RECFLOW POLICY: FAST AND ACCURATE VISUOMOTOR POLICY LEARNING VIA RECTIFIED ACTION FLOW

Rong Xue*, Jiageng Mao*, Mingtong Zhang & Yue Wang
Department of Computer Science
University of Southern California
Los Angeles, USA
{rx_452,jiagengm,zhangm94,yue.w}@usc.edu
*Equal Contribution

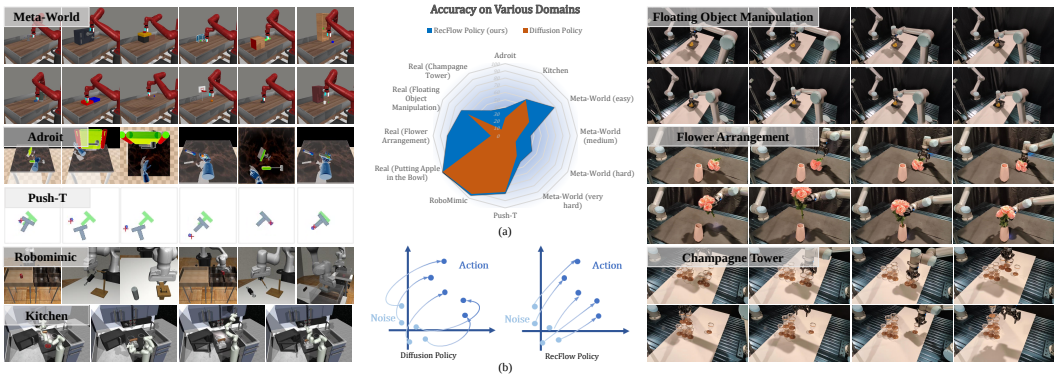


Figure 1: **Rectified Flow Policy** is a visual imitation learning algorithm that utilizes rectified flow with selective refinements, achieving superior effectiveness in diverse simulation and real-world tasks, with a significant inference acceleration. (a) Accuracy on various domains. (b) Sampling flow from noise to action of RecFlow Policy and Diffusion Policy.

ABSTRACT

We introduce RecFlow Policy, a fast, accurate, and scalable policy for robot learning, bridging the gap between generative modeling techniques and real-world robotic applications. Diffusion models have seen rapid adoption in robotic imitation learning, enabling autonomous execution of complex dexterous tasks. However, the dependence of multi-step iterative denoising makes action synthesis computationally expensive and slow, limiting their effectiveness in fast-reacting policies. RecFlow Policy replaces the diffusion process with a novel rectified flow parameterization, significantly enhancing both computational speed and policy accuracy. RecFlow Policy learns a deterministic coupling to achieve rapid policy inference. This deterministic nature allows for precise visuomotor control with minimal inference time, making it highly suitable for real-time robotic applications. Unlike conventional iterative training methods, our approach selectively refines the rectification process using expert demonstrations to reduce accumulated errors. Leveraging nearly straight flows, RecFlow Policy achieves high accuracy with just a single denoising step. To evaluate the effectiveness of RecFlow Policy, we conducted extensive experiments across both simulated and real-world tasks. Results show that our method matches or surpasses the performance of state-of-the-art diffusion-based methods while while offering greater simplicity and computational efficiency. Compared to Diffusion Policy, which involves numerous iterative steps and incurs significant computational overhead, our approach offers a streamlined and scalable solution for real-time visuomotor policy learning. Code is available on **RecFlow Policy code**.

1 INTRODUCTION

The development of efficient and scalable visuomotor policies has been a central focus in robotics, particularly for tasks involving imitation learning where robots must replicate complex human skills. Generative models, such as diffusion models Chi et al. (2023a), have demonstrated state-of-the-art performance in modeling these tasks. However, their reliance on iterative denoising processes for action synthesis results in slow inference, restricting their practical applicability in real-time robotic environments. This limitation is especially problematic for dynamic tasks requiring rapid, reactive control.

To overcome the limitations, recent work has explored various techniques to accelerate the inference process of diffusion-based policies, including reducing the number of denoising steps Song et al. (2021a), parallelizing the denoising process Shih et al., and distilling the diffusion model into a faster student model Prasad et al. (2024). However, these methods often come with trade-offs, such as reduced sample quality, increased memory requirements, or the need for extensive hyperparameter tuning. Despite these efforts, achieving real-time performance while maintaining high accuracy remains a significant challenge, particularly in resource-constrained robotic systems.

In this paper, we introduce RecFlow Policy, a novel approach that addresses these limitations by replacing the iterative denoising process with a deterministic rectified flow Liu et al. (2023b). Our method directly learns a *coupling* between Gaussian noise and clean action distributions, which allows for efficient, one-step action synthesis while maintaining high accuracy. By eliminating the need for multiple denoising steps, RecFlow Policy dramatically accelerates inference, achieving a **98.7%** reduction in latency compared to traditional diffusion models without compromising performance.

RecFlow Policy leverages the principles of rectified flow, where actions are mapped from noise to precise target through a straightforward deterministic process. RecFlow Policy also inherits the merits of flow matching, i.e., the ability to encode high-dimensional multimodal distributions. This approach not only boosts computational efficiency but also ensures that high-precision actions are generated rapidly, making it highly suitable for real-time robotic control in dynamic environments. Furthermore, by selectively refining the flow during training, RecFlow Policy mitigates the potential errors introduced by early-stage approximations, ensuring robustness even with fewer training samples.

We validate the effectiveness of RecFlow Policy through extensive experiments across a variety of simulations and real-world tasks. In the simulation, RecFlow Policy achieves significant improvements over baseline diffusion models, with an average success rate increase of **60.3%** across 66 tasks. Real-world evaluations further demonstrate its advantages, particularly in tasks involving high precision and long-horizon manipulation, where RecFlow Policy outperforms traditional methods by a large margin. Notably, in complex real-world robotic manipulation tasks like flower arrangement and building a champagne tower, which demand continuous fine-grained control, our approach excels in managing dynamic objects and executing highly accurate sequential manipulations.

Our contributions are: (i) We present RecFlow Policy, a method that combines the efficiency of deterministic flow with the power of generative modeling to enable fast and accurate visuomotor control; (ii) We demonstrate the superiority of our approach in a broad range of robotic manipulation tasks, showcasing its ability to handle both simple and complex scenarios with minimal computational overhead; (iii) We highlight the potential for RecFlow Policy to enable real-time robotic applications, where fast decision-making and robust performance are critical.

2 RELATED WORK

2.1 DIFFUSION MODELS IN ROBOTICS

Diffusion models have demonstrated their ability to express complex multimodal distributions, exhibiting stable training dynamics and robustness to hyperparameter variations. This has led to their widespread application in various robotic tasks, including motion planning Janner et al. (2022); Luo et al. (2024); Carvalho et al. (2023); Saha et al. (2024); Huang et al. (2023), imitation learning Pearce et al. (2023); Chi et al.; Ha et al. (2023); Xian et al. (2023); Li et al.; Ze et al. (2024); Li et al.;

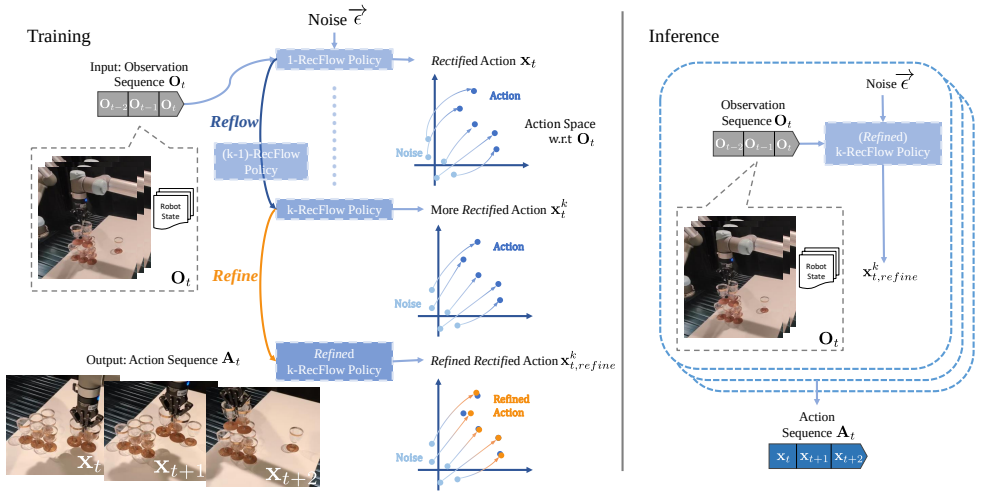


Figure 2: **Overview of our method.** We train a visuomotor policy in an iterative manner to transport straight between noise distribution and target action space, hence enabling lightning one-step sampling during inference. The rectified action flow is selectively *refine*d, lowering the potential accumulated error brought by multiple *reflow*s.

Wang et al. (2024); Chen et al. (2024); Sridhar et al. (2023); Zhao et al. (2024); Chi et al. (2024), goal-conditioned imitation learning Reuss et al.; 2024); Chen et al. (2023a); Zhang et al. (2023), and grasp prediction Urain et al. (2023). Most of these works focus on sequential trajectory generation by denoising over the full horizon. For instance, Diffuser Janner et al. (2022) produces a sequence spanning the entire episode while Diffusion Policy Chi et al. samples plans over a shorter action horizon. Chen et al. (2024) leverages the Stochastic Interpolants to integrate source distributions into diffusion-style imitation learning, but it relies on a sufficiently informative source policy and its precision decreases when the number of diffusion steps is reduced to a single digit. Although these methods have shown impressive results in modeling complex distributions, their reliance on iterative denoising steps makes them impractical for real-time robotic applications. Our proposed RecFlow Policy addresses this limitation by replacing the diffusion process with a rectified action flow, enabling a deterministic and fast action synthesis without sacrificing accuracy.

