RFS: REINFORCEMENT LEARNING WITH RESIDUAL FLOW STEERING FOR DEXTEROUS MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Imitation learning has been an effective tool for bootstrapping sequential decision making behavior, showing surprisingly strong results as methods are scaled up to high-dimensional, dexterous problems in robotics. These "behavior cloning" methods have been further bolstered by the integration of generative modeling techniques such as diffusion modeling or flow matching for training expressive multimodal behavior policies. However, these pretrained models do not always generalize perfectly, and require finetuning to maximize deployment-time performance. This finetuning procedure must retain the strengths of pretraining for exploration, while being able to quickly correct for local inaccuracies in model performance. In this work, we propose an efficient reinforcement learning (RL) framework for fast adaptation of pretrained generative policies. Specifically, our proposed methodology - residual flow steering, instantiates an efficient RL technique that quickly adapts a pretrained flow-matching model by steering it jointly by optimizing a policy for selecting both a latent noise distribution and a residual action. Doing so allows policies to perform both local (residual actions) and global exploration (latent noise), data-efficient adaptation. We demonstrate that this technique is effective for dexterous manipulation problems, serving both as a tool to pretrain behaviors in simulation and efficiently finetune them in the real world. Website is here [1].

1 INTRODUCTION

Imitation learning from human data has proven to be a strong tool for obtaining performant policies in robotics and sequential decision making (Zhao et al., 2023; 2025; Chi et al., 2023; Hussein et al., 2017; Ho and Ermon, 2016). Of particular interest has been the success of *generative* imitation learning, where powerful generative modeling techniques such as diffusion models (Ho et al., 2020) or flow models (Lipman et al., 2022) have been adopted to model the distribution of human demonstrations. These models have been particularly effective due to the multimodality of human demonstrations (Chi et al., 2023; Black et al., 2024), allowing them to naturally scale with robust empirical performance (Team et al., 2025). However, imitation learning in itself is often insufficient to generalize to all possible test-time scenarios (Team et al., 2025), leaving considerable room for policy improvement with test-time finetuning algorithms to production-level success rates.

While supervised finetuning (Ouyang et al., 2022; Black et al., 2024) is effective for decision-making systems, it relies on high-quality, curated expert data. We seek methods that avoid this requirement by leveraging cheaper, suboptimal, or self-collected data. The conceptual framework of reinforcement learning (and offline reinforcement learning (Levine et al., 2020)) offers an appealing tool for this type of behavior improvement, as agents learn optimal behaviors by optimizing a reward maximization objective rather than relying on expert human data. While a plethora of techniques have been proposed to bootstrap imitation learning pre-training with reinforcement learning finetuning (Rajeswaran et al., 2018a; Hu et al., 2024a; Nair et al., 2018; Nakamoto et al., 2024), these methods primarily rely on the availability of closed-form likelihoods (Lillicrap et al., 2015) and reparameterization (Wang et al., 2019), and are thus not directly applicable to rich modern architectures based on diffusion or flow models. Furthermore, a significant challenge in this setting is striking the balance between finetuning for new problems and ensuring knowledge retention from the pre-training.

[1] https://residualflowsteering-png.github.io/residualflowsteering-png/

We specifically consider two promising classes of methods that are able to perform RL adaptation of generative imitative policies while avoiding the closed-form policy likelihood requirement - residual policy learning (Ankile et al., 2024b) and diffusion steering (Wagenmaker et al., 2025). While seemingly distinct, both are instances of a broader class of policy modulation methods that adapt pre-trained policies by modulating inputs or outputs rather than parameters. Residual learning is limited to local refinements through additive action corrections, making it difficult to achieve global behavioral shifts. Diffusion steering, by contrast, can induce significant global changes by modulating latent noise vectors, but struggles to adapt to behaviors outside the coverage of the imitation policy, limiting dexterous control.

In this work, we instantiate and study a new class of policy modulation algorithms, residual flow steering (RFS), that adapts pre-trained generative imitation learning policies to new scenarios with reinforcement learning. The premise of residual flow steering is to allow a reinforcement learning algorithm to modulate both the initial noise vector for a generative policy and provide an affine corrective transformation to the outputted action. This balances the semantic modifications needed for fast adaptation with the required local precision for dexterous manipulation. We introduce a general framework for RL finetuning with RFS and demonstrate its use for both policy pre-training and test-time adaptation under online and offline settings.

In this work, we instantiate residual flow steering for dexterous manipulation with multi-fingered hands. Due to the difficulty of collecting high-quality real-world data, we first develop controllers in simulation and show that residual flow steering yields robust multi-object grasping policies. These policies transfer reasonably well to the real world. They can be further refined by continuing to steer residual flow with limited human demonstration data, enabling data-efficient adaptation to real system dynamics while maintaining high precision and dexterity.

The contributions of this work are (1) We define the framework of residual flow steering, an efficient way of improving a pretrained flow matching policy, (2) We show that this framework provides an effective tool for pre-training in simulation, learning complex multifinger coordination behavior from human data, (3) We show that RFS can enable policy finetuning in the real world via offline reinforcement learning, more effectively than typical bootstrapped imitation learning alternatives, (4) We conduct detailed ablations to identify which RFS components are most critical for performance in simulation and the real world.

2 Related Work

Imitation-Bootstrapped Reinforcement Learning. Imitation learning (IL) has been widely used to bootstrap reinforcement learning (RL) for complex robotic tasks (Hu et al., 2024a) to constrain exploration and guide policies. (Nair et al., 2018) systematically uses demonstration for imitation loss, a replay buffer, and resetting to guide RL exploration. Additionally, (Ball et al., 2023), (Hester et al., 2017) balance demonstration data with online data to bootstrap exploration. Though (Rajeswaran et al., 2018b) investigates dexterous manipulation in simulation using a 24-DoF hand, it requires a large amount of demonstrations, and no real-world application is verified. While demonstrations aid exploration, they are often noisy and insufficient for guiding fine-grained dexterous motions.

Finetuning Diffusion-Based Policies. Reinforcement learning has recently been applied to generative models such as diffusion policies (Ren et al., 2024) and flow matching (Zhang et al., 2025). However, the iterative refinement structure of diffusion models and the large action spaces of action-chunked policies often make direct RL fine-tuning unstable (Ren et al., 2024; Uehara et al., 2024). To mitigate this, (Park et al., 2025) proposed training an expressive one-step policy with RL instead of directly optimizing the iterative flow. In contrast, we follow the approach of (Wagenmaker et al., 2025), which leverages reinforcement learning to steer exploration directly in the diffusion model's latent noise space. Yet, because steering is confined to the generative model's latent space, the policy may fail to generate effective actions outside this space, leading to errors in challenging states.

Residual Policy Learning Residual RL has been explored in various contexts. (Ankile et al., 2024a) pre-train an expressive policy on offline data and then refine it online using a residual policy. Residual learning has been applied to vision-based manipulation (Zeng et al., 2020) and to uncertain robotic environments (Silver et al., 2019), demonstrating strong improvements over imperfect policies in complex tasks. To improve stability, (Zhang et al., 2020) proposes a bidirectional target

network, while (Alakuijala et al., 2021) extends residual formulations to visual inputs and sparse rewards with demonstrations. (Davchev et al., 2022; Carvalho et al., 2022) combine residuals with structured primitives, enabling efficient residual corrections. Residual policies can refine base actions, but struggle when the base policy is poor. Our method addresses this by jointly refining the base with RL while applying residuals only where they are most effective.

