value and policy losses throughout, continues improving monotonically, and successfully converges to the high-return policy used for real-world deployment. These curves provide evidence for a core motivation of our work: vanilla FPO is unstable on realistic, high-DoF robot control tasks, whereas the algorithmic components of FPO++ prevent collapse and enable successful sim-to-real transfer. We will release training code for reproducibility.

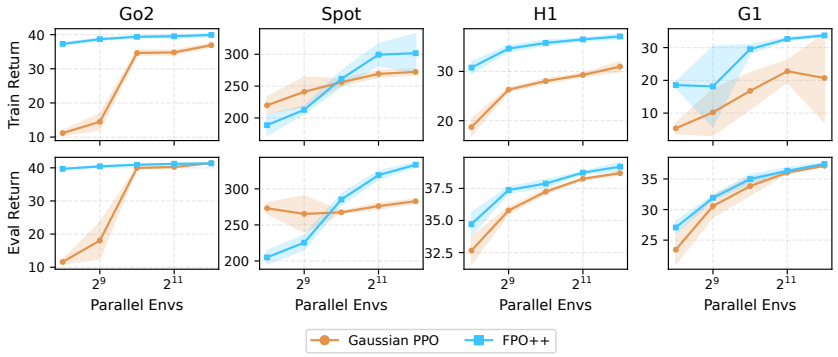


Figure A.5: **Sample efficiency for FPO++ and our Gaussian PPO baseline.** We compare policy returns between Gaussian PPO and FPO++ with environment counts $\in \{2^8, 2^9, 2^{10}, 2^{11}, 2^{12}\}$ after 1500 updates for quadrupeds and 2000 updates for humanoids. Minibatch size per policy update increases linearly with the number of parallel environments; other hyperparameters are fixed between runs. Filled regions show standard deviation computed over 5 seeds.

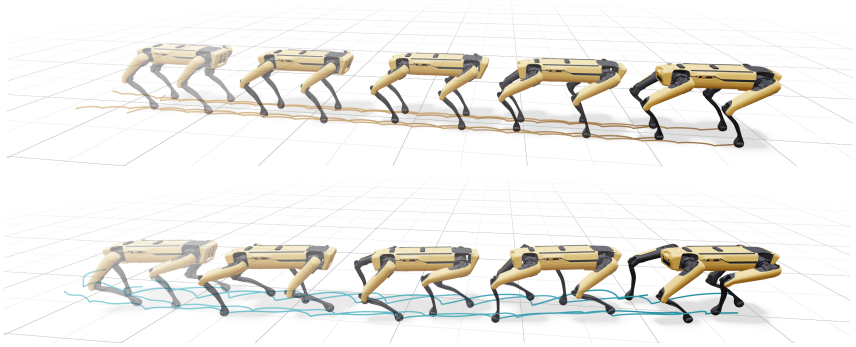


Figure A.6: **Different gaits from different algorithms.** Rollouts using Gaussian PPO (top) and FPO++ (bottom) for 1500 policy updates. Policies are trained with the same rewards, 4096 parallel environments, and given the same forward velocity command (1 m/s). We found differences reproducible across seeds, clip parameters, and learning rates. We use Viser [67] for visualization.

