SEARCH TO STEER: LEARNING DESIRABILITY-GUIDED FIELDS FOR FLOW POLICY REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow matching provides an expressive policy representation for robot learning, but policies trained solely on offline demonstrations remain limited by data quality and coverage. Reinforcement learning can address these limitations, yet fine-tuning flow policy faces costly likelihood computation and the risk of mode collapse under reward maximization. In this paper, we propose a flow policy refinement framework that learns to steer action generation through desirability-guided search. Specifically, desirability-guided search explores local improvements in noisy action space without collapsing distinct behaviors modes. The resulting search directions then provide supervision for learning a guidance field that steers action generation toward higher-return outcomes. This formulation avoids calculation of policy likelihood ratios and backpropagation through the critic or the full sampling trajectory. The resulting guidance field can be integrated directly into flow sampling stage, enabling a lightweight inference pipeline without additional online planning.

1 INTRODUCTION

Flow matching has achieved success across a range of generative modeling tasks through its ability to represent complex, multimodal distributions (Lipman et al., 2022; Esser et al., 2024; Le et al., 2023). More recently, Diffusion and flow based models have been applied to imitation learning, for example, Diffusion Policy generates action sequences through conditional denoising (Chi et al., 2025), while FlowPolicy uses consistency flow matching for efficient action generation in robot manipulation (Zhang et al., 2025). Beyond task-specific imitation learning, flow matching also supports action generation in vision–language–action (VLA) models. For example, π_0 series model combine a pretrained vision–language model with a flow-matching action expert to generate continuous action chunks Black et al. (2024) Together, these developments establish flow matching as an important approach to action generation, from task-specific imitation policies to generalist VLA models Yu et al. (2026).

Despite this progress, policies trained by imitation remain constrained by the quality and coverage of their demonstrations. Since imitation learning only encourages policies to reproduce behaviors present in the dataset, it cannot directly improve upon or discover behaviors beyond the demonstrated trajectoriesLuo et al. (2024). Reinforcement learning (RL) provides a pathway for policy improvement by optimizing task returns, enabling the discovery of higher-performing behaviors outside the demonstration distribution Yamagata et al. (2023); Wang et al. (2022).

For flow policies, however, converting reward signals from RL into policy updates remains challenging compared with policies with explicit action densities. Since actions are generated through multiple integration steps, the velocity network does not directly provide an action likelihood. Recent approaches address this challenge by introducing stochastic flow transitions or flow-matching objectives for policy optimization (Zhang et al., 2026; McAllister et al., 2026). Alternatively, value-guided approaches propagate gradients from a learned value function through the generation process, which requires differentiating through the sampler. While these methods provide effective policy optimization mechanisms, they can introduce substantial complexity when the goal is to refine an already capable pretrained flow policy rather than perform full policy optimization.

Instead of directly optimizing the entire flow policy, we consider a simpler question: *can we identify local improvements around the behaviors already generated by a pretrained policy?*

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.

As shown in Fig.1, SEARCH TO STEER first performs local search around intermediate points of the flow trajectory to identify promising action improvements. Specifically, noisy perturbations around a flow anchor are decoded into executable action chunks and evaluated using a desirability utility, where within-group utility differences provide local improvement directions. These search directions are then distilled as a timestep-conditioned residual network for steering reference flow policy. During deployment, the learned residual field can be directly added to the base vector field, requiring only one additional network evaluation per integration step without online candidate search or utility queries.

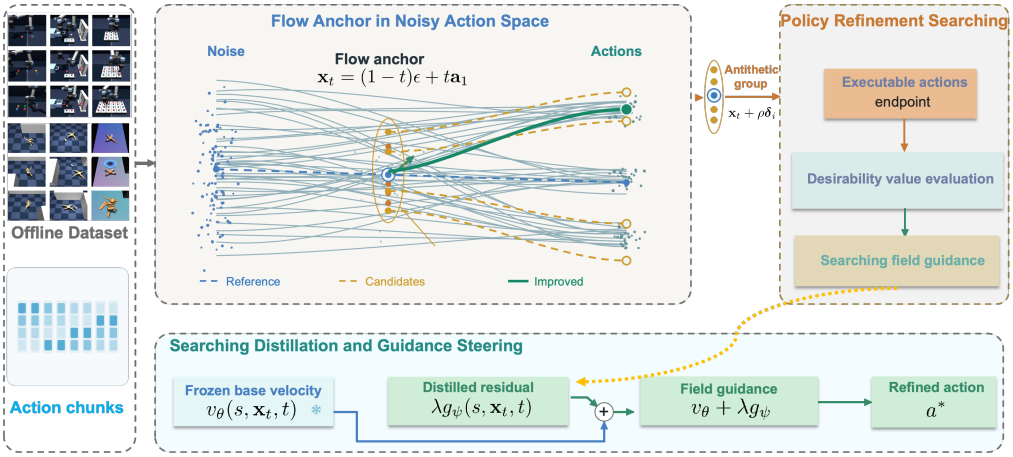


Figure 1: Overview of SEARCH TO STEER

2 RELATED WORK

Generative policies for imitation learning and VLA models. Diffusion Policy demonstrated the effectiveness of diffusion-based action generation for visuomotor imitation learning (Chi et al., 2025). Flow matching provides an alternative vector-field formulation for generative policy learning (Lipman et al., 2022), and FlowPolicy adopts consistency flow matching for efficient manipulation policy generation from demonstrations (Zhang et al., 2025). At a larger scale, π_0 combines flow-based action generation with pretrained vision–language models to enable general-purpose robot control (Black et al., 2024). These approaches primarily focus on learning expressive action distributions through large-scale supervised pretraining. In contrast, we study the subsequent policy refinement problem: how to improve a pretrained flow policy using return-based candidate evaluations while preserving the original generative behavior.