3 BACKGROUND

3.1 FLOW MATCHING FOR GENERATIVE POLICY LEARNING

Flow matching (Lipman et al., 2023; Liu, 2022) is a generative modeling framework that learns a time-dependent velocity field transporting a base distribution p_0 (f(e.g., Gaussian or uniform) to a target data distribution p_1. Consider a random variable x_t evolving over $t \in [0,1]$ according to the ODE $\frac{dx_t}{dt} = v_\theta(x_t, t)$ where $v_\theta : \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ is a neural network parameterizing the velocity field. The goal of flow matching (Lipman et al., 2023) is to aligns the pushforward of the source distribution p_0 with the target distribution p_1 , while avoiding both ODE backpropagation and closed-form likelihoods by supervising conditional probability paths rather than intractable marginals (Lipman et al., 2023).

Mechanistically, to train v_θ with flow matching, one samples pairs $(x_0, x_1) \sim p_0 \times p_1$, picks a random $t \in [0,1]$, and defines an interpolant $x_t = (1-t)\, x_0 + t\, x_1$. The true "velocity" that transports x_t along this straight path to x_1 is $v^\star(x_t, t) = x_1 - x_0$. The flow-matching loss then minimizes the squared error between the predicted and true velocities:

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0 \sim p_0,\, x_1 \sim p_1,\, t \sim \mathcal{U}[0,1]} \big\| v_\theta(x_t, t) - v^\star(x_t, t) \big\|^2. \tag{1}$$

Given this learned velocity field v_θ, we can then perform sampling from p_1 by a straightforward numerical integration of the learned ODE defined by v_θ, through standard Euler (Lipman et al., 2023) integration:

$$x^{k+1} \;=\; x^k \;+\; \Delta t_k\, v_\theta(x^k, t_k), \;\; x^0 \sim p_0, \;\; x^K \approx x_1. \tag{2}$$

This can further be made conditional by learning $v_\theta(x_t, t, c)$, an external conditioning variable c. In our setting, given a (multimodal) expert human-provided dataset $\mathcal{D} = \{s_i, a_i\}_{i=1}^N$, we can use flow matching to learn a policy (parameterized as a velocity field). The flow matching training objective (Eq 3) for training a policy velocity field $v_\theta(a_t, t, s)$ is

$$\mathcal{L}(\theta) = \mathbb{E}_{a_0 \sim p_0,\, (s,a) \sim \mathcal{D},\, t \sim \mathcal{U}[0,1]} \big\| v_\theta(a_t, t, s) - (a - a_0) \big\|^2, a_t \quad = (1-t)\, a_0 + t\, a. \tag{3}$$

The learned velocity field $v_\theta(a_t, t, s)$ can then be used for sampling by integrating the flow field (Eq 2). This provides a simple way to model complex, multimodal action distributions, while also offering knobs for adaptation. For flow-based policies, we will use π_θ and v_θ interchangeably for notational convenience.

3.2 REINFORCEMENT LEARNING

For adaptation with reinforcement learning (RL), we build on the framework of Markov decision processes (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma)$. $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T(s' \mid s, a)$ is the transition kernel, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0,1)$ is a discount factor. The goal of reinforcement learning algorithms is to leverage interaction to learn a stochastic policy $\pi_\theta(a \mid s)$ that maximizes the expected discounted sum of rewards $J(\theta) \;=\; \mathbb{E}_{\tau \sim p_\pi}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. There is a huge variety of algorithms to optimize this objective (Sutton and Barto, 2018) (for instance, on-policy, off-policy, model-based, and hybrid variants), and we will not be prescriptive on which precise optimization algorithm is to be used. Instead, it is useful to consider two specific variants of imitation-bootstrapped RL frameworks that we will build on in this work:

Residual reinforcement learning: These methods aim to learn a small "residual" policy $\pi_r(a|s)$ with reinforcement learning that can correct for the mistakes of a so-called base policy $\pi_\theta(a|s)$ that

is trained with imitation learning (or variants such as flow matching). Residual RL methods (Ankile et al., 2024b; Silver et al., 2019) execute actions through an additive transformation on the executed actions, optimizing the following objective:

$$\max_{\pi_r} \mathbb{E}_{\substack{s_0\sim p_0(s), s'\sim p(s'|s,a) \\ a=a_r+a_b \\ a_r\sim\pi_r(a|s),\ a_b\sim\pi_\theta(a|s)}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]; \quad a = a_r + a_b, \quad a_r \sim \pi_r(a|s), a_b \sim \pi_\theta(a|s) \tag{4}$$

Diffusion steering on latent noise: A recently proposed alternative technique for adapting pretrained generative policies to test-time scenarios is diffusion-steering (Wagenmaker et al., 2025). These methods specifically apply to diffusion and flow-based policies where an initial latent "noise" vector is pushed forward through an integration process to generate an action. While sampling from the base policy is performed by choosing the initial latent "noise" vector from a unit Gaussian distribution, adaptation can be done by directly choosing the initial latent noise from a different distribution defined by $\pi_{\text{DS}}(a|s)$. Using $\text{Push}(s, a_0, v_\theta)$ as a shorthand for the integration process in Eq 2 for the learned flow field $v_\theta(a_t, s, t)$, the diffusion steering objective optimizes Eq 5

Intuitively this optimizes for policy performance by "steering" the base policy π_θ through modulation of the initial latent noise a_0, leading to a reweighting of the inferred action distribution within the coverage of the base policy v_θ.

$$\max_{\pi_{\text{DS}}} \mathbb{E}_{\substack{s_0\sim p_0(s),\ s'\sim p(s'|s,a) \\ a_0\sim\pi_{\text{DS}}(a_0|s),\ a=\text{Push}(s,a_0,v_\theta)}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] \tag{5}$$

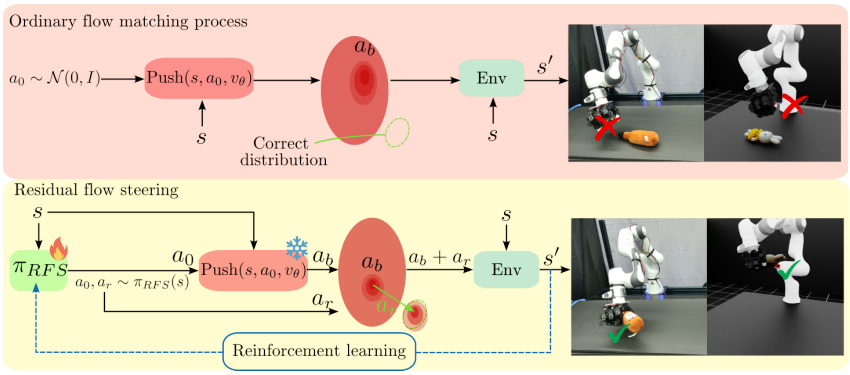

Figure 1: Overview of Residual Flow Steering (RFS). In ordinary flow matching, noise $a_0 \sim \mathcal{N}(0, I)$ is sampled and pushed through $\text{Push}(s, a_0, v_\theta)$ to generate an action a_b. RFS instead learns the initial noise distribution and introduces a residual action a_r, enabling direct shifts beyond the flow matching manifold. This allows actions to be steered toward the desired distribution using the residual policy.

4 Residual Flow Steering for Policy Adaptation

In this work, we propose **Residual Flow Steering (RFS)**, a technique that unifies latent steering for global adaptation with residual actions for fine-grained corrections, building on flow-matching pretraining. While residual RL (Ankile et al., 2024b; Silver et al., 2019) and latent noise adaptation (Wagenmaker et al., 2025) appear distinct, we show they are both instances of a broader class of *policy modulation* methods. Leveraging this connection, RFS enables efficient policy improvement for high-dexterity problems and supports simulation-to-reality transfer in dexterous manipulation.