C.3 Comparison with Gaussian PPO for motion tracking

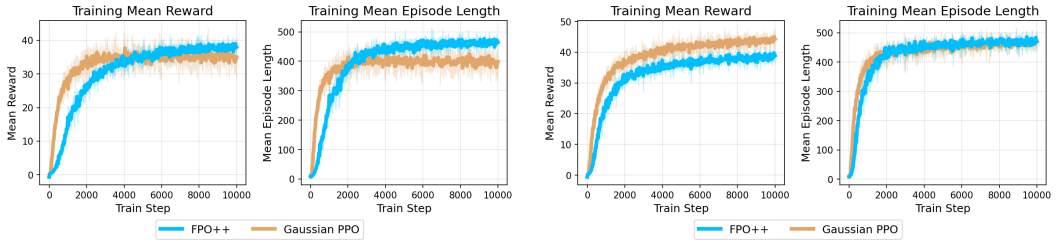
FPO++ enables successful sim-to-real transfer for high-DoF motion tracking, but this does not imply higher simulation returns than tuned Gaussian PPO. Figure A.7 compares the two algorithms with and without entropy regularization and adaptive learning rates. Without these components, FPO++ achieves higher final returns and longer episodes than the simplified Gaussian PPO baseline. With these components enabled for both algorithms, Gaussian PPO achieves higher returns, while FPO++ attains similar or slightly longer episodes.

For this comparison, we augment FPO++ with simple entropy regularization and adaptive learning-rate mechanisms. These are experimental variants rather than changes to the core FPO++ objective.

Entropy regularization. We use a non-parametric Kozachenko–Leonenko estimator based on nearest-neighbor distances between action predictions. This estimates local density using pairwise L_2 distances and supplies an entropy bonus to encourage exploration.

Adaptive learning rate. The FPO++ variant uses an L_2-based noise-prediction-change proxy to adapt the learning rate. Noise predictions are recovered algebraically from velocity predictions and sampled actions; the resulting statistic measures policy change without explicitly computing action likelihoods. It is a heuristic proxy, not an exact policy KL divergence.

These components improve FPO++ training in this comparison but do not close the return gap to tuned Gaussian PPO. More effective regularization and learning-rate adaptation for flow policies remain directions for future work.



(a) Without entropy regularization or adaptive learning rate.

(b) With entropy regularization and adaptive learning rate.

Figure A.7: **Training curves compared to Gaussian PPO for motion tracking.** Blue denotes FPO++ and orange Gaussian PPO. Without entropy regularization or adaptive learning rates, FPO++ achieves higher final returns and longer episodes. With both components enabled, Gaussian PPO attains higher returns, while episode lengths are similar.

C.4 Further comparisons with baselines in manipulation finetuning

We compare FPO++ with DPPO and ReinFlow [8] using the original implementations of these methods in Figure A.8. In this setting, we adapt our training pipeline to match their implementation details, including adopting an action chunk size of four, rather than sixteen, to align with the DPPO/ReinFlow formulation. We train FPO++ and vanilla FPO within this adapted codebase, and retrain DPPO and ReinFlow using the authors' released code, using comparable base policies and evaluating all methods under the same simulation environments. This additional experiment allows for a further algorithmic comparison while controlling for implementation-specific factors.

Base policy setting. The base policies' performance, reflected in the success rates at 0 total environment steps in Figure 5, provides the anchor for finetuning. FPO++ and vanilla FPO share the same base policy, while DPPO and ReinFlow utilize different pretrained policies structured according to their original papers. The policies for DPPO and ReinFlow were trained and evaluated using 100 denoising/flow steps, whereas the base policy for FPO++ and vanilla FPO was trained and evaluated with 10 flow steps. The base policy success rates (evaluated on 1,000 episodes) are

- **Can task**: FPO++/vanilla FPO (73.76%), ReinFlow (76.30%), and DPPO (76.10%).
- **Square task**: DPPO (37.60%), ReinFlow (36.50%), and FPO++/vanilla FPO (28.62%).

Analysis of finetuning performance. Figure A.8 shows finetuning success rates, evaluated by collecting 200 episodes per checkpoint using 50 parallel simulation environments. FPO++ achieves the highest final success rates on both the Can and Square tasks. On Can, FPO++ demonstrates rapid early learning, reaching high success significantly faster than all baselines. Notably, vanilla FPO also performs robustly in this finetuning setting, achieving competitive performance without the policy collapse or gradual degradation as observed in Figure 5. Nevertheless, FPO++ yields substantial gains over vanilla FPO, both in terms of convergence speed and final performance on the more challenging Square task, which requires higher-precision control.