Recently, flow matching Lipman et al. (2022), a variant of diffusion, has shown its potential to represent complex continuous action distributions. AdaFlow Hu et al. (2024) devises a variance-adaptive ODE solver that can adjust its step size in the inference stage. However, AdaFlow only reduces to a one-step generator when the action distribution is uni-modal, partly due to the lack of *reflow*. Also, the method is not tested on real robots or more comprehensive simulation domains. Braun et al. (2024) leverages the Riemannian extension of flow matching models, but it only conducts two proof-of-concept experiments on LASA handwriting dataset. π_0 Black et al. (2024) utilizes a pre-trained Vision-Language Model to produce actions via flow matching and able to handle high-frequency action chunks. Although it is trained via the flow matching loss, π_0 still requires 10 integration steps since *reflow* or *refine* is not applied. In contrast, our method shows high precision even with one-step prediction.

Notably, we not only successfully adapt rectified ODE, namely *reflow*, for robot motion learning, but also mitigate the possible accumulated error caused by the multiple *reflow*s with our proposed *refine*ment. Also, RecFlow Policy is verified on comprehensive simulation and real-world experiments.

2.2 Accelerating Diffusion Models for Robotics

Efforts to speed up diffusion models have been explored extensively in both image generation Karras et al.; Song et al. (b;a); Kim et al. (2024) and robotics. For example, Reuss et al. reduced denoising steps to 3 for goal-conditioned action generation, while Consistency Policy Prasad et al. (2024) adapted Consistency Trajectory Models Kim et al. (2024) to achieve faster inference with minimal performance loss. However, these methods often require complex distillation processes or introduce

constraints, such as overly smooth trajectories in Dynamical Motion Primitives (DMPs) Scheikl et al. (2024). Streaming Diffusion Policy (SDP) Høeg et al. (2024) and related approaches like Rolling Diffusion Ruhe et al. (2024) and Temporally Entangled Diffusion Zhang et al. (2024) improve speed through parallelization or buffering, but they often incur significant memory overhead or require intricate implementation.

In contrast, RecFlow Policy simplifies the process by leveraging rectified action flow, which eliminates the need for iterative denoising. By learning a deterministic coupling between noisy and clean actions, RecFlow Policy achieves fast and memory-efficient inference without relying on complex buffering or distillation procedures, making it highly suitable for real-time robotic control.

2.3 Distillation and Consistency Models

Distillation-based techniques have been explored to accelerate diffusion model inference speeds in the text-to-image domain Song et al. (b). Many of these distillation techniques start with a pre-trained teacher model and train a new student model to take larger steps over the ODE trajectories that the teacher has already learned Kim et al. (2024); Prasad et al. (2024). By taking these larger steps, the student model can complete a generation in a smaller total number of steps. Consistency models, in particular, support both single and multi-step sampling of outputs. Consistency distillation techniques exploit the self-consistency property of ODE trajectories by training the student model to predict the same output when given two distinct points along the same ODE trajectory Song et al. (b). This objective was first introduced by Song et al. (b), who chose a pair of adjacent input points and taught the student model to map those input points to the same starting point on the given ODE trajectory. Kim et al. (2024) generalized this method by training for arbitrary step sizes and arbitrarily spaced input points, achieving state-of-the-art results in the image-generation domain.

While distillation and consistency models have shown promise in reducing inference time, they often require extensive training and careful tuning of hyperparameters. RecFlow Policy avoids these complexities by directly learning a deterministic coupling between noisy and clean actions, enabling single-step generation without the need for iterative distillation or consistency training.

2.4 Non-Diffusion Based Alternatives

There has been a long line of work using non-diffusion-based model architectures for visuomotor robotics policies. Such alternatives often perform worse than diffusion policies on the same tasks or require external computational resources that may be unavailable in many robotics settings. For example, Zhao et al. offers policy learning via a Conditional VAE instead of diffusion model, and Behavioral Transformers Shafiullah et al. (2022) represent a key alternative to Diffusion Policies, but they often struggle to match the performance of diffusion-based methods, especially in complex, multi-modal tasks. While non-diffusion methods offer simplicity, they often fail to capture the complexity of real-world tasks. RecFlow Policy bridges this gap by combining the simplicity of deterministic models with the expressive power of diffusion-based approaches, enabling fast and accurate visuomotor control without the need for external computational resources.

Overall, while significant progress has been made in accelerating diffusion models for robotics, achieving real-time performance without sacrificing accuracy remains a challenge. Our work builds on these advancements by introducing RecFlow Policy, which leverages rectified flow to achieve fast and accurate visuomotor control. By replacing the iterative denoising process with a deterministic coupling, RecFlow Policy offers a streamlined and efficient solution for real-time robotic applications.

3 Method

A visuomotor policy solves the task of observing a sequence of visual observations and predicts the next action to execute in the environment. We formulate our visuomotor policy as a Rectified Flow Model, denoted as *RecFlow Policy*. As formulated in the method of probability flows Song et al. (2021b), the target at timestep 0 is iteratively denoised from a random noise at timestep T. In robot learning scenarios, an action space with respect to the corresponding task observation consists

of all possible actions of the robot. The condition in visuomotor policies often includes image sequences, i.e. what the robot has visually observed until now. In this paper, we use a CNN network to encode an image sequence as conditioning. We model specific conditional action spaces $\mathcal{A}$ with its corresponding conditioning, and construct the *couplings* $(\mathbf{a}_T, \mathbf{a}_0)$ drawn from noise distribution $\mathbf{a}_T \in \mathcal{N}(0, \mathbf{I})$ and action distribution $\mathbf{a}_0 \in \mathcal{A}$, facilitating one-step inference with even superior accuracy.

In this section, we first describe our modeling of action space in Subsection 3.1. Then, we introduce the training process and inference procedure of RecFlow Policy in Subsection 3.2 and Subsection 3.3, respectively. The training details are covered in Subsection 3.4.

3.1 Rectified Action Flow

We first introduce our method by describing how we model the action space. The objective of modeling is to construct a simple yet unique mapping between two empirical observations derived from their respective distributions. We interpret the mapping in the form of ordinary differentiable model (ODE) on time $t \in [0, T]$, as per the rectified flow model Liu et al. (2023a). In a vanilla diffusion-based policy, we have two actions $(\mathbf{a}_T, \mathbf{a}_0)$. The former is a noisy robot motion $\mathbf{a}_T \in \mathbb{R}^d$ sampled from the unit Gaussian $\mathcal{N}(0, \mathbf{I})$, and the latter is the target clean action $\mathbf{a}_0 \in \mathbb{R}^d$ derived from the expert action distribution conditioned on the current observations, for example, RGB images. We denote the conditional action distribution as $\mathcal{A}$. Given a specific action and a random noisy action, we want to find the *coupling*, i.e., the transport plan, of their distributions $\mathcal{N}(0, \mathbf{I})$ and $\mathcal{A}$. Note that $\mathbf{a}_t$ with $t \in [0, T]$ represents a vanilla noisy robot action, without specifying whether it belongs to a *coupling*.

Before the policy is well-trained, the drift force $v : \mathbb{R}^d \to \mathbb{R}^d$ is set to drive the flow to follow the direction $\big(\mathbf{a}_0 - \mathbf{a}_T\big)$ of the linear path pointing from $\mathbf{a}_T$ to $\mathbf{a}_0$ as straight as possible, by solving a simple least squares regression problem:

$$\min_v \int_0^1 \mathbb{E}\left[||(\mathbf{a}_0 - \mathbf{a}_T) - v\big(\mathbf{a}_t, t\big)||^2\right] \mathrm{d}t, \tag{1}$$

where $\mathbf{a}_t$ is the linear interpolation of $\mathbf{a}_T$ and $\mathbf{a}_0$, i.e.,

$$\mathbf{a}_t = \frac{t}{T}\mathbf{a}_T + \frac{T-t}{T}\mathbf{a}_0, t \in [0, T]. \tag{2}$$

Naturally, $\mathbf{a}_t$ follows the ODE of $\mathrm{d}\mathbf{a}_t = (\mathbf{a}_0 - \mathbf{a}_T)\mathrm{d}t$, where any update of $\mathbf{a}_t$ requires the information of the target clean action $\mathbf{a}_0$. By fitting the drift v with $\mathbf{a}_0 - \mathbf{a}_T$, the rectified action flow causalizes the paths of linear interpolation $\mathbf{a}_t$, relieving the burden of involving the target action (which is unknown during inference) when simulating the ODE flow.

Given the Cauchy-Lipschitz Theorem, the solution of a well-defined ODE should be unique. This gives the non-crossing property of the flows, i.e., paths following $\mathrm{d}\mathbf{a}_t = v(\mathbf{a}_t, t)\mathrm{d}t$ is unique and will never intersect each other at any time $t \in (0, T]$. Otherwise, at the intersection $\hat{\mathbf{a}}$, the flow can go towards different directions, making the solution non-unique. However, the paths of the interpolation $\mathbf{a}_t$ may cross each other. Thanks to the Equation (1), the interleaved trajectories are rewired after optimizing it.

Now the solution of the Equation (1) is our policy network. Given a noise $\mathbf{a}_T \sim \mathcal{N}(0, \mathbf{I})$, our policy gives the corresponding *reflow*ed action $\mathbf{a}_0$. This rectified *coupling* $(\mathbf{a}_T, \mathbf{a}_0)$ guarantees that the transport cost is not higher than any random (action$\times$noise) pair simultaneously for all convex cost functions, which can easily be proved via Jensen's inequality. In this way, we have found a deterministic mapping between the action space $\mathcal{A}$ and a unit Gaussian $\mathcal{N}(0, \mathbf{I})$.

In practice, the drift v is parameterized with our neural policy network and we can solve Equation (1) with any stochastic optimizer. The trained 1-RecFlow Policy is denoted as v^1. To make a distinction, our 1-RecFlow Policy is trained with the random noise and the groundtruth action pairs $(\mathbf{a}_T^0, \mathbf{a}_0^0)$, and the derived *coupling* is denoted as $(\mathbf{a}_T^1, \mathbf{a}_0^1)$. Eventually, the desired 1-RecFlow Policy action flow induced between their distributions ($\mathcal{N}(0, \mathbf{I})$ and $\mathcal{A}$) is

$$\mathrm{d}\mathbf{a}_t^1 = v^1(\mathbf{a}_t^1, t)\mathrm{d}t, \quad t \in [0, T], \tag{3}$$

which converts the noise $\mathbf{a}_T^1 \in \mathcal{N}(0, \mathbf{I})$ in the *coupling* $(\mathbf{a}_T^1, \mathbf{a}_0^1)$ to the action $\mathbf{a}_0^1$ which follows the conditioned expert action distribution. Algorithm 1 shows the training streamline.