Reinforcement learning for diffusion and flow policies. DPPO fine-tunes diffusion control policies using policy gradients through the denoising process (Ren et al., 2025). For flow policies, ReinFlow introduces stochasticity into deterministic flow trajectories to obtain tractable transition likelihoods (Zhang et al., 2026), while Flow Policy Optimization develops a flow-matching surrogate for PPO-style policy updates (McAllister et al., 2026). These methods demonstrate alternative approaches for optimizing flow policies without relying on explicit action likelihoods. In contrast, our method studies how to efficiently refine pretrained flow policies by leveraging their existing generative capabilities.

Value-guided policy improvement and extraction. Recent works have explored how value functions can be integrated with generative policies to enable reward-driven policy improvement and behavior extraction. Diffusion-QL combines expressive behavior modeling with value-based policy optimization for diffusion policies (Wang et al., 2022). FQL trains a separate one-step actor through value maximization with a flow-policy distillation regularizer (Park et al., 2025b). Beyond direct value optimization, CFGRL studies conditional and unconditional generative guidance as a controllable policy improvement mechanism (Frans et al., 2025), while DIPOLE combines reward-maximizing and reward-minimizing policies for controllable behavior extraction (Liang et al., 2026).

Q-Guided Flow further explores test-time refinement of flow policies through value-gradient guidance (Zhou et al., 2026). In contrast, our method improves a pretrained flow policy by learning a residual steering field from training-time searching stage, avoiding test-time planning and the need to backpropagate value gradients through the generation process.

3 BACKGROUND

3.1 POLICY IMPROVEMENT FROM OFFLINE DATA

We consider a discounted Markov decision process with state $s \in \mathcal{S}$, action $a \in \mathcal{A}$, transition kernel P, reward function r, discount factor $\gamma \in [0, 1)$, and initial-state distribution p_0. For goal-conditioned tasks, we define $z = (s, g)$ to the state and goal. The policy objective is

$$J(\pi) = \mathbb{E}_{p_0,P,\pi}\left[\sum_{k=0}^{\infty} \gamma^k r(z_k, a_k)\right]. \tag{1}$$

An offline dataset $\mathcal{D}$ provides trajectories for learning a reference policy. Our goal is to improve this policy using return-based supervision. In the primary setting, we learn a chunk utility function from recorded transitions. The utility may alternatively be derived from online environment interaction, restored simulator rollouts, or a learned world model, each of which introduces different access requirements and estimation errors.

The generated variable is an action chunk $A = (a_k, \ldots, a_{k+L-1}) \in \mathbb{R}^D$ with horizon L, where $D = Ld_a$. The standard single-action setting is recovered when $L = 1$. A chunk policy executes the first $\ell \leq L$ actions before replanning. We use k to denote environment time and $t \in [0, 1]$ to denote the internal generation time of the flow process.

3.2 RECTIFIED-FLOW POLICIES

A conditional flow policy transforms Gaussian noise into an action representation using a velocity field $v_\theta(z, x_t, t)$. For an offline pair (z, A), the rectified-flow path and its training objective are

$$\begin{aligned} x_t &= (1-t)\epsilon + tA, \qquad \epsilon \sim \mathcal{N}(0, I_D), \quad t \sim \mathcal{U}[0, 1], \\ \mathcal{L}_{\mathrm{FM}}(\theta) &= \mathbb{E}_{(z,A)\sim\mathcal{D},\epsilon,t}\left\|v_\theta(z, x_t, t) - (A - \epsilon)\right\|^2. \end{aligned} \tag{2}$$

This conditional flow-matching objective learns the velocity from noise at $t = 0$ toward data at $t = 1$ (Lipman et al., 2022). At inference, we draw $x_0 \sim \mathcal{N}(0, I_D)$ and numerically integrate

$$\frac{\mathrm{d}x_t}{\mathrm{d}t} = v_\theta(z, x_t, t) \tag{3}$$

to obtain an action x_1. The policy randomness comes from x_0, even though the subsequent integration is deterministic. Flow time controls action generation and does not advance the environment. In our refinement stage, θ remains frozen and a separate residual modifies this velocity field.

4 METHOD

SEARCH TO STEER learns a steering field for refining a frozen flow policy from local action search. The method first learns a desirability value for evaluating action chunks, then uses this utility to identify local improvement directions around intermediate flow states. These directions are then used to train a residual field that guides the original flow dynamics during action generation. The following subsections describe the utility learning, desirability-guided search, and residual field learning procedures.

4.1 DESIRABILITY VALUES

Flow policies can represent multiple behaviors for the same state. However, reward-driven fine-tuning may shift the policy distribution toward a small number of high-scoring actions and reduce

behavioral diversity. To better compare these behaviors, we introduce desirability values based on their return distributions. This enables the search procedure to favor actions with desirable long-term outcomes while avoiding the assumption that higher return variability is always beneficial.

Specifically, we consider an action chunk $A = (a_k, \ldots, a_{k+L-1})$ and let G denote the discounted return obtained by executing the chunk at context z and then following a continuation policy π_c. Its desirability value is

$$\mathcal{H}_\beta(z, A) = \mathbb{E}[e^{\beta G} \mid z, A], \qquad U_\beta(z, A) = \begin{cases} \beta^{-1} \log \mathcal{H}_\beta(z, A), & \beta \neq 0, \\ \mathbb{E}[G \mid z, A], & \beta = 0. \end{cases} \tag{4}$$

The logarithm maps the desirability measure back to the return scale. A positive β emphasizes higher-return outcomes by assigning larger weights to favorable returns. As a result, the desirability value captures not only the average return but also the shape of the return distribution, allowing actions with different return profiles to be compared beyond their expected values.