C.5 Detailed analysis on the RoboMimic Can experiment

Although the DPPO paper reports high success rates on the Can task, our experiments in the main text show DPPO variants struggling. We attribute this discrepancy to differences between base policies.

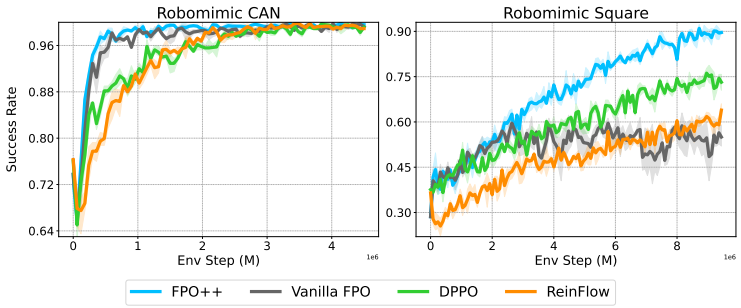


Figure A.8: **Additional Manipulation Comparison.** Finetuning success rates on the RoboMimic *Can* and *Square* tasks for flow-based RL methods. All methods start from pretrained behavior-cloning base policies; FPO++ and vanilla FPO share the same base policy, while DPPO and ReinFlow use architectures and implementation from their original papers. Please refer to the base policy setting written in the text for further details. FPO++ learns fastest and attains the highest final success on both tasks, while vanilla FPO, DPPO, and ReinFlow remain stable but underperform relative to FPO++.

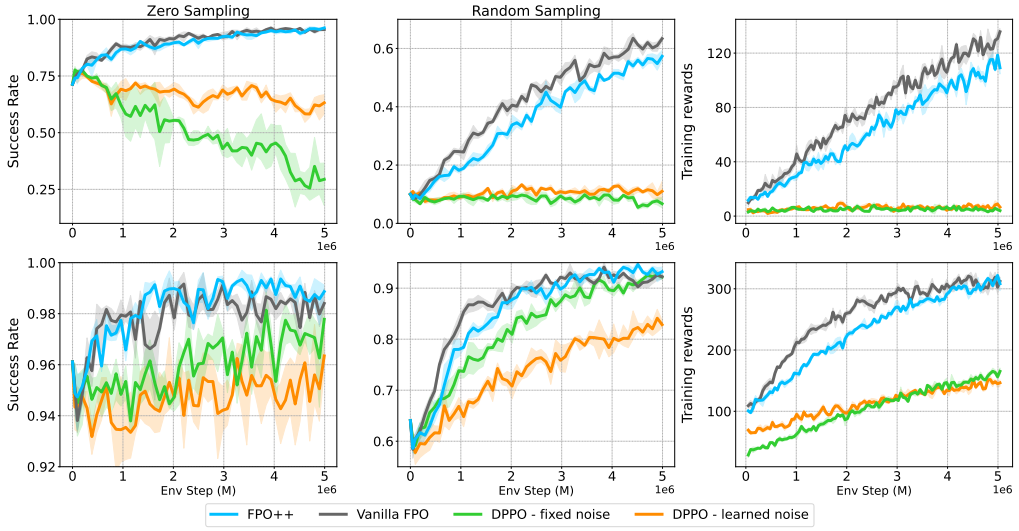


Figure A.9: **Analysis of base policy quality in RoboMimic Can.** This figure illustrates how the initial success rate of the base policy impacts different finetuning methods. The top row shows results for a base policy with 71.35% success under zero-sampling versus 10.00% under random sampling, while the bottom row depicts a higher-quality base policy (96.11% zero-sampling / 64.36% random sampling), demonstrating that DPPO variants are significantly more sensitive to the base policy quality.