4.1 Policy Modulation Algorithms for Adapting Pretrained Generative Policies

We begin by delving more carefully into the structure of both residual RL (Ankile et al., 2024b; Silver et al., 2019) and latent-noise steering (Wagenmaker et al., 2025; Du and Song, 2025) algorithms, given a generative pretrained policy $v_\theta(a_t, t, s)$ and a pushforward sampling function $a \sim \text{Push}(s, a_0, v_\theta); a_0 \sim \mathcal{N}(0, I)$. Let us put the two objectives side-by-side:

$$\max_{\pi_{\text{DS}}} \mathbb{E}_{\substack{s_0\sim p_0(s)\\ s'\sim p(s'|s,a)\\ a=\text{Push}(s,a_0,v_\theta)\\ a_0\sim\pi_{\text{DS}}(a_0|s)}} \left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] \quad (6) \qquad \max_{\pi_r} \mathbb{E}_{\substack{s_0\sim p_0(s)\\ s'\sim p(s'|s,a)\\ a=a_r+a_b\\ a_r\sim\pi_r(a|s)\\ a_b\sim\text{Push}(s,a_0,v_\theta);\ a_0\sim\mathcal{N}(0,I)}} \left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] \quad (7)$$

Figure 2: Side-by-side optimization objectives. Left: diffusion steering. Right: residual RL.

We observe that both objectives modulate the behavior of the base policy v_θ through different mechanisms (initial noise versus an affine transformation of the output). More generally, a policy modulation algorithm for a generative policy $v_\theta(a_t, t, s)$ is defined as an optimization procedure that aims to modulate the policy of v_θ through parametric input (g) or output (f) modifications, without modifying the actual parameters of the policy θ.

This general framework can encompass residual RL(Silver et al., 2019; Ankile et al., 2024a), latent noise steering(Wagenmaker et al., 2025; Singh et al., 2020), hierarchical skill adaptation (Hu et al., 2024b; Sun et al., 2025), and even a broader class of methods for finetuning diffusion models with RL Uehara et al. (2024). The differentiating factor between these methods is the particular choice of f and g in Eq 8.

$$\max_{\phi} \mathbb{E}_{\substack{s_0\sim p_0(s), s'\sim p(s'|s,a)\\ a=f_\phi(a_b,s),\ a_0=g_\phi(s)\\ a_b\sim\text{Push}(s,a_0,v_\theta)}} \left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] \quad (8)$$

Residual Flow Steering for Joint Local and Global Adaptation: In this work, we choose a particular instantiation of f and g, which we refer to as residual flow steering (RFS). Put simply, this performs efficient yet precise adaptation by bringing together the benefits of residual RL (Silver et al., 2019; Alakuijala et al., 2021) for output modulation, and diffusion steering RL Wagenmaker et al. (2025) for input modulation. By controlling the latent noise of a flow-matching policy, input modulation provides "global" changes in behavior actions, while output modulation provides the fine "local" changes needed for precise behavior through an affine transformation on the predicted push-forward actions.

Intuitively, consider the case of a robotic hand tasked with performing dexterous grasping (Fig. 1). Modulating the input noise changes the broad finger gaiting strategy itself, providing explore strategies. However, if the desired *precise* behavior is not covered under the base policy, then this is unrealizable, no matter what latent noise is chosen. In this often encountered scenario, where a small degree of "off-manifold" behavior is required, local changes can be made by an additional residual output transformation. Concisely, we instantiate output modulation f_ϕ as learning a residual action correction $a = \text{Push}(s, a_0, v_\theta) + a_r; a_r \sim \pi_r(a|s)$ and the input modulation g_ϕ as $a_0 = \pi_H(a_0|s)$. These can be combined for a high-level modulation policy $\pi_{RFS}(a_0, a_r|s)$ to jointly output both the latent initial noise a_0 (for global steering) and the residual action a_r (for local refinement):

$$\max_{\pi_{RFS}} \mathbb{E}_{\substack{s_0\sim p_0(s), s'\sim p(s'|s,a)\\ a_0,a_r\sim\pi_{RFS}(a_0,a_r|s)\\ a_b\sim\text{Push}(s,a_0,v_\theta)\\ a=a_r+a_b}} \left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right] \quad (9)$$

In this formulation, the policy outputs two components: a_0 for **global adaptation**, steering the generative model toward coherent, task-relevant motions, and a_r for **local corrections**, compensating for misalignments from environment variability, controller artifacts, or limited coverage in the base policy v_θ. The objective in Eq. 9 can be optimized with various reinforcement learning algorithms (off-policy, on-policy, or model-based). In Section 5, we present a specific instantiation of RFS for dexterous manipulation, enabling efficient pretraining in simulation and finetuning in the real world.

5 Applications of Residual Flow Steering to Problems in Dexterous Manipulation

In this section, we instantiate residual flow steering (RFS) for dexterous manipulation. Since safe, large-scale real-world training is challenging, we adopt a simulation-to-reality approach: pre-train policies using inexpensive simulated data, then fine-tune with limited real-world data. RFS proves effective both for exploration in simulation and for data-efficient real-world finetuning. For simplicity, we focus on dexterous grasping, though the method applies more broadly.

5.1 Data Generation with Residual Flow Steering in Simulation

The primary challenge in generating dexterous data is efficient exploration in high-dimensional spaces. RFS addresses this by seeding exploration with a small set of VR-teleoperated demonstrations, which train a base policy $v_\theta(a_t, s, t)$ via a flow-matching objective (Lipman et al., 2022). Although not broadly successful, this policy captures meaningful hand–arm motions and can be finetuned with RFS. As described in Section 4.1, a high-level policy π_{RFS} is then trained atop v_θ to optimize task-specific rewards such as grasp success and stability (task and reward details in the Appendix A.2.1).

For simulation data generation, the RFS policy π_{RFS} outputs (a_0, a_r): (1) **Latent noise** a_0 steers the pretrained policy v_θ toward globally diverse, task-relevant motions, while (2) **Residual actions** a_r provide fine-grained corrections for local misalignments. This framework refines joint hand–arm policies over both finger and end-effector actions. To mitigate VR data limitations, we also consider a variant (Section 6.1.2) where only finger motion is pretrained with flow matching, and arm motion is learned from scratch via reinforcement learning.

Using RFS, as described above, we can generate data with low-level privileged state information s (such as object positions, velocities) in simulation. This generated data can be distilled to point-cloud based policies with standard student-teacher methodology (Chen et al., 2021), parameterizing the distillation with visuomotor flow-matching policies $v_\phi(a_t, o_{\text{pc}}, s_{\text{pro}}, t)$, conditioned on point cloud (o_{pc}) and proprioception s_{pro}, producing hand-arm actions directly $a_t = (q_{\text{hand}}, q_{\text{arm}})$. The result is a pretrained policy that can be deployed and then adapted in the real world.

5.2 Real-World Policy Finetuning with Offline Residual Flow Steering

Directly transferring the distilled visuomotor policy $v_\phi(a_t, o_{\text{pc}}, s_{\text{pro}}, t)$ to the real world often fails, especially for unseen objects and initial conditions. To bridge this gap, we apply data-efficient finetuning with an *offline* variant of RFS, leveraging a small human-collected dataset $\mathcal{D} = \{((o,s),(a,a_b),(o',s'),r)_i\}_{i=1}^N$ that records base policy actions a_b, human corrections a, and resulting transitions. Our goal is to train $\pi_{RFS}(a_0, a_r \mid o_{\text{pc}}, s_{\text{pro}})$ via offline RL (Levine et al., 2020), maximizing reward when combined with the simulation-pretrained v_ϕ. For evaluation, we adopt TD3+BC (Fujimoto and Gu, 2021), which alternates actor–critic updates with an added imitation term. We detail both critic and actor updates below.