To estimate the desirability value in practice, we learn a chunk value function $Q_\phi(z, A, \beta)$ to approximate $U_\beta(z, A)$. The corresponding continuation value accounts for future chunks selected by the continuation policy π_c at the next decision point:

$$\widehat{V}_{\phi,\beta}(z) = \frac{1}{\beta} \log \mathbb{E}_{A' \sim \pi_c(\cdot|z)} \left[\exp\{\beta Q_\phi(z, A', \beta)\} \right]. \tag{5}$$

Let $R_L = \sum_{h=0}^{L-1} \gamma^h r_{k+h}$ be their discounted reward sum, z_L the following context, and m_L indicate whether continuation is possible. For a deterministic reward prefix and transition, the regression objective is

$$\begin{aligned} y_\beta &= R_L + \gamma^L m_L \widehat{V}_{\bar{\phi},\gamma^L\beta}(z_L), \\ \mathcal{L}_Q(\phi) &= \mathbb{E}_{\mathcal{D}} \left[\left(Q_\phi(z, A, \beta) - \mathrm{sg}(y_\beta) \right)^2 \right], \end{aligned} \tag{6}$$

where $\bar{\phi}$ is a slowly updated target network and sg stops gradients through the targetFujimoto et al. (2019). The factor $\gamma^L\beta$ accounts for discounting the continuation return. After training, we freeze the value and use it to score search candidates.

After training, the learned value function provides the utility estimates used for candidate comparison during search.

$$\widehat{U}^{\mathrm{off}}_\beta(z, A) = Q_\phi(z, A, \beta). \tag{7}$$

The score estimates the return of executing the candidate chunk followed by the completion policy π_c, without requiring additional environment interaction. In our experiments, we focus on the offline setting where the entire score is estimated from the learned value function. However, the formulation is not restricted to this case and can naturally incorporate additional rollout information when available. Specifically, if environment interaction or a pretrained world model is accessible, we can execute π_c for the first H steps, combine the accumulated rollout rewards with the predicted future return, and obtain a hybrid rollout-based score:

$$\widehat{U}_{\beta,H}(z, A) = \frac{1}{\beta} \log \left[\frac{1}{M} \sum_{j=1}^{M} \exp\left\{ \beta \left(R_H^{(j)} + \gamma^H m_H^{(j)} \widehat{V}_{\phi,\beta\gamma^H}(z_H^{(j)}) \right) \right\} \right]. \tag{8}$$

Here $R_H^{(j)}$ contains the rewards observed during rollout j, and the terminal value estimates what remains after that rollout.

4.2 Local search in noisy action space

Local search explores how perturbations around an intermediate flow state change the utility of the decoded action chunk. The perturbed noisy intermediate states are mapped to an executable action chunk and scored by the utility function. By comparing these candidate utilities, the search procedure extracts a direction in noisy action space that improves the generated behavior.

Given an offline state–action pair (z, A), the search procedure first constructs an intermediate point along the flow trajectory. Specifically, we sample a flow time t and Gaussian noise ϵ, and define the

anchor as $x_t = (1-t)\epsilon + tA$. Around this anchor, we generate K antithetic perturbations: each uniformly sampled unit direction δ_i is paired with its opposite direction $-\delta_i$. The resulting noisy candidates are given by

$$x_t^{(i)} = x_t + \rho_t \delta_i, \qquad \rho_t = c\sqrt{L}(1-t), \qquad i = 1, \ldots, K. \tag{9}$$

To evaluate each perturbed candidate, we use the frozen flow model to estimate its clean action chunk. A single evaluation of the velocity field yields the following endpoint approximation:

$$F_t(z, x) = \mathrm{clip}_{\mathcal{A}^L}(x + (1-t)v_\theta(z, x, t)) \,. \tag{10}$$

This approximation uses the current velocity field to perform a first-order extrapolation to the end of the flow trajectory. The resulting action chunk is clipped to the environment's valid range and evaluated by the utility function as follows

$$u_i = \widehat{U}_\beta\Big(z, F_t(z, x_t^{(i)})\Big) \,, \tag{11}$$

As shown above, the utility values of candidate action chunks can determine whether the sampled perturbations provide a meaningful search signal. However, when all candidates receive similar utilities, the resulting direction is dominated by numerical noise rather than meaningful preference differences. Let $\bar{u}$ denote the mean candidate utility and s_u its standard deviation. We therefore introduce a spread gate to suppress search directions when candidate utilities are nearly indistinguishable:

$$\bar{u} = \frac{1}{K}\sum_i u_i, \qquad s_u = \sqrt{\frac{1}{K}\sum_i (u_i - \bar{u})^2}, \qquad b = \mathbf{1}[s_u \geq \tau]. \tag{12}$$

The gate threshold $\tau > 0$ filters out candidate sets with nearly identical utility scores, preventing normalization from amplifying negligible score differences.

For an accepted comparison, we weight each perturbation by how far its score lies above or below the mean to obtain the search direction d_x:

$$\widetilde{A}_i = \frac{u_i - \bar{u}}{\max(s_u, \varepsilon_u)}, \qquad d_x = \frac{b}{K \max(\rho_t, \varepsilon_\rho)} \sum_i \widetilde{A}_i \delta_i. \tag{13}$$

Candidates with above-average utility reinforce their perturbation directions, whereas below-average candidates reverse them. The standard-deviation normalization makes the resulting direction less sensitive to variations in utility scale across states. The constants ε_u and ε_ρ provide numerical stability when normalizing utility differences and perturbation magnitudes, respectively.