After the drift v is estimated, we solve the ODE forwardly starting from $\mathbf{a}_T \sim \mathcal{N}(0, \mathbf{I})$ to transfer $\mathcal{N}(0, \mathbf{I})$ to $\mathcal{A}$, or backwardly starting from $\mathbf{a}_0 \sim \mathcal{A}$ to transfer $\mathcal{A}$ to $\mathcal{N}(0, \mathbf{I})$. By doing this, we obtain the *coupling* of these distributions. In our settings, we first sample noise $\mathbf{a}_T^1 \sim \mathcal{N}(0, \mathbf{I})$ and then generate $\mathbf{a}_0^1$ forwardly following $\mathrm{d}\mathbf{a}_t = v^1(\mathbf{a}_t^1, t)\mathrm{d}t$ starting from noise $\mathbf{a}_T^1$. Here v^1 is the 1-RecFlow Policy trained with data $(\mathbf{a}_T^0, \mathbf{a}_0^0)$, and $t \in [0, T]$ is the sampling timestep. The *coupling* generation steps are outlined in Algorithm 2.

We can train our policy recursively using the rectified *coupling*s as the substitution of the former pairs. The k-th policy network yielded by the k-th iteration is denoted as k-RecFlow Policy, where $k \in \mathbb{N}^*$. To be specific, the k-RecFlow Policy is trained using the *coupling* $(\mathbf{a}_T^{k-1}, \mathbf{a}_0^{k-1})$ are generated via (k-1)-RecFlow Policy. The k-RecFlow Policy action flow has a velocity v^k that satisfies

$$\mathrm{d}\mathbf{a}_t^k = v^k(\mathbf{a}_t^k, t)\mathrm{d}t, \quad t \in [0, T], \tag{4}$$

where the noise $\mathbf{a}_T^k \in \mathcal{N}(0, \mathbf{I})$ and the the action $\mathbf{a}_0^k$ in the *coupling* $(\mathbf{a}_T^k, \mathbf{a}_0^k)$ are generated via k-RecFlow Policy and can be used to train (k+1)-RecFlow Policy. Algorithm 2 displays the detailed *reflow* pipeline. To simplify, the RecFlow Policy in this paper refers to the 2-RecFlow Policy. This procedure increasingly straightens the paths of the flows. The straighter the paths are, the smaller the time-discretization error in numerical simulation will be. Perfectly straight paths can be exactly simulated with a single Euler step. This addresses the very bottleneck of high inference cost in existing continuous-time ODE-based models, such as the Diffusion Policy Chi et al. (2023a) built upon Probability Flow ODE Song et al. (2021b).

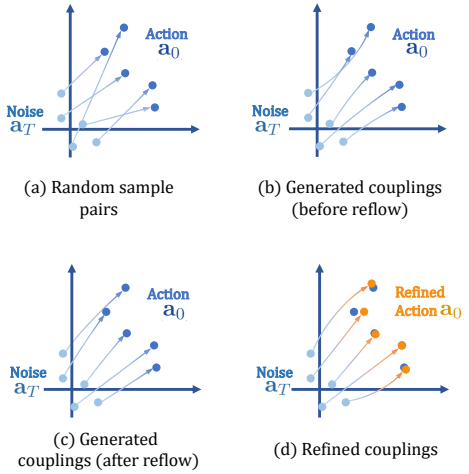


Figure 3: **Sampling trajectories of RecFlow Policy at different stages.** Randomly sampled pairs in (a) have crossing flows. *Coupling*s in (b) have been rewired so they do not intersect with each other at the same denoising timestep. The trajectories in (c) and (d) are nearly straight. Therefore, we can apply one-step sampling without compromising performance.

3.2 Training

To train a model capable of single-step generation, we begin by training a 1-RecFlow Policy model.

3.2.1 1-RecFlow Policy model

Our 1-RecFlow Policy model takes as input the current noisy action $\mathbf{a}_t$ at timestep $t \in (0, T]$ along the ODE, as well as the condition $\mathbf{O}$. We used the current-observed images as the visual condition, and agent position as the low-dimensional condition. Both modalities go through an observation encoder to obtain the condition $\mathbf{O}$.

We denote our policy network as $v : \mathbb{R}^d \to \mathbb{R}^d$ with parameter θ, where d is the dimension of the action. Firstly, a timestep t is randomly sampled from Uniform($[0, T]$). $\mathbf{a}_0^0$ is the groundtruth action.

Algorithm 1: 1-RecFlow Policy

Input: Noise and target action pairs $(\mathbf{a}_T^0, \mathbf{a}_0^0)$, sampled from $\mathcal{N}(0, \mathbf{I})$ and $\mathcal{A}$ respectively. The corresponding visual observation sequence $\mathbf{O} = ([\text{Image}],\ [\text{robot state}])$. Initial parameters of the policy $v^0 \colon \mathbb{R}^d \to \mathbb{R}^d$.

Procedure:

while *terminal condition* **do**
- Sample timestep $t \sim \text{Uniform}([0, 1])$.
- Compute $\mathbf{a}_t^0 = t\mathbf{a}_T^0 + (T - t)\mathbf{a}_0^0$.
- Evaluate $\mathbb{E}\big[||\mathbf{a}_0^0 - \mathbf{a}_T^0 - v_\theta(\mathbf{a}_t^0,\ t)||^2\big]$.
- Update parameters parameters of v^0.

end

Output: The trained 1-RecFlow Policy with velocity estimation v^1.

Algorithm 2: $\star k$-Reflow

Input: k-RecFlow Policy with velocity estimation v^k. Number of *couplings* N.

Procedure:

`// Coupling generation`

for $i = 1$ *to* N **do**
- Sample noise $\mathbf{a}_T^k \sim \mathcal{N}(0, \mathbf{I})$.
- Generate action $\mathbf{a}_0^k$ following $\mathrm{d}\mathbf{a}_t = v^k(\mathbf{a}_t^k, t)\mathrm{d}t$ starting from noise $\mathbf{a}_T^k$.
- Construct *coupling* $(\mathbf{a}_T^k, \mathbf{a}_0^k)$.

end

`// Training`

Initialize parameters of v^{k+1} as the same as v^k.

while *terminal condition* **do**
- Sample timestep $t \sim \text{Uniform}([0, 1])$.
- Compute $\mathbf{a}_t^k = t\mathbf{a}_T^k + (T - t)\mathbf{a}_0^k$.
- Evaluate $\mathbb{E}\big[||\mathbf{a}_0^k - \mathbf{a}_T^k - v^{k+1}(\mathbf{a}_t^k,\ t)||^2\big]$.
- Update parameters of v^{k+1}.

end

Output: $(k + 1)$-RecFlow Policy. *Coupling* $(\mathbf{a}_T^k, \mathbf{a}_0^k)$.

$\star$*Reflow* is optional and can be done multiple times.

The current noisy action is computed as

$$\mathbf{a}_t^0 = \frac{t}{T}\mathbf{a}_T^0 + \frac{T - t}{T}\mathbf{a}_0^0. \tag{5}$$

RecFlow Policy's output is trained to fit the vector $\mathbf{a}_0^0 - \mathbf{a}_T^0$, i.e., the difference between target action and the initial noise. Then we estimate the loss

$$\mathcal{L}(\theta) = \mathbb{E}\left[||(\mathbf{a}_0^0 - \mathbf{a}_T^0) - v\big(t\mathbf{a}_T^0 + (T - t)\mathbf{a}_0^0, \mathbf{O}, t\big)||^2\right]. \tag{6}$$

After minimizing Equation 6, we obtain our 1-RecFlow Policy model. Algorithm 1 demonstrates the training routine.

3.2.2 K-RecFlow Policy model

To straighten the paths of flow to a larger extent, we can further repeat almost the same procedure in Subsubsection 3.2.1. The difference is that we use the generated *couplings* of the (k-1)-RecFlow Policy.

Assume that k-RecFlow Policy is already trained. We randomly sample multiple noise $[\mathbf{a}_T^k]_n$ from $\mathcal{N}(0, \mathbf{I})$. Then pass them through k-RecFlow Policy to get the denoised actions $[\mathbf{a}_0^k]_n$. Note that we still use the same conditioning as in the last training round during this procedure. The *coupling*

Algorithm 3: †Refine after k-reflow

Input: Target action $\mathbf{a}_0^k$ and corresponding visual observation sequence $\mathbf{O}$. Action distance threshold δ.

Procedure:

Find the nearest condition in ground-truth conditions to $\mathbf{O}$. Its corresponding action is $\mathbf{a}_0^{k,*}$.

if $d(\mathbf{a}_0^k, \mathbf{a}_0^{k,*}) < \delta$ **then**

  $\mathbf{a}_0^{k,\textit{refine}} \doteq \mathbf{a}_0^{k,*}$

else

  $\mathbf{a}_0^{k,\textit{refine}} \doteq \mathbf{a}_0^k$

end

Output: Refined target action $\mathbf{a}_0^{k,\textit{refine}}$.

†*Refine* can be selectively done before the next (k+1)-*reflow*.

pairs $\left(\mathbf{a}_T^k, \mathbf{a}_0^k\right)_n$ are then used as training data of the (k+1)-RecFlow Policy. This process is named as *reflow*. Algorithm 2 demonstrates the routine.

3.2.3 REFINEMENT

It is noteworthy that in the visuomotor settings, one of the conditioning is the RGB images, which are rendered in expert demonstrations in advance of training 1-RecFlow Policy. However, after *reflow*, the generated actions $[\mathbf{a}_0^1]_n$ are no longer the same as groundtruth actions. Therefore, the corresponding images are no longer aligned with them.