As illustrated in Figure A.9, our flow-based base policy exhibits a high success rate when evaluated with zero-sampling. However, the success rate under the random sampling used for exploration during training is significantly lower. As a result, very few trajectories successfully reach the goal in each rollout phase. Combined with rewards only provided at the end of rollouts, this results in high gradient variance.

We find that FPO++ and vanilla FPO are relatively robust to this high-variance learning regime, while DPPO variants degrade substantially. In particular, although all methods begin from the same base policy and therefore share the same initial success rate under random sampling, FPO-based methods achieve noticeably higher training returns early in finetuning, providing sufficient reward signal for stable policy-gradient updates. In contrast, DPPO introduces additional stochasticity by injecting noise at each diffusion timestep during training, which further reduces the probability

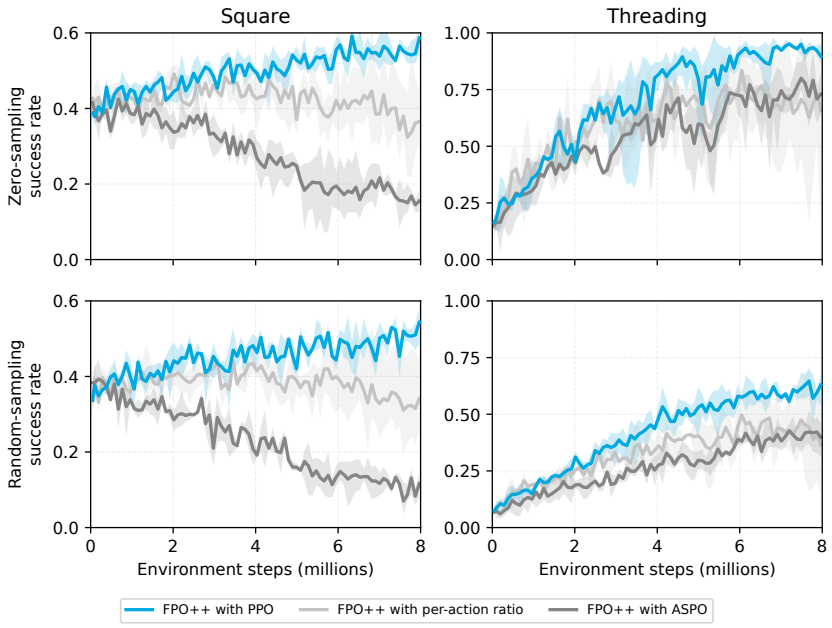


Figure A.10: **Ablation study in manipulation tasks.** Blue denotes per-sample ratios with PPO, light gray per-action ratios with PPO, and dark gray per-sample ratios with ASPO. Results are averaged over three training seeds per configuration, with 200 evaluation episodes per checkpoint per seed. Per-sample PPO achieves the highest final mean success on both Square and Threading. ASPO underperforms per-sample PPO in both tasks, despite its benefits for locomotion.

that rollouts reach the task completion state and leads to near-zero returns for a large fraction of trajectories. This exacerbates gradient noise and prevents effective learning in the early stages of training. As shown in the bottom row of Figure A.9, when the base policy's random sampling success rate is high enough (64.36%) to provide consistent initial rewards, DPPO variants succeed in finetuning the policy and achieving competitive success rates.

C.6 Ablation study for manipulation finetuning

In our main manipulation finetuning experiments, we observed that while FPO++ consistently outperforms baselines, specific algorithmic components contribute differently to success than they do in the from-scratch locomotion setting.

We conducted an ablation study across two manipulation tasks: the RoboMimic **Square task** and the DexMimicGen **Threading task**. These tasks both involve insertion, requiring high-precision control, and demonstrated a meaningful performance margin between FPO++ and vanilla FPO. As illustrated in Figure A.10, while the per-sample ratio consistently benefits FPO++ as it does in locomotion, the ASPO trust region is detrimental for manipulation finetuning.

Appendix C.7 tests whether this stems from pretraining by repeating the PPO and ASPO comparison on Lift with randomly initialized action networks.