Critic Update: The challenge in using an offline RL algorithm on the dataset $\mathcal{D} = \{((o,s),(a,a_b),(o',s'),r)_i\}_{i=1}^N$ is that it does not have actions readily in the form (a_0, a_r) that is needed for residual flow-steering. To infer a dataset of this form, we can perform a simple transformation of the recorded actions (a, a_b). a, a_b, a_0 and a_r share the following relationships - $a = a_b + a_r$ and $a_b = \text{Push}(o, a_0, v_\theta)$. This suggests that to obtain (a_0, a_r) for RFS, we can trivially construct $a_r = a - a_b$. To obtain a_0, we need only invert the push operation $a_b = \text{Push}(o, a_0, v_\theta)$. While many techniques could be used here (such as flow inversion), we opt for a simple optimization based strategy to find the initial noise a_0 that best explains a_b when pushed forward - $a_0 \leftarrow \arg\min_{a_0} \|a_b - \text{Push}(o, a_0, v_\theta)\|$. These operations then allow us to convert the dataset $\mathcal{D}$ into a form directly amenable to RFS $\mathcal{D}_{RFS} = \{((o,s),(a_0,a_r),(o',s'),r)_i\}_{i=1}^N$. Given this dataset $\mathcal{D}_{RFS}$ we can then instantiate a standard TD-learning critic update as

$$\min_\phi \mathbb{E}_{\substack{((o,s),(a_0,a_r),(o',s'),r)\sim D_{RFS} \\ a_0',a_r'\sim\pi_{RFS}(a_0',a_r'|o',s') \\ a=a_b+a_r,\ \ a_b\sim\text{Push}(o,s,a_0,v_\phi) \\ a'=a_b'+a_r',\ \ a_b'\sim\text{Push}(o',s',a_0',v_\phi)}} \left[||Q_\phi(o,s,a) - r(s) - \gamma Q_{\bar{\phi}}(o',s',a')||^2 \right] \tag{10}$$

Note that we make the design to provide the critic with the combined actions a, a' rather than the separated actions (a_0, a_r), which we justify through an empirical comparison in Section 6.2

Actor Update: We follow the standard actor-update described in TD3+BC (Fujimoto and Gu, 2021), where the actor maximizes the Q-values while applying a behavior cloning (BC) regularization of the residual actions a_r against the offline data.

$$\arg\max_{\pi_{RFS}} \mathbb{E}_{\substack{((o,s),(a_0,a_r),(o',s'),r)\sim D_{RFS} \\ \hat{a}_0,\hat{a}_r\sim\pi_{RFS}(\cdot|o,s) \\ \hat{a}_b\sim\text{Push}(o,s,\hat{a}_0,v_\phi) \\ \hat{a}=\hat{a}_b+\hat{a}_r}} [Q(o,s,\hat{a}) - \lambda_{\text{BC}} \|\hat{a}_r - a_r\|^2] \tag{11}$$

6 EXPERIMENTS

We perform the experiments to answer the following research questions: **Q1**: Can residual refinement further improve grasp success in simulation? **Q2**: Can RFS policy boost the pretrained policy for finger motion by steering latent noise? **Q3**: Which design choices are most effective in both simulation and real-world settings? **Q4**: How latent noise augmentation will affect the performance in real world adaptation? As outlined in Section 5, we first evaluate RFS for simulation data generation and then for real-world adaptation. Our experiments focus on dexterous grasping(power[2] and pinch[3] with a multifingered hand, though the approach generalizes to broader tasks.

6.1 RFS AS A DATA GENERATION TOOL IN SIMULATION

Setup: To evaluate RFS for simulation data generation, we collected 1441 grasping demonstrations in IsaacLab (Mittal et al., 2023) using a Franka arm with a 16-DoF Leap Hand (Fig. 1). Demonstrations were recorded via an Apple Vision Pro AR device across diverse objects, from rigid bottles to plush items[4]. These were used to pretrain a generative base policy $v_\theta(a_t, t, s)$ with flow matching (Lipman et al., 2022). Although not broadly successful, this policy provides meaningful motion priors and is finetuned with RFS using PPO (Schulman et al., 2017), as detailed in Section 5.1. Additional environment and RL setup details are in Appendix A.2. Our focus is on evaluating whether RFS can produce robust grasping policies across varied objects compared to standard RL baselines.

Figure 3: Simulation and real objects used in our experiments.

6.1.1 BASELINES AND ABLATIONS

Baselines: We compare against several alternative methods, which reparameterize action spaces in different ways for reinforcement learning in simulation - (1) **Tabula-rasa RL**: We train standard on-policy RL with PPO using different hand-defined action spaces. We consider both *absolute joint pose* control, where actions specify absolute joint values, and *relative joint pose*, where actions specify relative changes, (2) **Action-Space Reduction with PCA**: To reduce action dimensionality, finger motions are compressed into a 4-D latent space using principal component analysis (PCA), following (Li et al., 2025). **Action Codebooks with VQ-VAE**: We consider using a VQ-VAE to learn a codebook of meaningful coordinated finger motions from the human collected data (Xue et al., 2025).

Ablations: To understand the benefits of the *combination* of both residual action refinement and latent-noise based steering, we conduct thorough ablations. We evaluate the following ablated variants of RFS - (1) **DSRL (Wagenmaker et al., 2025):** Allowing the RL policy to steer the latent noise of the base policy *without* residual actions, (2) **Residual RL:** Only allow the RL policy to output residual actions, without actually steering the latent noise injected into the base model.

To compare with these methods, we evaluate two different instantiations of our method: RFS with hand and RFS with hand and arm. RFS w/ hand performs generative pretraining on the hand only, while RFS w/ hand and arm performs generative pretraining on the whole arm and hand data collected by human demonstrators. We also evaluate the efficacy of predicting action chunks (RFS-Chunk) rather than one-step actions.

[2]Power grasp uses the entire hand to firmly hold an object.
[3]Pinch grasp uses the thumb and one or two fingers to grip an object.
[4]Plush items are modeled as rigid bodies in simulation.

6.1.2 Empirical Results in Simulation

Comparison to Baselines: As shown in Fig. 4 and Tab. 1, the **absolute joint pose** fails due to large, unstable finger motions and penetration between fingers, while the **relative joint pose** struggles on pinch grasps, with 55% of motions infeasible under the penetration metric (Wang et al., 2023) (vs. 5% for ours; see Appendix A.2.3). The **PCA** baseline fails on power grasps, often losing fine-grained details such as early finger closure. The **VQ-VAE** baseline also fails on power grasps, producing jittery, poorly localized, and unnatural motions;(see Appendix A.2.3 for more analysis). In conclusion, stable, scalable grasping requires structured exploration and smooth, continuous motion representations. Additional evaluation images are provided in the Appendix A.2.

Comparison to Ablations: As shown in Tab. 1, RFS outperforms all ablations, highlighting the importance of combining residual and flow-steering components. DSRL (Wagenmaker et al., 2025) performs on par with PCA, VQ-VAE, and RL-from-scratch baselines, while RFS consistently surpasses them through global exploration via flow steering and precise local refinement with residual actions. Moreover, RFS is more sample-efficient, requiring fewer interactions to match or exceed baseline performance (Fig. 4). We also find that the hand-only setting outperforms hand–arm training, likely due to noise and redundancy in teleoperation data. Finally, the chunked variant (RFS-Chunk), which executes consecutive hand actions with residual refinement, further improves performance by enforcing smoother motion continuity and temporal consistency.

Given data generated via RFS in simulation on privileged states, we distill a point cloud based policy via standard student-teacher methods (Chen et al., 2021) for real world deployment. Using ∼1,800 simulated demonstrations, the distilled policy achieves a 95% success rate in simulation. However, this success rate drops considerably on real world deployment, necessitating real-world finetuning with RFS.