To mitigate the performance degradation brought by *reflow*, we design an approach to *refine* the generated actions $[\mathbf{a}_0^1]_n$. Per generated action $\mathbf{a}_0$, we apply a traversal among groundtruth actions to find its nearest twin. If the twins' distance is below the threshold, we replace the generated action with the groundtruth one. We emphasize that action spaces are different in terms of conditioning, so this approach is equivalent to freezing conditioning and finding the *coupling*. There are three cases: Having found its nearest twin means that the prediction error results from the *reflow*. Replacing it will not change the least transport costs property. Or, the *reflow*ed model predicts another possible solution rather than that generated by the expert policy. It reflects the policy's multimodality and we do not interfere with it. Otherwise, it predicts an action that does not help complete the task. However, this mistake happens irregularly, so it is likely that the error caused by a few steps will be offset afterward.

The pipeline is shown in Algorithm 3.

3.3 INFERENCE

3.3.1 RECFLOW POLICY SAMPLING

During sampling, we compute $\mathrm{d}\mathbf{a}_t = v(\mathbf{a}_t, t)\mathrm{d}t$. Starting from $\mathbf{a}_T$, we iteratively solve for $\mathbf{a}_0$, i.e.,

$$\mathbf{a}_0 = \mathbf{a}_T + \sum_{t=1}^{T} \mathrm{d}\mathbf{a}_t. \tag{7}$$

3.3.2 ONE-STEP SAMPLING

Using a well-trained k-RecFlow Policy, a one-step prediction is sufficient to approximate the target *coupling* action using $\mathbf{a}_0^k = \mathbf{a}_T^k + v\left(\mathbf{a}_T^k, t\right)$. There exists a trade-off between training costs and performance. If the k-RecFlow Policy has heavily degraded performance with one-step sampling, it means the flow is not straight enough. In this case, it is wise to repeat the reflow process for a few more iterations to further smooth the paths of the ODE. However, it may sacrifice accuracy after too many *reflow*s. If the ancestor model has an accuracy below 100%, which is an extremely common case, the generated couplings will not be 100% correct. In this case, the next *reflow*ed model may learn from a set of actions composed of successful actions and pseudo-ground truth (failed) actions, decreasing its success rate. Applying our *refine* before each *reflow* would lessen its damage.

3.4 IMPLEMENTATION DETAILS

For RecFlow Policy network, we adopt the 1D convolutional U-Net architecture from Diffusion Policy Chi et al. (2023a). This architecture conditions on observations and the diffusion timestep t using FiLM Perez et al. (2018) blocks, and diffuses through the action domain using 1D convolutional blocks. For the observation encoder, we randomly initialize per input modality a ResNet-18 followed by a spatial softmax pooling and a ReLU activation. The BatchNorm is replaced with GroupNorm for stable training.

For the training methodology, we make a few changes to our baseline from Chi et al. (2023a). We change the training objective from sample or epsilon prediction in Diffusion Policy to difference between noise and target action prediction. To be specific, the policy predicts the drift $v = \mathbf{a}_0 - \mathbf{a}_T$, where $\mathbf{a}_0 \in \mathcal{A}$ is the clean action and $\mathbf{a}_T \in \mathcal{N}(0, \mathbf{I})$ is the noise. Since we only make minor changes to the training of vanilla diffusion-based policies, it does not involve extra costs and is easy to implement on any diffusion-based policy by modifying a few lines of code.

We adopt a CNN-based backbone, since a Transformer-based backbone requires more hyperparameter tuning, as described in Diffusion PolicyChi et al. (2023a). However, as Chi et al. (2023a) recommend, one may replace CNN with Transformer if the task is complex or action changes at a high rate. Nevertheless, this substitution would have similar effects on RecFlow Policy and our baselines. Hence, choosing the CNN-based backbone does not weaken or nullify our fair comparison.

When performing the *reflow*, we freeze the observation encoder for two reasons. First, the ground truth actions and their corresponding images are fed into the model in former training. Pairing images with changed actions would hurt the observation encoder's performance. Next, it would accelerate the training process and thus mitigate the training costs of *reflow*.

Other dataset-related details are described in the Subsection 4.2.

4 SIMULATION EXPERIMENTS

4.1 ALGORITHM

We follow the procedure in Algorithm 1, 2, and 3. Starting with drawing noise and action pair $(\mathbf{a}_T, \mathbf{a}_0)$ from $\mathcal{N}(0, \mathbf{I}) \times \mathcal{A}$, the 1-RecFlow Policy is trained by minimizing Equation 1. K-RecFlow Policy is obtained by repeating *reflow* $k-1$ times. The only difference between training the first RecFlow Policy and its $k-1$ successors is that the input pair of *reflow* is the generated *coupling* $(\mathbf{a}_T^{\cdot}, \mathbf{a}_0^{\cdot})$.

After obtaining a k-RecFlow Policy, distilling the relation of $(\mathbf{a}_T^k, \mathbf{a}_0^k)$ into a student policy would help to directly predict the target action without simulating the flow. However, we empirically found that distillation is not necessary in most cases. To be specific, the target is well approximated by the 1-step update via our k-RecFlow Policy so we save the effort of distillation. Nevertheless, the distillation can be done efficiently given that the flow is already nearly straight. Please refer to Figure 3 for the sampling trajectories visualization of our policies. We should underscore that *reflow* is not a kind of distillation, for the process only aims at finding the *coupling*s with lower transport costs which would further facilitate 1-step inference.

In the sampling phase, it is also feasible to process backwardly, i.e., start from $\mathbf{a}_0 \sim \mathcal{A}$ and follow $\mathrm{d}\mathbf{a}_0 = -v(\mathbf{a}_0, t)\mathrm{d}t$ to derive its coupling $\mathbf{a}_T$. Intuitively, starting from groundtruth action $\mathbf{a}_0$ would circumvent the error brought by generating pseudo-groundtruth actions. However, the generated noise $\mathbf{a}_T$ would not completely conform to a Gaussian distribution as observational conditioning is introduced, and thus still cause degradation.

4.2 EXPERIMENT SETUP

We systematically evaluate RecFlow Policy on 66 tasks from 5 benchmarks in simulation. We found RecFlow Policy consistently outperforms the current state-of-the-art algorithms on all of the

simulation benchmarks, with an average success rate improvement of **60.0**%. In this section, we provide an overview of our simulation domains, our evaluation methodology on the tasks, and our core findings.

We use DDIM Song et al. (2021a) as the noise scheduler and predict the vector $\mathbf{a}_0 - \mathbf{a}_T$ instead of epsilon or sample prediction, with 100 timesteps during training. We train 1000 epochs for Meta-World and Adroit tasks given their simplicity and 3050 epochs for RoboMimic tasks and Push-T. For Franka Kitchen, we train 5000 epochs due to its long-horizon and multi-task complexity. Real-world tasks are trained with 1000 epochs. The optimizer used is AdamW with the same hyperparameters as that used in Chi et al. (2023a). Batch size is 64 for RecFlow Policy and all the baselines except Franka Kitchen where the batch size is 256.

Before the *reflow* phase, we sample 10 *couplings* in each action space, saving both generation time and *reflow* training time. More importantly, the relatively small group of samples is empirically sufficient to learn from.

4.2.1 Simulation Benchmark

Though the simulation environments are increasingly realistic nowadays (Makoviychuk et al., 2021; Xiang et al., 2020; Todorov et al., 2012; Zhu et al., 2020), a notable gap between simulation and real-world scenarios persists (Ze et al., 2023; Lei et al., 2023; Chen et al., 2023b). This discrepancy underscores two key aspects: *(a)* the importance of real robot experiments and *(b)* the necessity of large-scale diverse simulation tasks for more scientific benchmarking. Therefore, for simulation experiments, we collect in total **66** tasks from **5** domains, covering diverse robotic skills. These tasks range from challenging scenarios like bi-manual manipulation (Mandlekar et al., 2021), and articulated object manipulation (Gupta et al., 2019; Rajeswaran et al., 2017; Yu et al., 2020), to simpler tasks like parallel gripper manipulation (Yu et al., 2020). Properties for each task are summarized in Table 1. Our experimental setup uses the MuJoCo (Todorov et al., 2012) physics simulator. The stable contact dynamics of MuJoCo makes it well suited for contact-rich manipulation tasks, especially with dexterous hands. The kinematics, the dynamics, and the sensing details of the physical hardware are carefully modeled to encourage physical realism, ensuring the rendered visual contexts are reasonable.

Adroit (Rajeswaran et al., 2017) domain involves controlling a 24-DoF dexterous hand manipulator, designed to address challenges in dynamic and dexterous manipulation. There are 4 tasks in this dataset, including object relocation, in-hand manipulation, manipulating environmental props, and tool use. A task example is shown in Fig 4. Trajectories for this domain are collected with agents trained by VRL3 Wang et al. (2022).

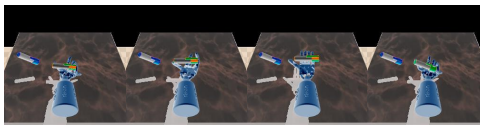


Figure 4: **A rollout of pen task from Adroit.** This task requires dexterous hand manipulation to spin the pen.

Robomimic (Mandlekar et al., 2021) is a large-scale robotic manipulation benchmark designed to study imitation learning and offline reinforcement learning. The benchmark consists of 5 tasks with a proficient human teleoperated demonstration dataset for each and mixed proficient/non-proficient human demonstration datasets for 4 of the tasks (9 variants in total). A task example is shown in Fig 5. We report results on proficient human tele-operated demonstrations for image-based observations. In this domain, we use position control, following Chi et al. (2023b)

Push-T (Florence et al., 2022) requires pushing a T-shaped block (gray) to a fixed target (red) with a circular end-effector (blue). Variation is added by random initial conditions for the T block and end-effector. The task requires exploiting complex and contact-rich object dynamics to push the T block precisely, using point contacts. There are two variants: one with RGB image observations and another with 9 2D keypoints obtained from the ground truth pose of the T block, both with proprioception for end-effector location.