C.7 ASPO under sparse and dense rewards

Experiment setup. To test whether ASPO's limitation is specific to finetuning, we compare PPO and ASPO trust regions in FPO++ on RoboMimic Lift, using the best-performing hyperparameter configuration found for each trust region under each reward setting. Action networks are randomly initialized, while the shared pretrained vision encoder remains frozen. We use ten training seeds per configuration, with budgets of 2M steps for sparse binary rewards and 1M for dense shaped rewards.

Each checkpoint is evaluated over 100 episodes per seed using the same task-completion criterion in both settings.

Results. PPO achieves higher final zero-sampling success under both reward settings (Figure A.11): 0.73 versus 0.07 for ASPO with sparse rewards, and 0.81 versus 0.28 with dense rewards. Sparse ASPO barely learns, whereas dense ASPO initially improves before success declines despite relatively high shaped return. Rollouts reveal grasp-and-hover behavior that earns shaping reward without completing the lift.

Analysis. These results suggest that initialization alone does not explain ASPO's limitations, even when each trust region uses its best-tuned configuration. With sparse rewards, limiting likelihood decreases for negative-advantage actions may keep the policy near its random initialization, hindering bootstrapping from rare successes. With dense rewards, the divergence between success and return highlights the importance of alignment with task completion. This supports the reward-dependent guidance in Section 4.4.

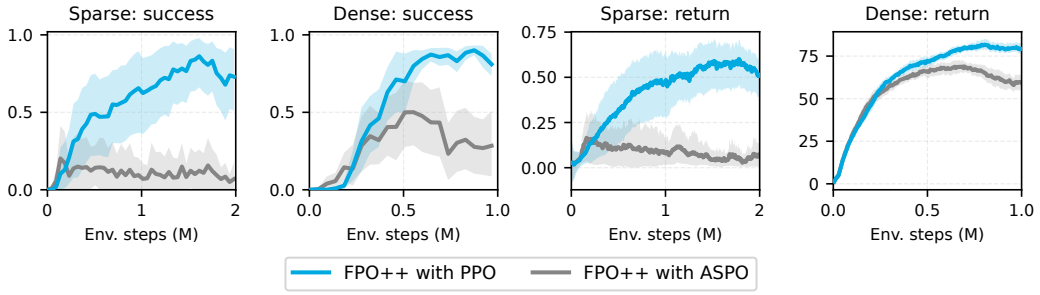


Figure A.11: **ASPO sensitivity to the reward regime in Lift.** PPO and ASPO each use their best-tuned configuration for each reward setting, evaluated over ten training seeds. Success uses zero-sampling and the task-completion criterion.

D Analysis of training dynamics

D.1 Empirical gradient variance

One explanation for FPO++'s improved stability over FPO is reduced gradient variance: finer-grained clipping with the per-sample ratio creates a larger effective batch size, while for negative advantages, ASPO provides gradients even for ratios beyond the trust region. We plot a cosine similarity metric inspired by [46] in Figure A.12: for each policy update, we compute cosine similarities between individual gradients and the average gradient within the policy update. We find higher similarity when the proposed ASPO and per-sample ratios are used during training.

D.2 Entropy and full-action-space flow fields

In addition to the qualitative analysis presented in the main text, we provide visualizations of the flow fields across the full action space (19 DoF) for the H1 humanoid's velocity-conditioned locomotion task. These supplementary figures illustrate the impact of the trust region objective on the policy distribution at different stages of training. Specifically, the FPO++ policy trained with the standard PPO trust region exhibits a clear narrowing of the action distribution (entropy collapse) as performance degrades, evident when comparing the field at its reward peak (Figure A.14) to the field during collapse (Figure A.15). In contrast, the policy trained with the ASPO trust region maintains a broader, more exploratory distribution, consistently preserving entropy from an intermediate checkpoint (Figure A.16) to its final, converged state (Figure A.17). This shows how the asymmetric objective prevents entropy collapse. For visualization, we plotted the transporting trajectories of the prior Gaussian noise to the action space by sampling $10{,}000$ noises per state and using 8 discretized Euler integration steps. Figure A.13 first summarizes this contrast for a single joint, comparing the