Eq. (13) combines the utility differences between paired perturbations into a single search direction. We next show that this direction is related to a finite-difference estimate of the gradient of a smoothed utility function. Fix z, t and make $x = x_t$ and $\rho = \rho_t > 0$. Define $f(x) = \widehat{U}_\beta(z, F_t(z, x))$ as the utility of a noisy action representation, and let its ball-smoothed version be $f_\rho(x) = \mathbb{E}_{v \sim \mathrm{Unif}(\mathbb{B}_D)}[f(x + \rho v)]$, where $\mathbb{B}_D$ denotes the unit ball in D dimensions.

The following proposition formalizes this connection by showing that the paired perturbation estimator recovers the gradient of the smoothed utility function in expectation.

Proposition 1 (Search direction and the smoothed score gradient)**.** *Suppose f is continuously differentiable on a neighborhood of $x + \rho\mathbb{B}_D$. Draw $K/2$ independent directions uniformly from the unit sphere and include their negatives. Then*

$$\widehat{g}_\rho(x) = \frac{D}{K\rho} \sum_{i=1}^{K/2} \big[f(x + \rho\delta_i) - f(x - \rho\delta_i)\big]\delta_i \tag{14}$$

satisfies $\mathbb{E}[\widehat{g}_\rho(x)] = \nabla f_\rho(x)$. For every accepted candidate set, the direction in Eq. (13) *obeys*

$$d_x = \frac{\rho}{D \max(\rho, \varepsilon_\rho) \max(s_u, \varepsilon_u)} \widehat{g}_\rho(x). \tag{15}$$

Algorithm 1 Search to Steer

Input: Offline trajectories $\mathcal{D}$; frozen flow v_θ; decoder F_t; search settings K, c, τ.
Output: Steering field g_ψ and validation-selected strength λ.

Training: local search Eq. (13)

Fit and freeze the chunk value Q_ϕ; initialize search cache $\mathcal{C} = \varnothing$.
for sampled offline context–chunk pair $(z, A) \in \mathcal{D}$ **do**
  Draw t in the search interval and $\epsilon \sim \mathcal{N}(0, I_D)$; set $x_t = (1-t)\epsilon + tA$.
  Sample $K/2$ unit directions and their negatives; set $\rho_t = c\sqrt{L}(1-t)$.
  Decode $A_i = F_t(z, x_t + \rho_t \delta_i)$ and score $u_i = Q_\phi(z, A_i, \beta)$ for $i = 1, \ldots, K$.
  Compute $\bar{u}$, s_u and b using Eq. (12); obtain d_x from Eq. (13).
  Add (z, x_t, t, d_x, b) to $\mathcal{C}$.
end for

Training: steering-field fitting Eq. (16)

Fit g_ψ on the training split of $\mathcal{C}$ using Eq. (16).
Select its checkpoint on held-out search examples and λ on validation episodes.

Deployment: guided generation Eq. (17)

while the episode continues **do**
  Observe context z and draw $x_0 \sim \mathcal{N}(0, I_D)$.
  for $j = 0, \ldots, N-1$ **do**
    Set $t_j = j/N$ and $x_{j+1} = x_j + \frac{1}{N}[v_\theta(z, x_j, t_j) + \lambda \chi(t_j) g_\psi(z, x_j, t_j)]$.
  end for
  Clip x_N to the action range, execute the chosen chunk prefix, and replan.
end while

The proposition shows that, up to sample-dependent normalization, the search direction corresponds to a finite-difference estimate of the smoothed utility gradient. This estimate relies only on scalar utility evaluations and does not require differentiating through either the value function or the flow decoder. Therefore, the resulting search directions can be directly used as supervision signals for learning the steering field in the next stage.

4.3 Steering Field Learning

Running the search for every action would be expensive. Instead, we learn a field $g_\psi(z, x_t, t)$ (teal) that predicts its directions, while keeping the base policy frozen. For cached anchors and search directions $d_{x,n}$, the training objective is

$$\mathcal{L}_g(\psi) = \frac{\sum_n b_n \left\| g_\psi(z_n, x_n, t_n) - d_{x,n} \right\|^2}{\max(\sum_n b_n, 1)}. \tag{16}$$

During generation, the learned field adds a correction to each flow step:

$$x_{j+1} = x_j + \Delta t \left[v_\theta(z, x_j, t_j) + \lambda \chi(t_j) g_\psi(z, x_j, t_j) \right], \qquad t_j = j/N, \quad \Delta t = 1/N. \tag{17}$$

Here λ controls steering strength; χ either enables steering throughout generation or restricts it to the times covered during training. We generate a complete chunk, execute its chosen prefix, and replan. Deployment uses one additional field evaluation per active flow step, with no candidate search or value queries. For more specific steps about our method, one can refer to Algorithm 1.

5 Experiments

We evaluate our method on ExORL continuous control and state-based, goal-conditioned OGBench, using raw return and task success rate as evaluation metrics, respectively (Yarats et al., 2022; Park et al., 2025a). Beyond these benchmarks, we include an eight-mode example to examine how return variability influences desirability-based preferences, and paired Scene Play rollouts to qualitatively compare the behaviors produced by our method and CFGRL. We further study the consistency of local search directions. Since removing inconsistent directions trades off supervision quality against the amount of available training data, we analyze its impact on direction fitting and downstream performance.