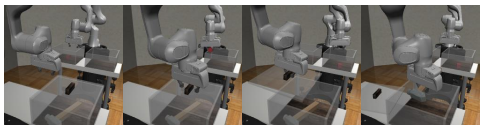


Figure 5: **A rollout of transport task from RoboMimic.** This task involves two agents interacting with three objects, including a lid, a hammer, and a cube.

Meta-World (Yu et al., 2020) is an open-source simulated benchmark for meta-reinforcement learning and multi-task learning consisting of 50 distinct robotic manipulation tasks. A task example is shown in Fig 6. The task distributions are sufficiently broad to test the generalization of the algorithms. We use the script policies in Meta-World to generate expert demonstrations. All the algorithms use the same demonstrations to ensure a fair comparison. Following Seo et al. (2023), tasks in Meta-World are categorized into different difficulty levels *easy*, *medium*, *hard*, and *very hard*. The tasks in each category are shown in Table 2.

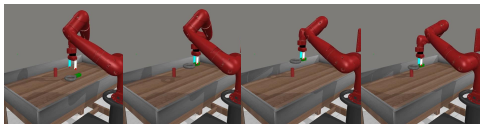


Figure 6: **A rollout of assembly task from Meta-World.** This task requires delicate picking and placing the circle around the stick.

Franka Kitchen (Gupta et al., 2019) is a popular environment for evaluating the ability of imitation learning and offline reinforcement learning methods to learn multiple long-horizon tasks. A task example is shown in Fig 7. The Franka Kitchen environment contains 7 objects for interaction and comes with a human demonstration dataset of 566 demonstrations, each completing 4 tasks in arbitrary order. The goal is to execute as many demonstrated tasks as possible, regardless of order, showcasing both short-horizon and long-horizon multimodality.

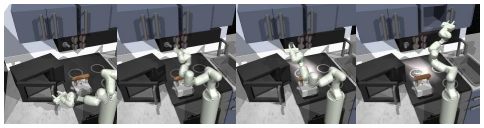


Figure 7: **A rollout of Franka Kitchen tasks.** This long-horizon task consists of several subtasks, including interacting with 7 potential objects.

4.2.2 Evaluation Methodology

Models are evaluated among 10 rollouts in all corresponding tasks except for lift, can, and square in RoboMimic and push-T where they are evaluated among 50 rollouts. Scores of push-T are calculated as the IoU of the T-shaped block and the fixed target, while the others report the average success rates.

Each trained policy is evaluated with 3 different seeds with all the other settings fixed, and then we report the average success rate and the standard deviation among them.

We evaluate the accuracy of RecFlow Policy with a 1-step Euler solver, ensuring a high inference speed. On Adroit, we evaluate our baselines with 10 sampling steps, while on the other domains, we do 100 timesteps if not specified.

4.3 Effectiveness

Comparing RecFlow Policy with SOTA methods in simulation. Our main baseline is the image-based Diffusion Policy Chi et al. (2023b). All settings, such as training epochs, seeds, learning rate schedule, and image resolution for all methods are the same across all experiments, ensuring a fair comparison. Please refer to Table 3 for a summarized report. We observe that RecFlow Policy achieves a success rate exceeding 80% in 31 tasks, whereas Diffusion Policy does in only 16

Table 1: **Task suite of RecFlow Policy in simulation,** including Adroit (Rajeswaran et al., 2017), RoboMimic (Zhu et al., 2020), Franka Kitchen (Gupta et al., 2019), Push-T (Florence et al., 2022), MetaWorld (Yu et al., 2020). ActD: the highest action dimension for the domain. #Demo: Number of expert demonstrations used for each task in the domain. Art: articulated objects. Rigid: rigid objects.

Simulation Benchmark (66 Tasks)						
Domain	End-effector	Object	Simulator	ActD	#Task	#Demo
Adroit	Dexterous hand	Rigid/Art	MuJoCo	28	3	50
RoboMimic	Gripper	Rigid	MuJoCo	7 or 14	5	50
Franka Kitchen	Gripper	Rigid/Art	MuJoCo	9	7	50
Push-T	Circular	Rigid	MuJoCo	2	1	50
MetaWorld	Gripper	Rigid/Art	MuJoCo	4	50	45

tasks. The average success rate of RecFlow Policy reaches **62.3**%, while Diffusion Policy only gets 38.9%. The policies trained on Franka Kitchen are only given low-dimensional conditioning to test our method without visual inputs. RecFlow Policy still achieves all 100% accuracy, surpassing the SOTA policy and showing the robustness of our method with respect to the modality of conditioning.

Comparing RecFlow Policy with more baselines in simulation. We also include Consistency Policy Prasad et al. (2024) as our baseline. Due to the time-consuming procedure of training a teacher model and then distilling it into a student model in Consistency Policy pipeline, we only evaluate its performance on a few randomly selected tasks across various domains. In RoboMimic tasks, we follow the original settings in its paper. For Push-T and Meta-World tasks, we run its pipeline using our settings. The results are reported in Table 4. RecFlow Policy shows consistent improvement on all benchmarks.

Table 2: **Main results on 66 simulation tasks.** Results for each task are provided in this table. A summary across domains is shown in Table 3.

Alg. \ Task	**Adroit** (Rajeswaran et al., 2017) Pen	Door	Hammer	**RoboMimic** (Mandlekar et al., 2021) Lift	Can	Square	Transport	Tool Hang
RecFlow Policy	**52**$\pm$**8**	**37**$\pm$**5**	**35**$\pm$**4**	**100**$\pm$**0**	**100**$\pm$**0**	91$\pm$3	**89**$\pm$**5**	**97**$\pm$**5**
Diffusion Policy	20$\pm$7	23$\pm$6	32$\pm$6	**100**$\pm$**0**	99$\pm$1	**95**$\pm$**3**	89$\pm$7	83$\pm$2

Alg. \ Task	**Kitchen** (Gupta et al., 2019) p1	p2	p3	p4	p5	p6	p7	**Push-T** (Florence et al., 2022)
RecFlow Policy	**100**$\pm$**0**	**100**$\pm$**0**	**100**$\pm$**0**	**100**$\pm$**0**	**7**$\pm$**3**	**0**$\pm$**0**	**0**$\pm$**0**	**80**$\pm$**2**
Diffusion Policy	**100**$\pm$**0**	**100**$\pm$**0**	99$\pm$1	98$\pm$2	3$\pm$2	**0**$\pm$**0**	**0**$\pm$**0**	78$\pm$1

Alg. \ Task	**Meta-World (Yu et al., 2020) (Easy)** Button Press	Button Press Topdown	Button Press Topdown Wall	Button Press Wall	Coffee Button	Dial Turn	Door Close
RecFlow Policy	**87**$\pm$**9**	**87**$\pm$**9**	**93**$\pm$**9**	93$\pm$9	**100**$\pm$**0**	**67**$\pm$**9**	**100**$\pm$**0**
Diffusion Policy	67$\pm$9	67$\pm$9	67$\pm$19	**100**$\pm$**0**	93$\pm$9	27$\pm$9	0$\pm$0

Alg. \ Task	**Meta-World (Easy)** Door Lock	Door Open	Door Unlock	Drawer Close	Drawer Open	Faucet Close	Faucet Open	Handle Press	Handle Pull
RecFlow Policy	**27**$\pm$**9**	**93**$\pm$**9**	**100**$\pm$**0**	**100**$\pm$**0**	**100**$\pm$**0**	87$\pm$9	**93**$\pm$**9**	**73**$\pm$**25**	**13**$\pm$**9**
Diffusion Policy	20$\pm$16	20$\pm$0	93$\pm$9	67$\pm$25	93$\pm$9	**93**$\pm$**9**	53$\pm$9	47$\pm$9	6$\pm$9

Alg. \ Task	**Meta-World (Easy)** Handle Press Side	Handle Pull Side	Lever Pull	Plate Slide	Plate Slide Back	Plate Slide Back Side	Plate Slide Side	Reach
RecFlow Policy	**93**$\pm$**9**	**27**$\pm$**9**	**60**$\pm$**16**	**87**$\pm$**9**	**73**$\pm$**9**	**100**$\pm$**0**	**100**$\pm$**0**	33$\pm$9
Diffusion Policy	20$\pm$16	0$\pm$0	20$\pm$0	80$\pm$0	73$\pm$19	67$\pm$25	**100**$\pm$**0**	**40**$\pm$**0**

Alg. \ Task	**Meta-World (Easy)** Reach Wall	Window Close	Window Open	Peg Unplug Side	**Meta-World (Medium)** Basketball	Bin Picking	Box Close	Coffee Pull	Coffee Push
RecFlow Policy	**67**$\pm$**9**	**100**$\pm$**0**	**87**$\pm$**9**	**47**$\pm$**25**	**0**$\pm$**0**	**53**$\pm$**25**	**27**$\pm$**19**	**80**$\pm$**16**	**100**$\pm$**0**
Diffusion Policy	47$\pm$9	73$\pm$19	33$\pm$25	20$\pm$16	**0**$\pm$**0**	0$\pm$0	13$\pm$9	27$\pm$9	20$\pm$16

Alg. \ Task	**Meta-World (Medium)** Hammer	Peg Insert Side	Push Wall	Soccer	Sweep	Sweep Into	**Meta-World (Hard)** Assembly	Hand Insert	Pick Out of Hole	Pick Place
RecFlow Policy	33$\pm$9	**13**$\pm$**19**	13$\pm$19	**27**$\pm$**9**	**13**$\pm$**9**	20$\pm$16	**33**$\pm$**9**	**40**$\pm$**16**	**47**$\pm$**25**	**33**$\pm$**9**
Diffusion Policy	**67**$\pm$**19**	0$\pm$0	**13**$\pm$**9**	20$\pm$16	27$\pm$9	**27**$\pm$**19**	27$\pm$9	27$\pm$9	0$\pm$0	0$\pm$0

Alg. \ Task	**Meta-World (Hard)** Push	Push Back	**Meta-World (Very Hard)** Shelf Place	Disassemble	Stick Pull	Stick Push	Pick Place Wall	**Average**
RecFlow Policy	**47**$\pm$**9**	**53**$\pm$**19**	**7**$\pm$**9**	**87**$\pm$**9**	**13**$\pm$**19**	**80**$\pm$**16**	**20**$\pm$**0**	**62.2**
Diffusion Policy	13$\pm$9	33$\pm$9	0$\pm$0	60$\pm$16	7$\pm$9	**80**$\pm$**16**	13$\pm$19	38.9

Table 3: **Main simulation results.** Averaged over 66 tasks, RecFlow Policy achieves $\mathbf{60.3}\%$ relative improvement compared to Diffusion Policy. Success rates for individual tasks are in Table 2.