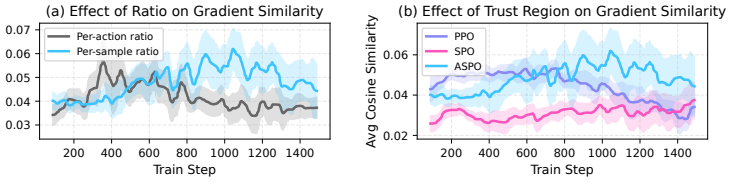


Figure A.12: **Algorithmic updates in FPO++ reduce gradient variance.** We observe that the per-sample ratio and ASPO trust region result in higher cosine similarity between gradients computed within each policy update, especially during the second half of training. Averages and standard deviations are reported over 5 seeds.

flow field under PPO and ASPO clipping over the course of training; Figures A.14–A.17 then show the full action space.

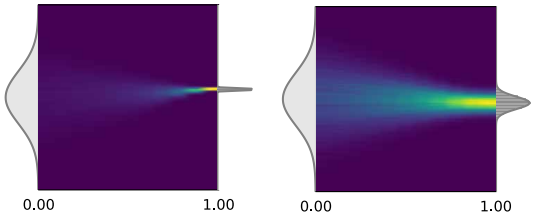


Figure A.13: **Effect of ASPO on entropy.** We show a flow field during training of H1 locomotion for a single robot joint, with FPO++ using PPO clipping (left) and ASPO clipping (right). The x-axis is integration step τ.

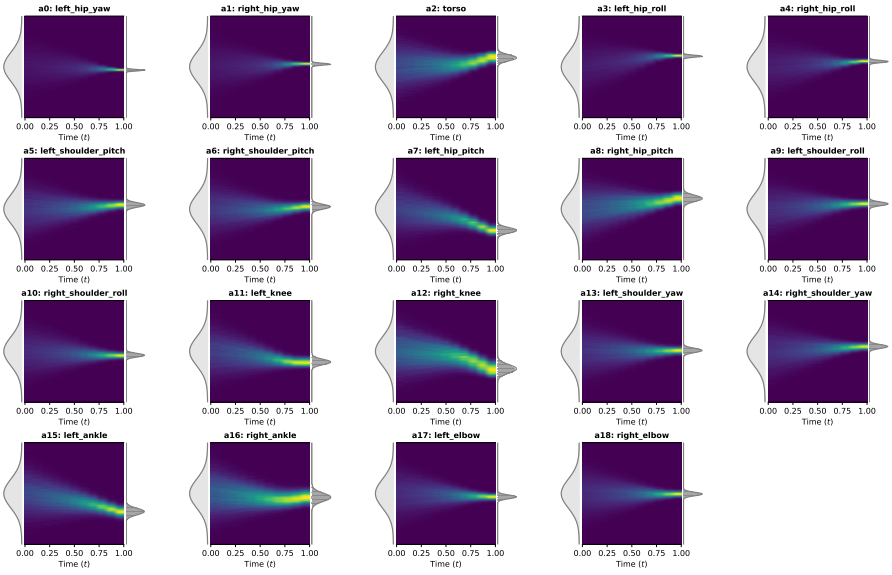


Figure A.14: **FPO++ with PPO clipping at peak performance**. Policy flow field density for the Unitree H1 humanoid trained using the standard PPO trust region, captured at the point of maximum average return. The flow is still relatively broad, indicating adequate exploration.