The ExORL benchmark compares our method with AWR, CFGRL, FQL, and DIPOLE, covering representative approaches based on advantage-weighted regression, classifier-free guidance, and

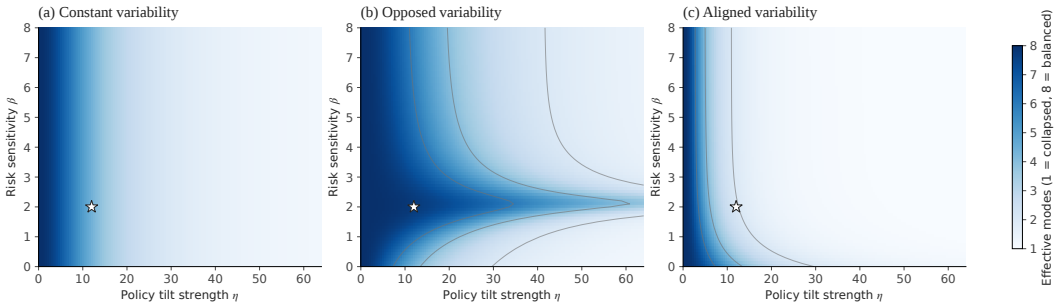


Figure 2: Effect of desirability on mode occupancy. Effective mode count as a function of policy tilt strength η and risk sensitivity β, under constant, opposed, and aligned return variability.

value-guided policy improvement. For DIPOLE, we adopt the single-action variant and disable Q-based action selection during inference for a fair comparison. On OGBench, we compare against BC, Flow-BC, GCBC, Flow-GCBC, and goal-conditioned CFGRL, covering both behavior cloning baselines and guided generative policy methods.

5.1 Analysis of desirability

Desirability is designed to account for the full return distribution rather than only the expected return of an action. We use an eight-mode bandit example to isolate how return variability influences the resulting action preferences. Each mode has a fixed mean return, while variability is assigned according to three settings: equal variability across modes (*constant*), higher variability for lower-mean modes (*opposed*), and higher variability for higher-mean modes (*aligned*). The desirability scores are computed analytically, and the reference distribution is reweighted with different values of β and weighting strength η. No policy is trained in this experiment; the complete setup is provided in Appendix C.

Figure 2 reports the effective number of modes computed from the entropy of the resulting mode distribution. In the constant setting, all modes receive the same variability contribution, and changing β therefore preserves their relative probabilities. In contrast, when variability is larger for lower-mean modes (*opposed*), increasing β can maintain probability mass on these modes because they retain access to higher-return outcomes. Conversely, when variability aligns with mean return (*aligned*), desirability further favors high-mean modes and leads to a more concentrated distribution. These results show that the effect of desirability on mode preservation depends on the relationship between return variability and mean performance.

5.2 Benchmark comparison

Tables 1 and 2 compare our method with the baselines on ExORL (Yarats et al., 2022) and OGBench (Park et al., 2025a). Our method achieves the highest mean return on six of the nine ExORL tasks and the highest mean success rate on eleven of the fifteen OGBench tasks.

On ExORL, our method performs particularly well on locomotion tasks, obtaining the best mean returns on Walker Stand, Walker Run, both Quadruped tasks, and both Cheetah tasks. FQL remains stronger on Walker Walk and the two Jaco reaching tasks. On OGBench, the gains extend across navigation and manipulation. These results suggest that local search provides useful supervision for refining action generation across many different tasks.

Our method improves the pretrained flow policy while keeping the original policy parameters fixed. The refinement is captured by an additional steering field learned from training-time search, where candidate exploration is only required during training. This preserves the efficiency of direct flow-based action generation while incorporating the benefits of search-based refinement.

Table 1: Performance of our method and baselines on ExORL (return).

Task	AWR	CFGRL	FQL	DIPOLE	Ours
Walker Stand	603 ± 8	782 ± 8	801 ± 4	793 ± 11	$\mathbf{815 \pm 6}$
Walker Walk	444 ± 4	608 ± 32	$\mathbf{755 \pm 12}$	679 ± 16	710 ± 14
Walker Run	247 ± 10	282 ± 6	294 ± 11	256 ± 12	$\mathbf{302 \pm 8}$
Quadruped Walk	776 ± 15	762 ± 25	739 ± 25	813 ± 21	$\mathbf{815 \pm 20}$
Quadruped Run	485 ± 7	571 ± 25	503 ± 5	560 ± 11	$\mathbf{590 \pm 16}$
Cheetah Run	168 ± 7	216 ± 15	222 ± 14	194 ± 9	$\mathbf{228 \pm 17}$
Cheetah Run Backward	146 ± 8	262 ± 26	231 ± 12	227 ± 7	$\mathbf{276 \pm 10}$
Jaco Reach Top Right	33 ± 2	72 ± 6	$\mathbf{224 \pm 17}$	84 ± 5	120 ± 14
Jaco Reach Top Left	30 ± 8	46 ± 6	$\mathbf{222 \pm 42}$	63 ± 8	70 ± 10

Table 2: Performance of our method and baselines on OGBench (success rate, %).