Method	Overall	Pinch Grasp	Power Grasp
absolute joint psose	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
relative joint pose	0.491 ± 0.491	0.000 ± 0.000	0.982 ± 0.003
PCA	0.394 ± 0.444	0.838 ± 0.017	0.000 ± 0.000
VQ-VAE	0.419 ± 0.419	0.714 ± 0.074	0.000 ± 0.000
Residual RL w/ hand & arm	0.174 ± 0.062	0.114 ± 0.005	0.234 ± 0.020
DSRL w/ hand & arm	0.625 ± 0.309	0.253 ± 0.062	0.874 ± 0.048
RFS w/ hand & arm (Ours)	0.708 ± 0.247	0.485 ± 0.153	0.930 ± 0.021
DSRL w/ hand	0.878 ± 0.045	0.833 ± 0.030	0.923 ± 0.080
RFS w/ hand (Ours)	0.937 ± 0.028	0.914 ± 0.012	0.959 ± 0.020
RFS-Chunk w/ hand (Ours)	$\mathbf{0.959 \pm 0.030}$	$\mathbf{0.930 \pm 0.030}$	$\mathbf{0.988 \pm 0.060}$

Table 1: Success rates (mean ± std) of different baselines across Overall (Pinch + Power), Power Grasp, and Pinch Grasp tasks for three random seeds.

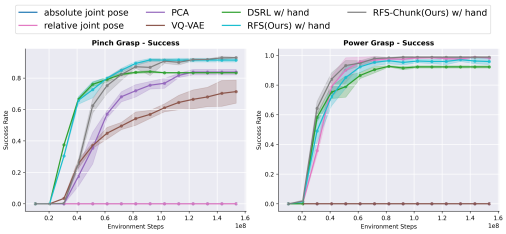

Figure 4: Comparison of different PPO-based baselines on two grasping tasks: pinch grasp (left) and power grasp (right).

6.2 Real World Results and Analysis

Next, we consider adapting distilled policies with offline RFS on a real-world robot setup shown in Fig. 1, for grasping both seen and unseen objects Fig.3. Real-world experiments were conducted using a Franka arm with a LEAP hand using a Cartesian impedance control running at 10Hz. For evaluation, we tested seven objects—two previously seen in simulation and five novel deformable objects. Unlike rigid simulation objects, real-world objects exhibit varying compliance, contact dynamics and appearance, introducing a sim-to-real gap that causes pre-trained policies to fail Appendix A.4.3). For offline RL finetuning, we collected 50 offline demonstrations using a SpaceMouse (Appendix A.4.1) for action correction. We programatically defined rewards by leveraging SAM2 (Ravi et al., 2024) to track the object and then extract its centroid (details in Appendix A.4.2).

6.2.1 Baselines and Ablations

Baselines: We evaluate RFS with offline RL (Section 5.2) as a data-efficient and performant method for finetuning policies from simulation. We compare RFS to a few different class of baseline adaptation methods (1) **Zero-Shot Transfer**, where the distilled policy from simulation is deployed directly without fine-tuning, (2) **BC Fine-Tuning**, Collecting and using 50 real-world demonstrations to finetune the distilled policy from simulation with flow matching, (3) **Co-Training**, performing finetuning of the sim-distilled policy with flow matching, but mixing 50 real demonstrations with

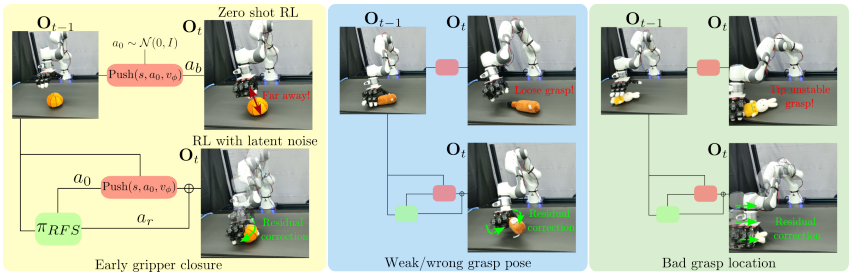

Figure 5: Illustration of three typical failure modes during real-world manipulation: (left) early hand closure, (middle) weak or incorrect grasp pose, and (right) poor grasp location. The top row shows the failure cases when using only the base policy, while the bottom row shows the corrected outcomes after applying our Residual Flow Steering (RFS) method. RFS effectively mitigates these errors, leading to more stable and successful grasps.

collected simulation data. We will show that using offline RL for finetuning rather than just standard flow matching based finetuning is more performant.

Critic Design Choices: Besides these comparisons to supervised fine-tuning, we also compare with different ablations and offline RL variants. Specifically, we compare three variants for critic design in offline-RL: (1) $Q(a_r, o)$: The critic evaluates the residual action res_a only and the observation o. (2) $Q([a_r, a_b], o)$: The critic takes as input the concatenation of the residual action and the base policy action, allowing evaluation of both, (3) $Q(a_b + a_r, o)$ **(Ours)**: The critic evaluates only the final executed action, which is the sum of the base policy output and the residual correction.

Ablations: We also perform ablative analysis of performing offline RL with just residual actions Residual RL, as well as offline RL with only latent noise steering DSRL.

6.3 Empirical Results in the Real World

Comparison to Supervised Fine-Tuning: As shown in Fig. 6, the zero-shot policy performs poorly, with failures such as translational offset, rotational offset, and insufficient finger tightness, highlighting a significant sim2real gap. Though co-training substantially improves performance on known objects (83.3% success), generalization to unseen objects remains poor. The noisy and inconsistent nature of human-collected data leads to instability and rollout failures, underscoring the limited generalization for novel objects.

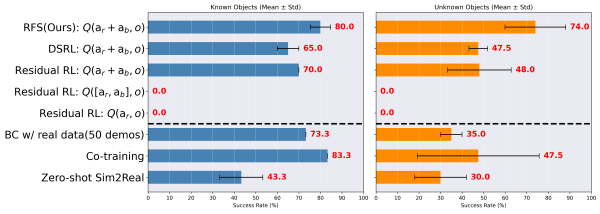

Figure 6: Real-world evaluation results. Bars and red values indicate mean success rates, with error bars showing the standard deviation.

Comparison to Variants of Offline RL: As discussed in Sec. 5.2, TD3+BC uses the actor output as input to the critic, leaving several options for how to represent the Q-function. We evaluate policies trained with three such Q-function designs(Fig. 6. Both $Q(a_r, o)$ and $Q([a_r, a_b], o)$ fail to generate consistent stable grasp poses. In contrast, $Q(a_b + a_r, o)$ achieved significantly higher performance over all baselines. This shows that coupling residuals directly with executed base actions is beneficial to the final performance.

Comparison to Ablations of RFS: We compared against variants using only residual actions ($\pi(a_r|o)$) and only DSRL latent noise steering ($\pi(a_0|o)$). As shown in Fig. 6, our full model $\pi(a_0, a_r|o)$ achieves the best performance, particularly on unseen objects. Qualitatively, this is because latent noise steering guides global exploration, while residual actions refine local adjustments, making their combination more effective than either alone.

7 Conclusion

We introduced Residual Flow Steering (RFS), a reinforcement learning framework that unifies latent noise steering for global exploration with residual corrections for local refinement. Our results across simulation and real-world dexterous grasping show that RFS substantially outperforms conventional baselines, including residual-only and diffusion-steering approaches, by enabling efficient pretraining in simulation and data-efficient finetuning in the real world.