Algorithm \ Task	Adroit (3)	RoboMimic (5)	Kitchen (7)	Push-T (1)	Meta-World Easy (28)	Meta-World Medium (11)	Meta-World Hard (6)	Meta-World Very Hard (5)	Average (66)
RecFlow Policy	**41.3**	**95.4**	**58.1**	**80.0**	**78.1**	**34.5**	**42.2**	**41.4**	**62.3 (↑ 60.3%)**
Diffusion Policy	25.0	93.2	57.1	78.0	38.3	19.5	16.7	32.0	38.9

Table 4: **Comparing RecFlow Policy with more baselines in simulation.** We include Diffusion Policy, 1-step DDIM, and Consistency Policy.

Algorithm \ Task	**Adroit** Pen	**Push-T**	**RoboMimic** Square	**Average**
RecFlow Policy	**52**$\pm$**8**	**80**$\pm$**2**	**97**$\pm$**5**	**76.3**
Diffusion Policy (100-step)	20$\pm$7	78$\pm$1	83$\pm$2	60.3
Diffusion Policy (1-step)	0$\pm$0	70$\pm$0	0$\pm$0	23.3
Consistency Policy (1-step)	32$\pm$8	71$\pm$2	89$\pm$2	64.0

4.4 Efficiency

We use assembly task in Meta-World to measure inference latency. Per action prediction step, RecFlow Policy only denoises once while Diffusion Policy does 100 times, according to their original settings. Since other functions, such as observation encoding, consume little time compared with sampling, RecFlow Policy achieves significant speedup.

After warming up for 200 iterations, we evaluate the average latency of one prediction over 800 rollouts. Then we exclude the top 25% and bottom 25% durations to compute the average time of 400 prediction steps. Table 5 showcases wall clock times for each of the algorithms in one simulation task, specifically over Meta-World assembly, without loss of generality. The settings are the same as described in the main experiments. RecFlow Policy achieves **98.7**% acceleration compared with Diffusion Policy, while reaching a higher average accuracy among all domains.

Consistency Policy uses 1-step sampling and obtains inference speed comparable to ours. However, we also observe much larger time consumption when distilling its teacher model into a student model, partly due to the Huber loss which involves more operands. Hence, its training takes more effort than RecFlow Policy. Additionally, while showing comparable inference speed, Consistency Policy performs worse compared to RecFlow Policy.

We also report the runtime of 1-step DDIM inference. Although its speed is on a par with ours, it sacrifices most of the performance and thus fails in almost every task, similar to Consistency Policy.

Table 5: **Inference latency for one action prediction step in simulation (ms).** Simulation inference speeds were measured on an NVIDIA RTX 6000 Ada GPU and averaged over 400 rollouts. Benchmarking was done with vanilla Diffusion Policy and Consistency Policy since we used them as our baselines, and with 1-step DDIM since we also only solve once.

Algorithm	NFE	Inference Latency (ms)
RecFlow Policy	1	16.72
Diffusion Policy	100	1287
Consistency Policy	1	18.35
1-step DDIM	1	16.16

4.5 Robustness

During evaluation, we observed that RecFlow Policy shows more robustness than Diffusion Policy. Take Adroit door and pen as examples. Evaluated on one checkpoint iteratively using different inference seeds, our algorithm sees a lower variance in accuracy than that of Diffusion Policy. This is an extremely important attribute because in diffusion-based algorithms, different noise latent are heavily related to the final performance. A well-generalized algorithm should maintain a steady success rate among different initial noise samples to avoid fluctuation in performance during random evaluation. The robustness of RecFlow Policy may be mainly attributed to the straight and stable sampling trajectories of rectified flow modeling. Results are shown in Figure 8.

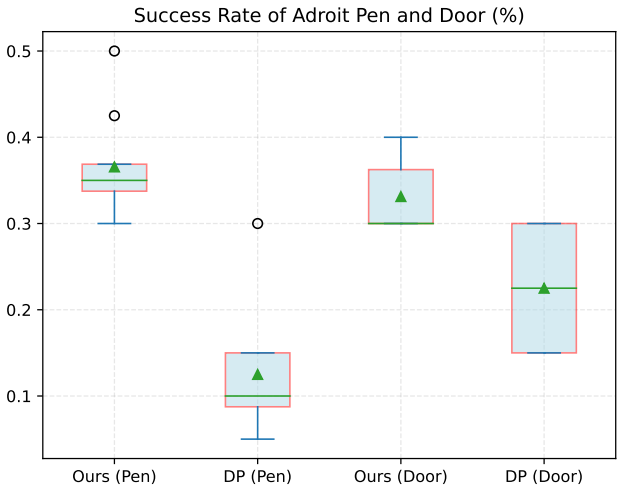


Figure 8: **Success Rate of Adroit (%).** We evaluate door and pen tasks to show the accuracy variance of RecFlow Policy and baseline Diffusion Policy (DP).

4.6 Ablation

3D inputs. To demonstrate our method has the capability to generalize to inputs with other modalities, we take the 3D point clouds as inputs. We compare RecFlow Policy with 3D Diffusion Policy (Ze et al., 2024) (DP3) on Adroit. We only substitute the policy network for DP3 policy. All the other settings are the same as the baseline, to make a fair comparison. The number of demos is 10, the same as reported in the 3D Diffusion Policy paper. We train the models with seed 0. We evaluate them with 3 seeds, and then report the average maximum success rate among them. As shown in Table 6, RecFlow Policy performs slightly better than 3D Diffusion Policy. Although 3D Diffusion Policy shows similar average accuracy, it is unstable and the performance varies among different initializations of the noise latent. This is reflected in the higher variance of the success rates. We further evaluate the models using 10 seeds and report the accuracy distribution in Figure 9.

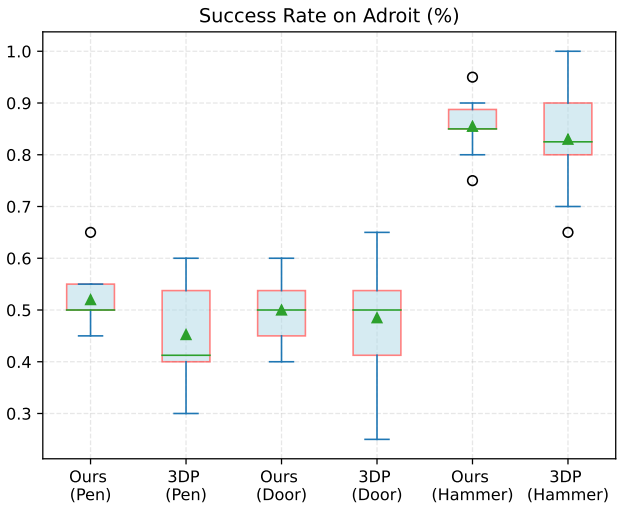


Figure 9: **Success Rate on 3D Adroit inputs (%).** We evaluate 3 simulated tasks to show the accuracy variance of RecFlow Policy and 3D Diffusion Policy (3DP).

Table 6: **Results on 3D inputs.** We compare RecFlow Policy and 3D Diffusion Policy on 3D inputs from Adroit.

Algorithm \ Task	Adroit			Average
	Hammer	Door	Pen	
RecFlow Policy	**95**$\pm$**4**	**57**$\pm$**6**	**59**$\pm$**4**	**70**$\pm$**5**
3D Diffusion Policy	83$\pm$12	53$\pm$12	50$\pm$15	62$\pm$13

Visual encoder. We test RecFlow Policy with different encoders for visual conditions, including widely-used pre-trained encoders, i.e., ViT-S/14 model of DINOv2 and the ViT-B/16 model of CLIP. As shown in Table 7, using pre-trained encoders experiences a drop in performance. There has been a long-standing bottleneck in collecting proficient human-teleoperated demonstrations in robotic settings. Large networks easily overfit to a small set of demonstrations, leading to performance degradation. Also, using a larger network increases the running time and overall training time. Experiments prove that our method easily performs well with our visual encoder trained from scratch, alleviating the burden of relying on a heavy, pre-trained one.

Table 7: **Ablation on encoders.** We replace RecFlow Policy Encoder with other widely-used pre-trained encoders, including pre-trained CLIP (Radford et al., 2021) encoder and DINOv2 (Oquab et al., 2023) encoder

Encoders	Network	Meta-World			RoboMimic	Average
		Door-unlock	Plate-slide-side	Pick-out-of-hole	Lift	
RecFlow Policy Encoder	CNN	**100**$\pm$**0**	**100**$\pm$**0**	**47**$\pm$**25**	**100**$\pm$**0**	**87**$\pm$**6**
CLIP encoder	ViT-B/16	33$\pm$9	87$\pm$9	0$\pm$0	91$\pm$4	53$\pm$6
DINOv2 encoder	ViT-S/14	80$\pm$0	**100**$\pm$**0**	7$\pm$9	**100**$\pm$**0**	72$\pm$2

Designs. Some works argue that although the distillation of the diffusion process can be used to accelerate policy synthesis, it is computationally expensive and can hurt both the accuracy and diversity of synthesized actions. In this paper, we show that although this statement holds true in some circumstances, we can still neutralize the drawbacks of *reflow* by our *refine* method. Here we test the performance of 1-RecFlow Policy, 2-RecFlow Policy, and *refine*d 2-RecFlow Policy. We choose 3 tasks with significantly different 1-RecFlow Policy accuracy on Meta-World and Adroit to prove our point. As presented in Table 8, on tasks with low success rate, such as pen and assembly, *reflow*ing may lead to poorer performance. However, the 2-RecFlow Policy score remains high when it comes to 100% 1-RecFlow Policy accuracy tasks, such as the plate-slide-side task. It proves our hypothesis that the performance degradation brought by *reflow* is a kind of error accumulation. Fortunately, our proposed *refine* technique compensates for accuracy loss and achieves a much higher score, sometimes even better than 1-RecFlow Policy. Note that π_0 Black et al. (2024) is equivalent to the first policy in Table 8. Although it is also trained via the flow matching loss, π_0 does not involve any *reflow* or *refine*, which effectiveness is underlined by the numbers in the last row in Table 8. This means that vanilla flow-based policies can be strengthened with our proposed methods, since ours can be implemented upon π_0 or any other policy that uses flow matching loss.