Task	BC	Flow-BC	GCBC	Flow-GCBC	CFGRL	Ours
Pointmaze Medium	0 ± 0	1 ± 2	9 ± 5	66 ± 4	77 ± 6	$\mathbf{82 \pm 5}$
Pointmaze Large	0 ± 0	0 ± 0	25 ± 9	74 ± 7	77 ± 5	$\mathbf{80 \pm 6}$
Pointmaze Giant	0 ± 0	0 ± 0	2 ± 2	4 ± 2	30 ± 10	$\mathbf{38 \pm 8}$
Pointmaze Teleport	0 ± 0	0 ± 1	24 ± 7	$\mathbf{46 \pm 4}$	41 ± 8	40 ± 6
Antmaze Medium	13 ± 2	15 ± 6	25 ± 8	42 ± 9	53 ± 12	$\mathbf{56 \pm 10}$
Antmaze Large	11 ± 4	5 ± 2	20 ± 4	22 ± 5	24 ± 10	$\mathbf{28 \pm 12}$
Antmaze Giant	0 ± 0	0 ± 0	0 ± 0	0 ± 0	$\mathbf{1 \pm 1}$	0 ± 0
Antmaze Teleport	4 ± 2	3 ± 2	19 ± 5	24 ± 7	35 ± 9	$\mathbf{42 \pm 8}$
Humanoidmaze Medium	3 ± 3	2 ± 2	6 ± 3	8 ± 3	19 ± 6	$\mathbf{22 \pm 5}$
Humanoidmaze Large	0 ± 1	0 ± 0	2 ± 1	1 ± 2	$\mathbf{3 \pm 2}$	2 ± 1
Antsoccer Arena	2 ± 2	4 ± 1	4 ± 3	10 ± 5	15 ± 6	$\mathbf{18 \pm 5}$
Antsoccer Medium	0 ± 0	0 ± 0	6 ± 4	5 ± 3	5 ± 3	$\mathbf{7 \pm 6}$
Cube Single	4 ± 2	7 ± 5	6 ± 2	8 ± 2	11 ± 5	$\mathbf{14 \pm 6}$
Cube Double	1 ± 1	2 ± 1	2 ± 1	$\mathbf{3 \pm 1}$	2 ± 2	2 ± 1
Scene Play	5 ± 2	5 ± 3	4 ± 3	15 ± 5	19 ± 4	$\mathbf{27 \pm 5}$

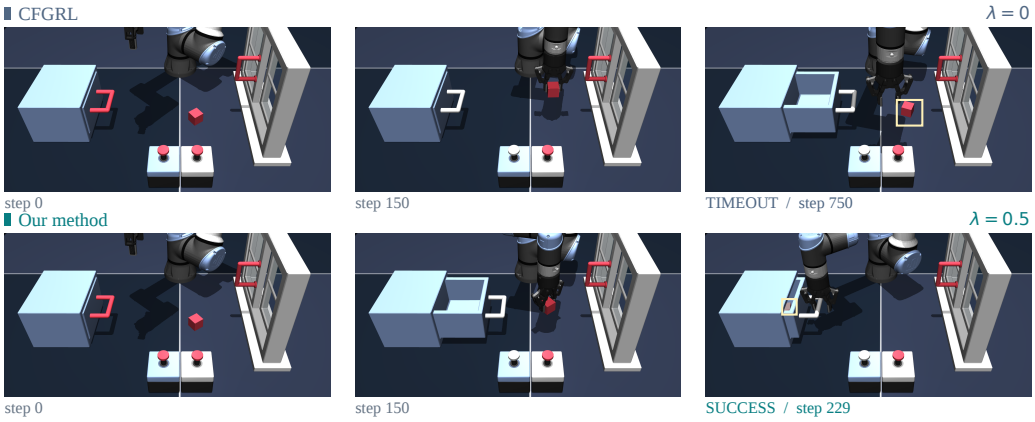


Figure 3: Example rollouts of CFGRL and our method on Scene Play Task.

5.3 Performance Analysis on Scene Play Task

We compare our method with CFGRL (Frans et al., 2025) on a Scene Play task in which a robot must place a cube inside a drawer. Success requires moving the cube to the target location; reaching or contacting it alone is insufficient. Figure 3 shows example rollouts from the same initial scene, with frames at the start, an intermediate step, and termination.

Both policies interact with the cube by step 150, CFGRL leaves the cube outside the drawer and reaches the episode limit at step 750, whereas our method places it inside the drawer and succeeds

at step 229. This example illustrates the difference between approaching the object and completing the placement sequence.

5.4 Analysis of Search-direction consistency

The steering field is trained from local search directions, making the reliability of these targets an important factor in refinement quality. However, when the search involves sampled continuations, repeated searches from the same anchor may produce inconsistent directions. We therefore study whether filtering inconsistent targets improves the quality of supervision. This analysis examines the trade-off between retaining more search samples and using a smaller set of more reliable directions.

At each anchor, two independent continuation samples produce search directions g_A and g_B. We retain an anchor when $\cos(g_A, g_B) > \tau_c$ and vary τ_c to control the filtering strength. Unlike the score-spread gate in Eq. (12), which evaluates utility variation within a single search, this criterion measures the agreement between independent searches. We evaluate the retained anchor ratio, the agreement between the learned field and search directions on a held-out set, and the resulting improvement over the frozen base policy. Together, these measurements characterize the trade-off between supervision quantity, target consistency, and the final quality of the learned steering field.