Reproducibility Statement:

We have taken several steps to ensure the reproducibility of our results. The main text (Sections 4 and 5) provides details of our proposed Residual Flow Steering (RFS) framework, including algorithmic design, training objectives, and ablation studies. Appendix A.2 describes the simulation setup, reinforcement learning design choices, and evaluation metrics in detail, while Appendix A.3 provides descriptions of real-world experiments, reward specification, and data collection protocols. To further facilitate replication, we include more details for the baseline methods, hyperparameter choices, and ablation analyses in the appendix for both the simulation and the real experiment.

REFERENCES

Minttu Alakuijala, Gabriel Dulac-Arnold, Julien Mairal, Jean Ponce, and Cordelia Schmid. Residual reinforcement learning from demonstrations, 2021. URL https://arxiv.org/abs/2106.08050.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement – residual rl for precise assembly, 2024a. URL https://arxiv.org/abs/2407.16677.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement - residual RL for precise visual assembly. *CoRR*, abs/2407.16677, 2024b. doi: 10.48550/ARXIV.2407.16677. URL https://doi.org/10.48550/arXiv.2407.16677.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data, 2023. URL https://arxiv.org/abs/2302.02948.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Joao Carvalho, Dorothea Koert, Marek Daniv, and Jan Peters. Residual robot learning for object-centric probabilistic movement primitives, 2022. URL https://arxiv.org/abs/2203.03918.

Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pages 297–307. PMLR, 2021. URL https://proceedings.mlr.press/v164/chen22a.html.

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

Todor Davchev, Kevin Sebastian Luck, Michael Burke, Franziska Meier, Stefan Schaal, and Subramanian Ramamoorthy. Residual learning from demonstration: Adapting dmps for contact-rich manipulation. *IEEE Robotics and Automation Letters*, 7(2):4488–4495, April 2022. ISSN 2377-3774. doi: 10.1109/lra.2022.3150024. URL http://dx.doi.org/10.1109/LRA.2022.3150024.

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

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep q-learning from demonstrations, 2017. URL https://arxiv.org/abs/1704.03732.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, pages 4565–4573, 2016. URL https://papers.nips.cc/paper/6391-generative-adversarial-imitation-learning.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 6840–6851, 2020. URL https://arxiv.org/abs/2006.11239.

Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning, 2024a. URL https://arxiv.org/abs/2311.02198.

Jiaheng Hu, Zizhao Wang, Peter Stone, and Roberto Martín-Martín. Disentangled unsupervised skill discovery for efficient hierarchical reinforcement learning, 2024b. URL https://arxiv.org/abs/2410.11251.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2), April 2017. ISSN 0360-0300. doi: 10.1145/3054912. URL https://doi.org/10.1145/3054912.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020. URL https://arxiv.org/abs/2005.01643.

Kailin Li, Puhao Li, Tengyu Liu, Yuyang Li, and Siyuan Huang. Maniptrans: Efficient dexterous bimanual manipulation transfer via residual learning, 2025. URL https://arxiv.org/abs/2503.21860.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015. URL https://api.semanticscholar.org/CorpusID:16326763.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. URL https://arxiv.org/abs/2210.02747.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport, 2022. URL https://arxiv.org/abs/2209.14577.

Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh 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.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations, 2018. URL https://arxiv.org/abs/1709.10089.

Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning, 2024. URL https://arxiv.org/abs/2303.05479.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022. URL https://arxiv.org/abs/2203.02155.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning, 2025. URL https://arxiv.org/abs/2502.02538.

Younghyo Park and Pulkit Agrawal. Using apple vision pro to train and control robots, 2024. URL https://github.com/Improbable-AI/VisionProTeleop.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations, 2018a. URL https://arxiv.org/abs/1709.10087.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations, 2018b. URL https://arxiv.org/abs/1709.10087.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. URL https://arxiv.org/abs/2408.00714.

Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization, 2024. URL https://arxiv.org/abs/2409.00588.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning, 2019. URL https://arxiv.org/abs/1812.06298.

Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning, 2020. URL https://arxiv.org/abs/2011.10024.

Jiankai Sun, Aidan Curtis, Yang You, Yan Xu, Michael Koehle, Qianzhong Chen, Suning Huang, Leonidas Guibas, Sachin Chitta, Mac Schwager, and Hui Li. Arch: Hierarchical hybrid learning for long-horizon contact-rich robotic assembly, 2025. URL https://arxiv.org/abs/2409.16451.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018. ISBN 978-0-262-03924-6. URL http://incompleteideas.net/book/the-book-2nd.html.

TRI LBM Team, Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, Naveen Kuppuswamy, Kuan-Hui Lee, Katherine Liu, Dale McConachie, Ian McMahon, Haruki Nishimura, Calder Phillips-Grafflin, Charles Richter, Paarth Shah, Krishnan Srinivasan, Blake Wulfe, Chen Xu, Mengchao Zhang, Alex Alspach, Maya Angeles, Kushal Arora, Vitor Campagnolo Guizilini, Alejandro Castro, Dian Chen, Ting-Sheng Chu, Sam Creasey, Sean Curtis, Richard Denitto, Emma Dixon, Eric Dusel, Matthew Ferreira, Aimee Goncalves, Grant Gould, Damrong Guoy, Swati Gupta, Xuchen Han, Kyle Hatch, Brendan Hathaway, Allison Henry, Hillel Hochsztein, Phoebe Horgan, Shun Iwase, Donovon Jackson, Siddharth Karamcheti, Sedrick Keh, Joseph Masterjohn, Jean Mercat, Patrick Miller, Paul Mitiguy, Tony Nguyen, Jeremy Nimmer, Yuki Noguchi, Reko Ong, Aykut Onol, Owen Pfannenstiehl, Richard Poyner, Leticia Priebe Mendes Rocha, Gordon Richardson, Christopher Rodriguez, Derick Seale, Michael Sherman, Mariah Smith-Jones, David Tago, Pavel Tokmakov, Matthew Tran, Basile Van Hoorick, Igor Vasiljevic, Sergey Zakharov, Mark Zolotas, Rares Ambrus, and Kerri Fetzer-Borelli. A careful examination of large behavior models for multitask dexterous manipulation. *arXiv preprint arXiv:2507.05331*, 2025. URL https://arxiv.org/abs/2507.05331.

Masatoshi Uehara, Yulai Zhao, Tommaso Biancalani, and Sergey Levine. Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review, 2024. URL https://arxiv.org/abs/2407.13734.

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

Huan Wang, Stephan Zheng, Caiming Xiong, and Richard Socher. On the generalization gap in reparameterizable reinforcement learning, 2019. URL https://arxiv.org/abs/1905.12654.

Ruicheng Wang, Jialiang Zhang, Jiayi Chen, Yinzhen Xu, Puhao Li, Tengyu Liu, and He Wang. Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation, 2023. URL https://arxiv.org/abs/2210.02697.

Han Xue, Jieji Ren, Wendi Chen, Gu Zhang, Yuan Fang, Guoying Gu, Huazhe Xu, and Cewu Lu. Reactive diffusion policy: Slow-fast visual-tactile policy learning for contact-rich manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics, 2020. URL https://arxiv.org/abs/1903.11239.

Shangtong Zhang, Wendelin Boehmer, and Shimon Whiteson. Deep residual reinforcement learning, 2020. URL https://arxiv.org/abs/1905.01072.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning, 2025. URL https://arxiv.org/abs/2505.22094.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023. URL https://arxiv.org/abs/2304.13705. ALOHA: Action Chunking with Transformers (ACT).

Tony Z. Zhao, Jonathan Tompson, Danny Driess, Pete Florence, Seyed Kamyar Seyed Ghasemipour, Chelsea Finn, and Ayzaan Wahid. Aloha unleashed: A simple recipe for robot dexterity. In *Proceedings of The 8th Conference on Robot Learning (CoRL)*, volume 270 of *Proceedings of Machine Learning Research*, pages 1910–1924. PMLR, 2025. URL https://proceedings.mlr.press/v270/zhao25b.html.