Table 8: **Ablation on *reflow* and *refine* design choices in RecFlow Policy.** We test our *refine* approach by training a *Refine*d 2-RecFlow Policy. Both 2-RecFlow Policy and *Refine*d 2-RecFlow Policy are trained upon the 1-RecFlow Policy in the first row.

Designs \ Task	#Steps	Adroit	Meta-World		Average
		Pen	Assembly	Plate Slide Side	
1-RecFlow Policy	100	43$\pm$6	27$\pm$9	**100**$\pm$**0**	56.7
***Reflow*ed 2-RecFlow Policy**	1	40$\pm$8	7$\pm$9	**100**$\pm$**0**	49.0
***Refine*d 2-RecFlow Policy**	1	**52**$\pm$**8**	**33**$\pm$**9**	**100**$\pm$**0**	**61.7**

Solver. A 1-step Euler solver works well in the default k-RecFlow Policy inference settings. To test its effect, we replace the 1-step solver with a 100-step solver. Intuitively, a solver with more steps would contribute to a higher accuracy since the truncation error introduced by using an approximation is decreased in every denoising loop. We also evaluate the performance of the Runge-Kutta method of order 5(4) from Scipy rk4, denoted as RK45. It adaptively decides the step size and number of steps N based on user-specified relative and absolute tolerances. This solver takes much longer time to reach its target when the policy is far from well-trained, since the flow direction is heavily deviated from $\mathbf{x}_0 - \mathbf{x}_T$ and what it does is just random walking. Nevertheless, RK45 should solve the ODE with only a few steps after the policy is trained. We choose the same RK45 parameters as Song et al. (2021b). Results are shown in Table 9. The RK45 solver shows the highest

success rate for RecFlow Policy without refinement, while the 100-step solver achieves the best performance for the refined 2-RecFlow Policy. While more denoising steps lead to higher accuracy, the difference between the 1-step solver and is reduced to 0 after the proposed refinement.

Notwithstanding the comparable success rate, a 1-step solver exceeds its 100-step counterpart in inference efficiency by approximately two orders of magnitude. Since the number of sampling steps of RK45 is not fixed to 1, it still lags behind RecFlow Policy with respect to efficiency.

Table 9: **Ablation on solver choices in RecFlow Policy.** We test the effect of the 1-step Euler solver by replacing it with a 100-step solver and a RK45 solver with adaptive timesteps N.

Solver \ Task		**Adroit** Pen	**Meta-World** Assembly	**Meta-World** Plate Slide Side	**Average**
1-RecFlow Policy	1-step solver	$43_{\pm 6}$	$27_{\pm 9}$	$\mathbf{100_{\pm 0}}$	$57_{\pm 5}$
	100-step solver	$45_{\pm 7}$	$30_{\pm 6}$	$\mathbf{100_{\pm 0}}$	$58_{\pm 4}$
	RK45 solver	$46_{\pm 7}$	$30_{\pm 9}$	$\mathbf{100_{\pm 0}}$	$\mathbf{59_{\pm 5}}$
2-RecFlow Policy	1-step solver	$40_{\pm 8}$	$7_{\pm 9}$	$\mathbf{100_{\pm 0}}$	$49_{\pm 6}$
	100-step solver	$40_{\pm 4}$	$25_{\pm 9}$	$\mathbf{100_{\pm 0}}$	$55_{\pm 4}$
	RK45 solver	$45_{\pm 6}$	$30_{\pm 8}$	$\mathbf{100_{\pm 0}}$	$\mathbf{58_{\pm 5}}$
Refined 2-RecFlow Policy	1-step solver	$52_{\pm 8}$	$33_{\pm 9}$	$\mathbf{100_{\pm 0}}$	$62_{\pm 6}$
	100-step solver	$\mathbf{52_{\pm 6}}$	$\mathbf{33_{\pm 6}}$	$\mathbf{100_{\pm 0}}$	$\mathbf{62_{\pm 4}}$
	RK45 solver	$50_{\pm 4}$	$31_{\pm 4}$	$\mathbf{100_{\pm 0}}$	$60_{\pm 3}$

Efficient scaling with demonstrations. We also evaluate the method's performance with respect to few-shot learning on 3 tasks in RoboMimic and 3 tasks in Meta-World. Table 10 shows that RecFlow Policy always obtains a success rate above 5% even with only 10% of the demonstrations. Note that the tasks we chose from Meta-World are all in *hard* and *very hard* levels, and *10%* refers to only 4 demonstrations. In RoboMimic tasks, RecFlow Policy always succeeds in more than 40% rollouts, even when the number of training demonstrations decreases to 5. This underscores that our method learns efficiently from training data.

5 Real World Experiments

Tasks. To comprehensively evaluate our method in the real world, we select 4 representative manipulation tasks: Placing Apples in a Fruit Bowl, Floating Object Manipulation, Flower Arrangement, and Mini Champagne Tower. Each task presents challenges in terms of precision, dexterity, and long-horizon.

1) Placing Apples in a Fruit Bowl: The robot arm is required to pick up an apple and place it into a fruit bowl. This task requires an accurate grasping pose for objects with irregular shapes and accurate pick-and-place capabilities.

2) Floating Object Manipulation: The robot arm is required to pick up a rubber duck that is floating on the water, move it a bit, and then put it back in the water. The rubber duck can move on the water and the interaction between the rubber duck, the gripper, and the water is quite complicated. This task requires the visuomotor policies to have spatial generalization to objects with distinct locations and effectively manipulate dynamic objects with visual observations.

3) Flower Arrangement: The robot arm is required to pick up a bouquet of flowers, change their orientations, and insert them into a narrow-neck vase. This task demands highly precise continuous control for successfully inserting deformable objects into a small opening, making it ideal for evaluating the imitation accuracy of visuomotor policies.

4) Mini Champagne Tower: The robot arm is required to build a three-layer champagne tower containing 10 champagne flutes: 6 on the first layer, 3 on the second, and 1 on the top. This involves sequentially picking up and precisely placing 3 champagne flutes at the center of the initial base layer of 6, followed by placing the final flute on top of the 3-flute layer. This task demands highly precise grasping poses to pick up the champagne flutes with very small openings, highly accurate place positions to maintain the stability of the champagne tower and also avoid collisions with other champagne flutes, making it ideal for evaluating visuomotor policies' long-horizon imitation accuracy.

Table 10: **Few-shot learning ability.** We use 10%, 25%, and 50% of the original demonstrations to test RecFlow Policy's few-shot learning performance.

Algorithm	#demos	Meta-World			RoboMimic				Average
		Disassemble	Pick-out-of-hole	Stick-Pull	Lift	Can	Square	Transport	
RecFlow Policy	100%	**87**$\pm$**9**	**47**$\pm$**25**	**13**$\pm$**19**	**100**$\pm$**0**	**100**$\pm$**0**	91$\pm$3	**89**$\pm$**5**	**75**$\pm$**9**
Diffusion Policy		60$\pm$16	0$\pm$0	7$\pm$9	**100**$\pm$**0**	99$\pm$1	**95**$\pm$**3**	89$\pm$7	64$\pm$5
RecFlow Policy	50%	13$\pm$9	**20**$\pm$**16**	**7**$\pm$**9**	**100**$\pm$**0**	**100**$\pm$**0**	**80**$\pm$**2**	**91**$\pm$**4**	**59**$\pm$**6**
Diffusion Policy		**40**$\pm$**0**	0$\pm$0	0$\pm$0	**100**$\pm$**0**	99$\pm$1	76$\pm$4	85$\pm$6	57$\pm$2
RecFlow Policy	25%	**13**$\pm$**9**	**7**$\pm$**9**	**27**$\pm$**19**	**100**$\pm$**0**	83$\pm$1	**60**$\pm$**2**	**89**$\pm$**3**	**54**$\pm$**6**
Diffusion Policy		**13**$\pm$**9**	0$\pm$0	7$\pm$9	**100**$\pm$**0**	**87**$\pm$**2**	57$\pm$2	87$\pm$3	50$\pm$4
RecFlow Policy	10%	**20**$\pm$**16**	**7**$\pm$**9**	**7**$\pm$**9**	**100**$\pm$**0**	**63**$\pm$**1**	**40**$\pm$**4**	63$\pm$5	**43**$\pm$**6**
Diffusion Policy		7$\pm$9	**7**$\pm$**9**	0$\pm$0	**100**$\pm$**0**	60$\pm$2	39$\pm$3	**65**$\pm$**6**	40$\pm$4

Table 11: **Main results for real robot experiments.** Each task is evaluated with 10 trials.