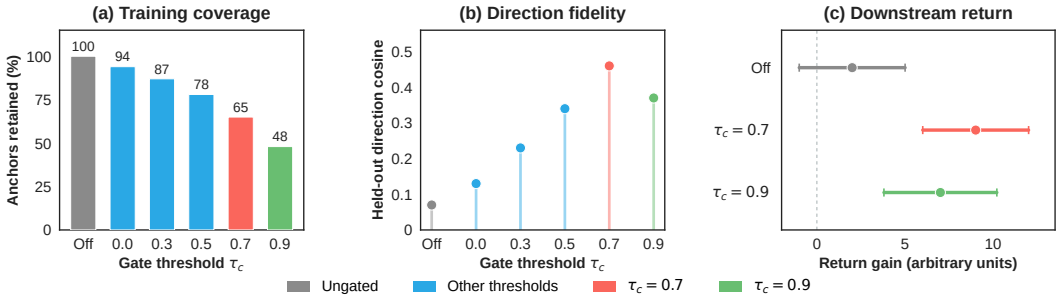


Figure 4: Search-direction consistency.

Figure 4 illustrates the intuition behind this trade-off. Moderate filtering can remove unreliable directions while retaining sufficient coverage for learning the steering field, whereas overly aggressive filtering may reduce the diversity of available supervision. The useful filtering strength therefore depends on its impact on the learned field and downstream policy performance rather than search agreement alone.

6 Conclusion

We presented a method for refining pretrained flow policies by distilling local search into a learned steering field. Candidate action chunks are evaluated through utility comparisons, and the resulting search directions provide supervision for a residual field added to the frozen flow policy. This enables the policy to benefit from training-time search without requiring candidate evaluation or planning during deployment. Experiments on ExORL and OGBench demonstrate competitive performance across continuous-control and goal-conditioned tasks, validating the effectiveness of the proposed refinement approach.

A promising direction for future work is to combine our method with pretrained world models. Model-generated rollouts could evaluate the longer-term consequences of candidate action chunks and provide additional supervision for the steering field without executing each candidate in the environment. Learning from these imagined outcomes could extend local search beyond the experience available in the offline dataset, while retaining the same lightweight policy at deployment.

Reproducibility Statement

Appendix B records experimental settings, and Appendix A provides derivations.

Ethics Statement

This work focuses on robot-control tasks and does not involve human subject. We do not claim that the resulting policies are suitable for deployment in real-world environments or around people.

AI Use Statement

Generative AI tools were used for language editing, manuscript polishing, mathematical writing refinement, and LaTeX preparation, and assistance in developing the numerical example. The authors remain fully responsible for the scientific content, including all claims, derivations, references, and experimental results.

References

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

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

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Diffusion guidance is a controllable policy improvement operator. *arXiv preprint arXiv:2505.23458*, 2025.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Matthew Le, Apoorv Vyas, Bowen Shi, Brian Karrer, Leda Sari, Rashel Moritz, Mary Williamson, Vimal Manohar, Yossi Adi, Jay Mahadeokar, et al. Voicebox: Text-guided multilingual universal speech generation at scale. *Advances in neural information processing systems*, 36:14005–14034, 2023.

Ruiming Liang, Yinan Zheng, Kexin Zheng, Tianyi Tan, Jianxiong Li, Liyuan Mao, Zhihao Wang, Guang Chen, Hangjun Ye, Jingjing Liu, et al. Dichotomous diffusion policy optimization. In *International Conference on Learning Representations*, volume 2026, pp. 61669–61690, 2026.

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

Jianlan Luo, Perry Dong, Yuexiang Zhai, Yi Ma, and Sergey Levine. Rlif: Interactive imitation learning as reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 36329–36351, 2024.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. In *International Conference on Learning Representations*, volume 2026, pp. 36352–36372, 2026.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pp. 94937–94982, 2025a.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. *arXiv preprint arXiv:2502.02538*, 2025b.

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.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Taku Yamagata, Ahmed Khalil, and Raul Santos-Rodriguez. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. In *International Conference on Machine Learning*, pp. 38989–39007. PMLR, 2023.

Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.

Zhaoshu Yu, Bo Wang, Pengpeng Zeng, Haonan Zhang, Ji Zhang, Zheng Wang, Lianli Gao, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. A survey on efficient vision-language-action models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. Flowpolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 14754–14762, 2025.

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.

Zhiyuan Zhou, Andy Peng, Charles Xu, Qiyang Li, Tobias Springenberg, Kevin Frans, and Sergey Levine. Test-time gradient guidance of flow policies in reinforcement learning. *arXiv preprint arXiv:2606.11087*, 2026.

A Search Directions and Steering

A.1 Proof of Proposition 1

Proof. Let $|\mathbb{B}_D|$ denote the volume of the unit ball. Differentiation under the integral and the divergence theorem give

$$\nabla f_\rho(x) = \frac{1}{|\mathbb{B}_D|}\int_{\mathbb{B}_D} \nabla f(x+\rho v)\,\mathrm{d}v \tag{18}$$

$$= \frac{1}{\rho|\mathbb{B}_D|}\int_{\mathbb{S}^{D-1}} f(x+\rho\delta)\delta\,\mathrm{d}S(\delta) \tag{19}$$

$$= \frac{D}{\rho}\mathbb{E}_\delta[f(x+\rho\delta)\delta]. \tag{20}$$

The last equality uses $|\mathbb{S}^{D-1}| = D|\mathbb{B}_D|$. Symmetry of the uniform sphere distribution implies

$$\mathbb{E}_\delta[f(x-\rho\delta)\delta] = -\mathbb{E}_\delta[f(x+\rho\delta)\delta]. \tag{21}$$

Taking expectations in Eq. (14), with $K/2$ sampled pairs, therefore yields $\mathbb{E}[\widehat{g}_\rho(x)] = \nabla f_\rho(x)$.