A APPENDIX

A.1 THE USE OF LARGE LANGUAGE MODELS

In this paper, we primarily use LLMs for language polishing, making our writing more concise and accessible. We provide raw drafts and ask the LLM to refine them. We also use LLMs for guidance on figure preparation, such as creating illustrations in Inkscape and adjusting figure size and font in Matplotlib.

A.2 SIMULATION SETUP AND TRAINING

A.2.1 REINFORCEMENT LEARNING DESIGN CHOICES

To enhance robustness and mitigate controller misalignment for sim-to-real transfer, we introduce the following design choices:

Arm Initialization: Randomize the initial joint configuration of the arm at the start of each episode to improve generalization.

External Disturbances: Apply random external disturbances to each joint at intervals of 2–5 steps, encouraging the policy to recover from perturbations.

Observation Space: In the simulation RL training, we use state-based information, including the robot proprioception, the current object pose, the target object pose, and binary contact signals between each fingertip and the object.

Action Space: The action space in our RL training depends on the specific method used for hand motion generation, with a detailed comparison provided in Tab. 2. Across all methods, the arm is controlled using a 6D delta pose controller (3D translation + 3D rotation). The delta translational range is constrained to $[-0.03, 0.03]$ m per step, while the delta rotational range is limited to $[-0.1, 0.1]$ rad per step. The control frequency in simulation is set to 20 Hz.

Reward Function: The reward consists of three stages: encouraging finger–object proximity to guide fingers toward the object, promoting stable fingertip-palm contact for secure grasping, and encouraging successful lifting to a target height while penalizing unsafe behavior such as excessive lifting or joint limit violations.

Success Indicator: A grasp is deemed successful if the object is lifted at least 20 cm above the table while the end-effector maintains a feasible pose without unnatural or unstable joint configurations. Each checkpoint is evaluated over 1,024 episodes.

Data Collection: We use the Vision Pro for data collection, leveraging an application built on IsaacLab Mittal et al. (2023).

A.2.2 CFM TRAINING IN SIMULATION

In simulation, we trained two flow matching policies: one using only finger joint poses as observations, and another using both finger and arm actions with object pose and robot proprioception. Both policies predict the following two consecutive actions.

Latent noise motion: Latent noise exploration generates diverse hand motions from the same initial configuration, enabling adaptation to varied settings (Fig. 7). Furthermore, our method converges faster and achieves higher normalized rewards than baselines (Fig. 4), highlighting both efficiency and robustness.

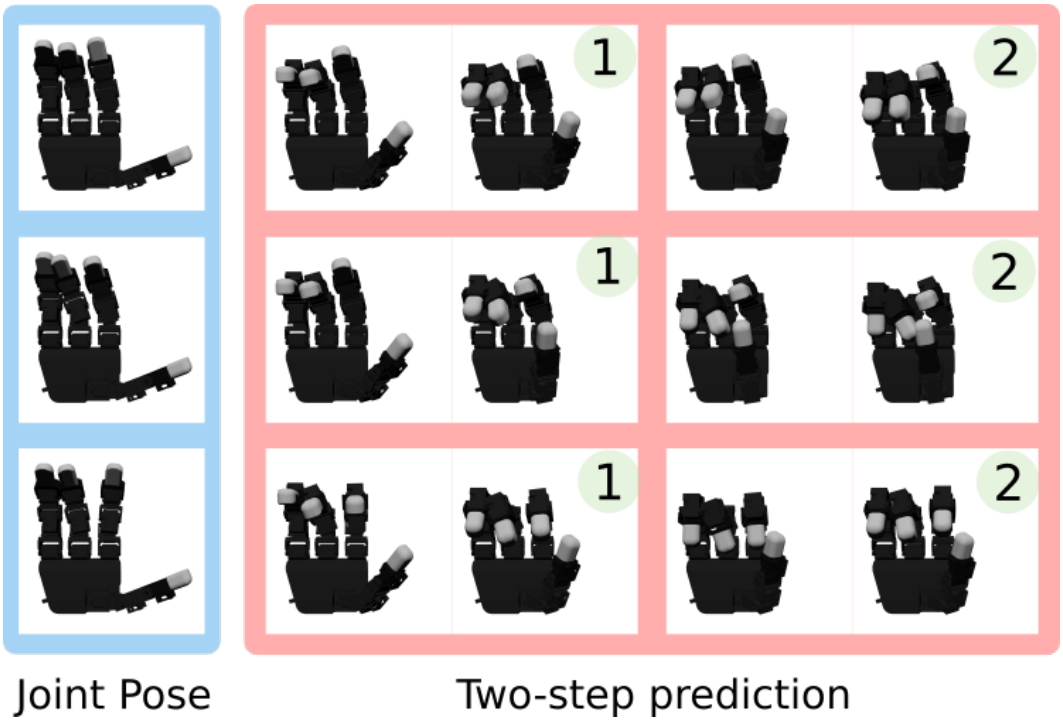

Figure 7: Illustration of latent noise conditioning on hand motion. The left column (blue) shows the given joint poses. The right columns (red) show two-step predictions from the same initial pose, where labels **1** and **2** indicate different latent noise conditions.

Table 2: Comparison of action dimension across different methods.

Method	**Action Dimension**
Relative joint pose	6 (delta arm pose) + 16 (relative finger joint pose)
Absolute joint pose	6 (delta arm pose) + 16 (absolute finger joint pose)
PCA	6 (delta arm pose) + 4 (PCA latent codes)
VQ-VAE	6 (delta arm pose) + 8 (VQ codes) $\times 2$ (horizon)
Residual RL (w/ arm & hand)	22 (residual actions)
DSRL (w/ arm & hand)	23×2 (horizon)
RFS (Ours, w/ arm & hand)	23×2 (horizon) + 22 (delta pose)
DSRL (w/ hand)	6 (delta arm pose) + 16×2 (horizon)
RFS/RFS-Chunk (Ours, w/ hand)	6 (delta arm pose) + 16×2 (horizon) + 16 (delta pose)

A.2.3 BASELINES

Tabula-Rasa RL: While Tabula-Rasa RL with relative joint poses achieves a success rate comparable to our method, a closer inspection of the finger motions (Fig. 9) reveals severe inter-finger penetration. To quantify this, we follow (Wang et al., 2023) and use Kaolin to compute the penetration depth between fingers. If the penetration exceeds 0.5 cm, we regard the grasp as infeasible. Under this criterion, although the relative joint pose policy reports a 98% success rate, at least 55% of the motions are infeasible, compared to only 5% with our method.

PCA: We reduce the high-dimensional finger actions into four latent values using PCA, and employ RL to explore this latent space. The raw demonstration data is used to define the minimum and maximum latent values, and RL actions are scaled accordingly. As shown in Fig. 4, PCA achieves performance comparable to DSRL on pinch grasps; however, for power grasps (Fig. 9), the thumb closing timing remains misaligned.