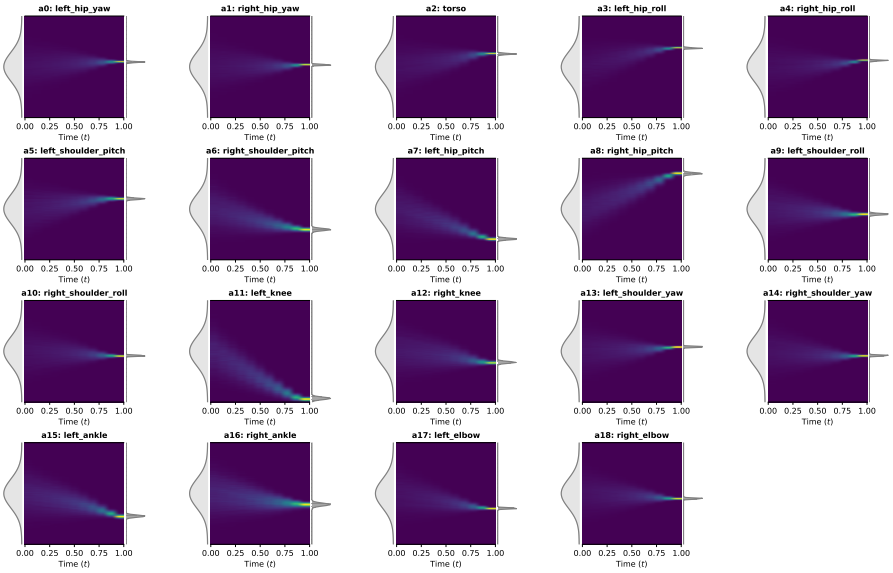


Figure A.15: **FPO++ with PPO clipping during policy collapse after convergence.** Policy flow field density for the Unitree H1 humanoid trained using the standard PPO trust region, captured as the average return begins to collapse. The action distribution has narrowed significantly for many joints, demonstrating the entropy collapse that leads to instability and performance degradation.

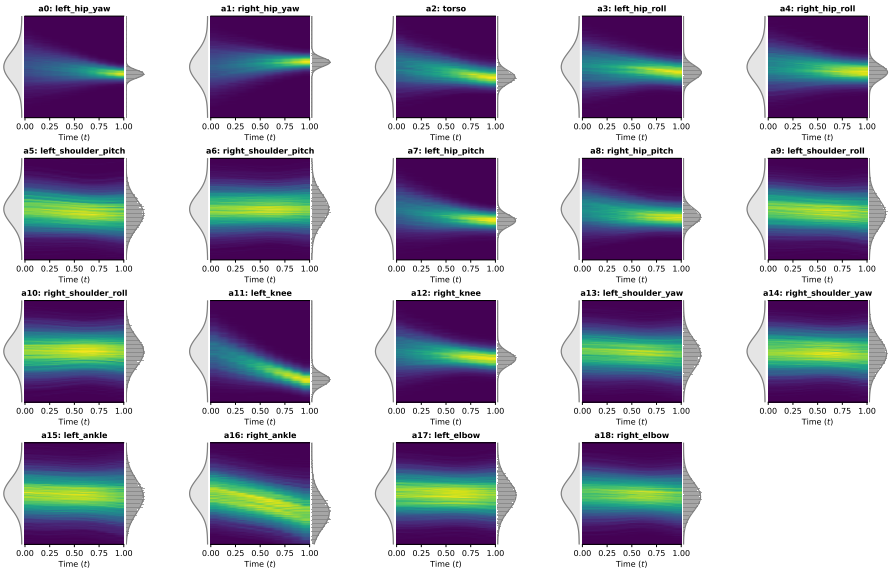


Figure A.16: **FPO++ with ASPO objective at intermediate training stage.** Policy flow field density for the Unitree H1 humanoid trained using the entropy-preserving ASPO objective, captured at an intermediate stage where the average return matches the peak average return of FPO++ with the standard PPO trust region. This distribution is already wider and more exploratory than the PPO-clipped policy at a similar or later time step, contributing to stable training.