For the second claim, fix a realized candidate set and write $u_i = f(x+\rho\delta_i)$. Antithetic pairing gives $\sum_{i=1}^K \delta_i = 0$, so

$$\sum_{i=1}^K (u_i - \bar{u})\delta_i = \sum_{i=1}^K u_i\delta_i \tag{22}$$

$$= \sum_{i=1}^{K/2}[f(x+\rho\delta_i) - f(x-\rho\delta_i)]\delta_i = \frac{K\rho}{D}\widehat{g}_\rho(x). \tag{23}$$

Substitution into Eq. (13), with $b = 1$, proves Eq. (15). Its multiplier is strictly positive, including when either denominator floor is active. □

The expectation in the first claim is over all sampled directions, before gating. The second claim is a pointwise identity for accepted sets; it does not assert conditional unbiasedness after selection. That algebraic identity requires only finite scores and antithetic pairing, so it remains valid when action clipping makes f nonsmooth.

Conditional local ascent. Suppose additionally that ∇f_ρ is L_f-Lipschitz, with $L_f > 0$, in a region containing the proposed step. For an accepted direction with $a = \langle \nabla f_\rho(x), d_x\rangle > 0$, smoothness implies

$$f_\rho(x+\alpha d_x) - f_\rho(x) \geq \alpha a - \frac{L_f\alpha^2}{2}\|d_x\|^2 > 0, \quad 0 < \alpha < \frac{2a}{L_f\|d_x\|^2}. \tag{24}$$

By Eq. (15), this alignment condition is equivalent to positive alignment of the paired estimate with $\nabla f_\rho(x)$. Neither a finite candidate set nor the spread gate ensures that condition. The statement concerns the fitted score at a fixed context and flow time; value error, field-fitting error, and subsequent policy execution lie outside its conclusion.

A.2 Alternative gate and displacement target

The spread gate checks whether candidate scores differ enough to form a useful comparison. An optional improvement gate also scores the decoded anchor, $u_0 = \widehat{U}_\beta(z, F_t(z, x_t))$, and requires $\max_i u_i > u_0 + m$ for a chosen margin m. This extra test costs one scoring query per anchor. The recorded scene-play experiments use the spread gate alone.

The main method learns a raw noisy-space search direction. A different target can be formed from a clean-action displacement ΔA between a selected candidate and the decoded anchor. For the unclipped endpoint approximation, at fixed z, x, t,

$$\Delta A = (1-t)\Delta v, \qquad d_v = b\,\mathrm{clip}_C\left(\frac{\Delta A}{\max(1-t, \varepsilon_t)}\right). \tag{25}$$

Here clip_C caps the vector norm at C, and $\varepsilon_t > 0$ prevents large corrections near $t = 1$. The division converts an endpoint displacement into velocity units; it should not be applied automatically to d_x. This identity holds for the fixed-input endpoint approximation before action clipping, not for an entire nonlinear ODE trajectory. Raw directions and calibrated displacements are therefore separate target choices.

B HYPERPARAMETERS

Table 3 lists a reference configuration for policy training, local search, and steering-field distillation.

Table 3: Reference hyperparameters for our method.

Component	Parameter	Value
Base policy	Hidden layers	$(512, 512, 512, 512)$
	Training updates	250,000
	Batch size	1,024
Utility	Training updates	20,000
	Risk sensitivity β	0
Local search	Candidates per anchor K	32 (antithetic)
	Perturbation scale c	0.25
	Flow time t	$\mathcal{U}[0.2, 0.85]$
	Action estimate	Clipped flow endpoint
	Supervision target	Noisy-space search direction
Steering field	Hidden layers	$(256, 256)$
	Maximum training updates	5,000
	Batch size	512
	Validation fraction	10% of anchors
Sampling	Euler integration steps	16
	Guidance strengths evaluated	$\{0, 0.05, 0.2, 0.5\}$

The search radius scales with chunk length and flow time as $\rho_t = c\sqrt{L}(1 - t)$, where L is the number of actions in a chunk.

C TOY EXAMPLE SETUP

Figure 2 uses an equally weighted mixture of eight Gaussians in two dimensions, with covariance $.28^2 I_2$ and centers $c_m = 2.4(\cos\theta_m, \sin\theta_m)$, where $\theta_m = \pi/8 + m\pi/4$ and $m = 0, \ldots, 7$. Assign each action to its nearest center, $m(a) = \arg\min_m \|a - c_m\|$, and define

$$G(a) = \mu(a) + \xi s_{m(a)}, \qquad \mu(a) = 1 - .05m(a) - \frac{\left\|a - c_{m(a)}\right\|^2}{2(.65)^2}, \quad \Pr(\xi = \pm 1) = \tfrac{1}{2}. \tag{26}$$

The constant, opposed, and aligned variability profiles are, respectively, $s_m = .35$, $s_m = .05 + .60m/7$, and $s_m = .65 - .60m/7$.

For this return model, the desirability score and reweighted reference density are

$$U_\beta(a) = \mu(a) + \frac{\log\cosh(\beta s_{m(a)})}{\beta}, \qquad q_{\beta,\eta}(a) \propto p_{\mathrm{ref}}(a) e^{\eta U_\beta(a)}, \tag{27}$$

with $U_0 = \mu$. Writing p_m for the mass of $q_{\beta,\eta}$ in the nearest-center cell m, the heatmaps report $N_{\mathrm{eff}} = \exp(-\sum_m p_m \log p_m)$. We integrate on a 601×601 grid over $[-3.8, 3.8]^2$, using $\beta = 0, .1, \ldots, 8$ and $\eta = 0, .5, \ldots, 64$.

This calculation concerns ideal reweighting, not a trained policy. Within a single cell, the desirability bonus is constant and cancels from standardized local score differences.