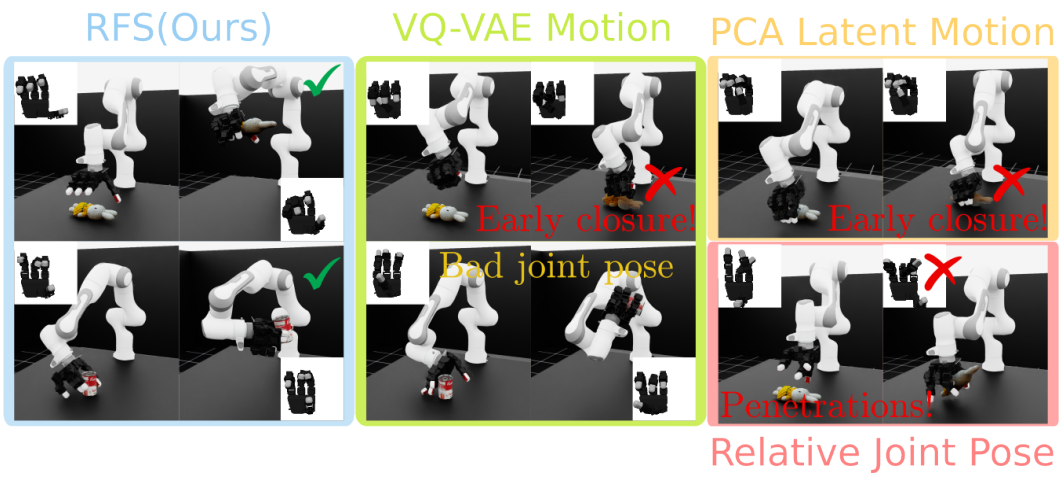

Figure 9: Simulation rollouts comparing our proposed method (RSF-Chunk) with baseline approaches (Tabula-rasa RL, PCA, and VQ-VAE). Our method produces more stable and collision-free hand motions, whereas Tabula-rasa RL frequently suffers from severe finger.

VQ-VAE: Following (Xue et al., 2025), we adopt a 1D-CNN encoder with a GRU decoder, where the model predicts the following two finger motions conditioned on the previous joint pose. During RL training, we steer the latent space and add residual actions for fine-grained correction, applying the same residual adjustment across two steps to ensure temporal consistency. As shown in Tab. 1, the VQ-VAE achieves a relatively high success rate in simulation for pinch grasps, but fails on power grasps (e.g., pushing). Fig. 9 further illustrates that the thumb often closes prematurely before reaching the object, resulting in misalignment and making training more challenging.

A.2.4 Robustness Analysis

To evaluate the robustness and stability of grasp motions, we conducted robustness analysis by gradually applying external forces and torques to each joint and measuring the success rate (Fig. 8). Since not all baseline methods can solve the task, we only compare their feasible grasp motions with our method. In contrast, our method is evaluated across all grasping tasks and consistently maintains a higher success rate under larger perturbations, demonstrating greater robustness and flexibility of the generated motions.

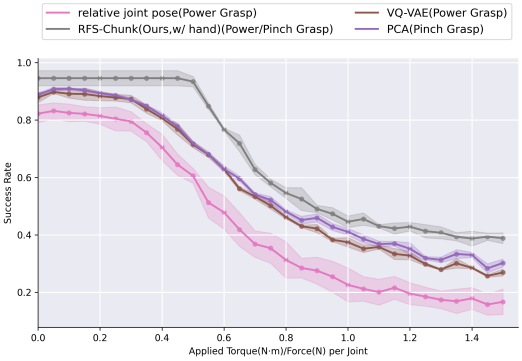

Figure 8: Robustness evaluation: We evaluate the robustness of our RFS method against baseline approaches under varying levels of external disturbances.

A.2.5 Policy Distillation

For better sim-to-real transfer, we use the point cloud as the visual observation. To offset deviations caused by camera calibration and hardware settings, we collect simulation data from multiple camera angles and calibrate the point cloud to the robot base. Additionally, random noise is added to the camera transformation matrix during data collection and to the point cloud during training, improving robustness. In the simulation, we have collected 1,800 demonstrations for the policy training.

A.3 Real World Experiment Details

A.3.1 Real Robot evaluation

For real robot evaluation, we tested our policy within a 40×20 cm region (Fig. 12), located 0.45–0.65 m from the robot base and spanning -0.25 to 0.25 m horizontally. We evaluated two known objects with 20 trials each, and seven unknown objects with 10 trials each.

A.4 Teleoperation Data Collection

For real-world teleoperation data collection in the co-training setup, we developed a Vision Pro application built on top of Park and Agrawal (2024), supporting teleoperation with a controlled frequency of 10Hz.

A.4.1 Offline data collection

When the hand is within $\sim$10 cm of the table—where most failures occur (Fig. 5), we enable human intervention via a SpaceMouse (Fig. 11). Rather than granting full manual control, which would shift the policy distribution, we compute residual corrections as bounded deltas from the base policy output, conditioned on the current observation. Specifically, the CFM module predicts the next action, and we take the difference between this rollout and the operator's input. Residuals are constrained to $\leq$1.5 cm in Cartesian translation and $\leq$0.05 rad for finger motion, applied uniformly across all joints. In practice, corrections were limited to Cartesian translation and minor finger adjustments, with the SpaceMouse z-axis mapped to finger motion.

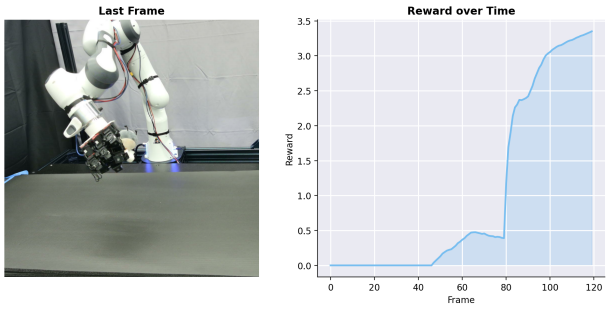

Figure 10: Reward progression over time in real-world experiments.

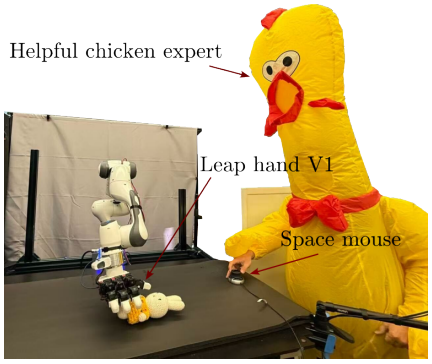

Figure 11: Offline real-world data collection setup using the Leap Hand V1 and a space mouse interface.

A.4.2 Reward Function Design

In the real world, the reward is constructed using SAM2 (Ravi et al., 2024) to track the object in image space and extract its point cloud, from which the object center is computed. The reward consists of two terms: (1) the distance between the object center and the palm, obtained via forward kinematics, and (2) the object's lifting height, which encourages stable grasp execution. Figure 10 illustrates the reward trend.

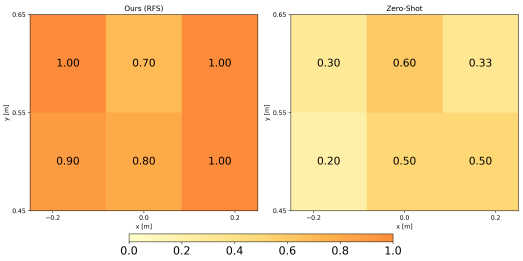

Figure 12: We evaluate the success rate of the bunny plush placement in different regions of the table using our method and a zero-shot transfer policy.

A.4.3 Failure analysis

As shown in Fig. 5, a common failure mode occurs when the fingers fail to grasp the object tightly. This arises for two main reasons: (1) the

sim-to-real gap in actuator dynamics and controller behavior for the Leap Hand, and (2) the mismatch in object properties, as plush objects are modeled as rigid bodies in simulation but are much softer in the real world, requiring greater force for a stable grasp. These discrepancies highlight the necessity of real-world adaptation to bridge the gap between simulation and deployment.

A.4.4 Regional Robustness Analysis in Real-World Experiments

To better understand whether failures in real-robot tasks arise from environmental randomness or policy limitations, we conducted an experiment measuring the bunny grasp success rate across different segmentation regions using our RFS method. As shown in Fig. 12, RFS achieves consistently higher success rates across a broader region compared to the zero-shot baseline, indicating improved robustness and reliability.