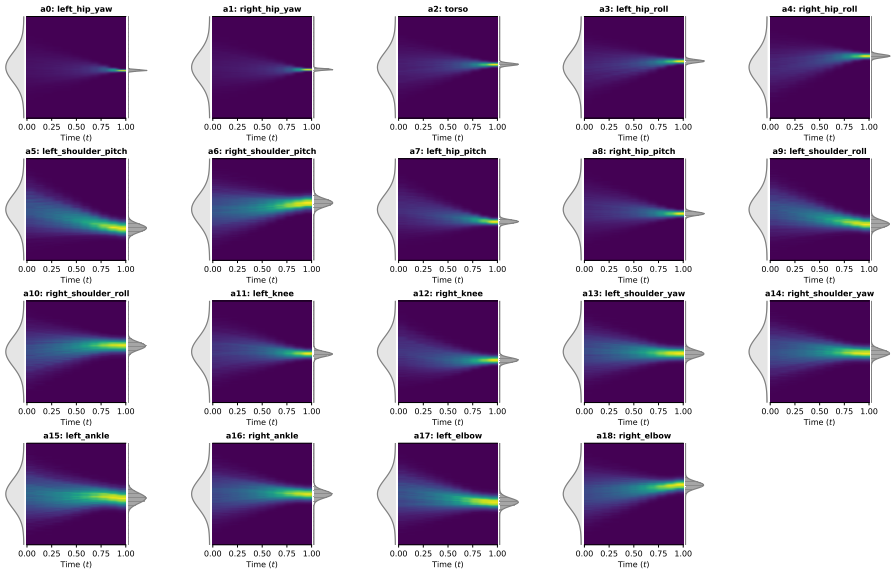


Figure A.17: **FPO++ with ASPO objective at converged state.** Policy flow field density for the Unitree H1 humanoid trained using the ASPO objective, captured at its final, high-reward converged state. The distribution remains wide and exploratory, confirming that ASPO effectively preserves entropy and prevents the collapse observed with PPO clipping.

D.3 Zero-sampling

Policy gradient methods require stochastic policies for exploration during training. Performance at test-time, however, can benefit from deterministically choosing actions. During training, FPO++ policies explore by drawing initial noises from $\epsilon \sim \mathcal{N}(0, I)$ and performing Euler integration over the learned flow field; at test-time and for computing evaluation metrics, we initialize flow integration from $\epsilon = \vec{0}$. We refer to this as *zero-sampling*. Similar to concurrent analysis on sampling in behavior cloning [68], we find that this improves performance across tasks. The effect can be observed in the gap between locomotion train and eval returns in Figure A.5, in the gap between zero-sampling and random sampling success rates for manipulation in Figure 5, and in motion tracking rewards with different sampling strategies in Table A.3.

Sampling method	5 steps	50 steps
Random sampling	34.7 ± 55.0	38.4 ± 26.6
Zero-sampling	$\mathbf{45.1 \pm 27.4}$	$\mathbf{45.5 \pm 23.2}$

Table A.3: **Zero-sampling improves motion tracking.** Returns are computed over 100 rollouts of a 2.5-minute dancing sequence, with domain randomization and push perturbations. The policy is trained with 50 flow steps; we evaluate using random sampling and zero-sampling with 5 and 50 flow steps.

Zero-sampling is critical, especially for sim-to-real. The gap between rollouts with random sampling and rollouts with zero-sampling can be drastic: G1 policies trained with 2^9 parallel environments, for example, achieve an average train return of under 20 but eval return above 32. Success rates in manipulation base policies, before any finetuning is applied at all, can jump from around 10% to over 70%. We also observe in Table A.3 that zero-sampling allows us to significantly reduce Euler integration steps with only a negligible drop in policy performance. This enables lower-latency action sampling using the on-board computer of the robot.