Real Robot Benchmark (4 Tasks)		
Task	Diffusion Policy	RecFlow Policy
Putting Apple in the Bowl	**100%**	**100%**
Floating Object Manipulation	60%	**70%**
Flower Arrangement	20%	**80%**
Champagne Tower	10%	**40%**

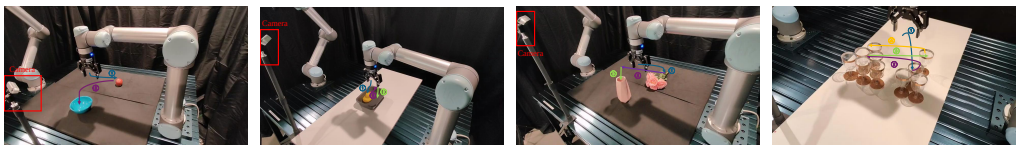
5.1 Experiment Setup

We use a 6-DoF UR5e robot arm with a 1-DoF Robotiq gripper as our test embodiment, and we use a RealSense 415 Camera mounted on the side of the workspace to capture visual observations. In addition, we leverage the Gello Wu et al. (2024) teleoperation system to collect real-world robotic demonstrations. To train the visuomotor policies, we collect 50 demos for each task of Placing Apples in a Fruit Bowl, Flower Arrangement, Mini Champagne Tower, and we collect 100 demos for Floating Object Manipulation. We adopt the Diffusion Policy Chi et al. (2023a) as the baseline method. For each task, we use a 480×640 image from the current timestep and proprioceptive data from the past six steps as inputs. The model outputs a sequence of 16 future actions. We adopt the same inputs, outputs, and model architecture as the Diffusion Policy for a fair comparison. Our real-world Experiment setup is shown in Figure 10.

5.2 Experimental Analysis

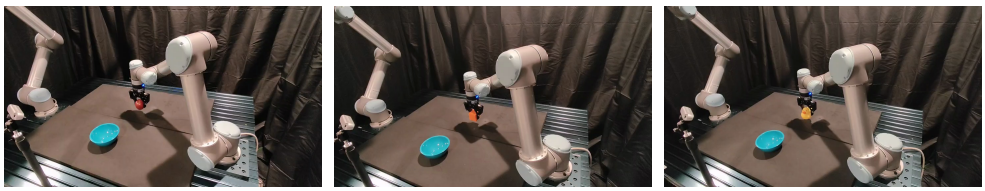
Real-world manipulation results. Table 11 summarizes the success rates compared to the Diffusion Policy in real-world robotic manipulation tasks. The results demonstrate that RecFlow Policy consistently outperforms Diffusion Policy in tasks requiring high precision, dexterity, and long-horizon sequential manipulation. For Placing Apples in a Fruit Bowl, both policies achieve a 100% success rate, indicating that they are equally capable of handling precise pick-and-place tasks for objects with irregular shapes. However, as the task complexity increases, RecFlow Policy exhibits superior performance, particularly in tasks that involve dynamic objects, deformable materials, or multi-step manipulation sequences. In Floating Object Manipulation, where the robot must handle a rubber duck floating on water, RecFlow Policy achieves 70% success, outperforming Diffusion Policy's 60%. The improvement suggests that the continuous rectification process in RecFlow Policy enables more stable and adaptive visuomotor control, allowing the policy to better handle dynamic interactions between the gripper, the floating object, and the water surface. The Flower Arrangement task, which requires delicate manipulation of deformable objects and precise insertion into a narrow vase, further highlights RecFlow Policy's advantages. RecFlow Policy achieves a success rate of 80%, significantly surpassing Diffusion Policy's 20%. This suggests that RecFlow Policy's ability to generate highly precise, single-step action predictions is particularly beneficial for tasks requiring continuous fine-grained control. The most challenging task, Mini Champagne Tower, involves long-horizon sequential manipulation and requires both precise grasping and stable object placement. Here, RecFlow Policy achieves 40% success, a notable improvement over Diffusion Policy's 10%. This improvement underscores RecFlow Policy's ability to generate stable and accurate actions in long-horizon manipulation scenarios.

Handling different object appearance. Figure 11 illustrates the ability of RecFlow Policy to perform grasping across different objects after being trained on a single instance (apple). The policy,



(a) **Putting Apple in the Bowl.** (b) **Floating Object Manipulation.** (c) **Flower Arrangement.** (d) **Mini Champagne Tower.**

Figure 10: **Real-world experimental setup.** We design 4 robotic manipulation tasks to evaluate the accuracy of visuomotor policies in the real world. Each task presents challenges in terms of precision, dexterity, and long-horizon sequential manipulation.



(a) **Grasping an apple.** (b) **Grasping an orange cube.** (c) **Grasping a rubber duck.**

Figure 11: **Grasping different objects with one policy.** RecFlow Policy trained on the apple can generalize to other objects (cube, rubber duck) with similar sizes and locations.

trained on grasping an apple, successfully executes grasping motions for objects with similar sizes and spatial locations, including an orange cube and a rubber duck. This suggests that RecFlow Policy can leverage learned visuomotor patterns to handle variations in object appearance while maintaining successful grasp execution.

Manipulating dynamic objects. Figure 12 illustrates how RecFlow Policy adjusts its action trajectory to approach and grasp a moving rubber duck on water. Unlike static object grasping, this task requires the policy to continuously refine its motion based on the real-time position of the floating object. The sequential images show that RecFlow Policy successfully tracks and aligns with the rubber duck's position, demonstrating its capability to handle object movement and generalize to different object locations. The results indicate that RecFlow Policy can generate effective visuomotor actions to compensate for object drift, allowing successful execution even when the object's location is not fixed. This highlights its suitability for real-world tasks that involve dynamic and continuously changing environments.

Comparison of manipulation accuracy. Figure 13 presents a qualitative comparison between the Diffusion Policy and RecFlow Policy in the Flower Arrangement task, which requires precise control to insert a bouquet into a narrow-neck vase. The sequential images highlight the differences in execution: while the Diffusion Policy struggles with accurate alignment and insertion, RecFlow Policy maintains a more stable and controlled trajectory, successfully positioning the bouquet into the vase. The improved performance of RecFlow Policy can be attributed to its ability to produce more direct and refined action trajectories, reducing deviations that could lead to failure. These results indicate that RecFlow Policy provides better control precision and stability, particularly in fine manipulation tasks that require high accuracy in both motion execution and final placement.

Long-horizon sequential manipulation. Figure 14 illustrates the execution of a complex long-horizon robotic manipulation task using RecFlow Policy — constructing a miniature champagne tower. This task requires precise grasping, careful placement, and sequential manipulation to ensure the stability of the tower. The sequential frames show that RecFlow Policy successfully completes the task by accurately positioning each champagne flute without causing instability or collisions. The success in this task highlights the ability of RecFlow Policy to handle multi-step decision-making while maintaining fine-grained control over object placement. This demonstrates the policy's effectiveness in long-horizon imitation learning, where accumulated errors in execution can significantly impact success rates. The ability to stably build structured object arrangements suggests that RecFlow Policy is well-suited for precise, sequential robotic tasks requiring sustained accuracy over multiple steps.

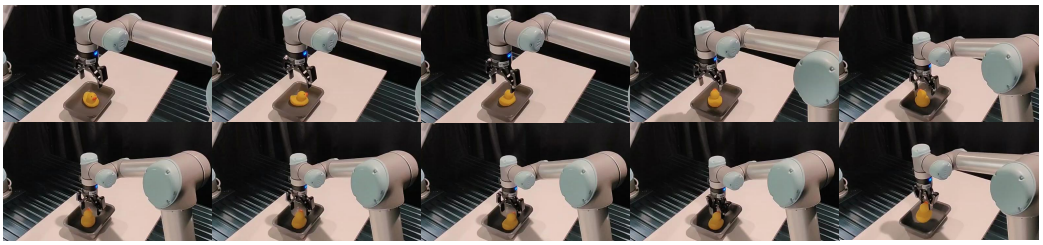
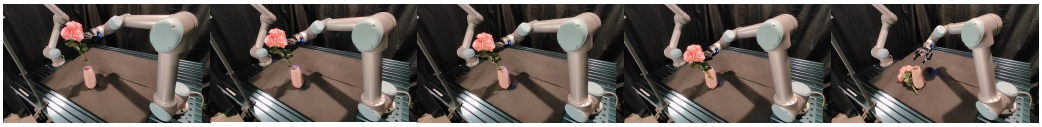
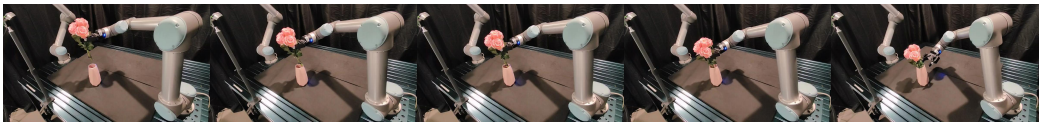


Figure 12: **Floating object manipulation.** RecFlow Policy dynamically adjusts its action trajectory to approach and grab the moving rubber duck on the water, which demonstrates generalization ability to different object locations.



(a) **Diffusion policy.**



(b) **RecFlow Policy.**

Figure 13: **Qualitative comparisons.** Compared to the diffusion policy, RecFlow Policy achieves higher manipulation accuracy, particularly in precise robotic manipulation tasks like inserting a bouquet into a narrow-neck vase.

6 Conclusion

In this work, we introduced RecFlow Policy, a fast and accurate visuomotor policy learning framework that replaces the iterative denoising process of diffusion models with rectified flow. Our approach enables efficient and precise action synthesis in one step, making it particularly suitable for real-time robotic manipulation. Through extensive real-world evaluations, RecFlow Policy demonstrated superior performance in tasks requiring high precision, dexterity, and long-horizon manipulation, such as flower arrangement and champagne tower construction. Compared to the diffusion policy, RecFlow Policy achieves higher success rates while significantly reducing inference latency, allowing for more responsive and stable robot execution. These results highlight the potential of RecFlow Policy as a scalable and effective solution for real-time visuomotor control in complex manipulation tasks.

7 Limitation

While RecFlow Policy achieves fast and accurate visuomotor policy learning, it still requires a substantial number of demonstrations to ensure strong performance across different tasks. Similar to Diffusion Policy and others, the reliance on high-quality expert demonstrations may limit its applicability in scenarios where large-scale data collection is challenging. Future work could focus on reducing the demonstration requirement through more effective data augmentation, self-supervised learning, or integrating multi-modal sensory inputs for enhanced adaptability.

Figure 14: **Long-horizon and highly-precise robotic manipulation.** RecFlow Policy successfully constructs a miniature champagne tower, demonstrating effective long-horizon imitation learning and highly precise sequential robotic manipulation.

REFERENCES

Scipy rk45 function. https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.RK45.html. Accessed: 2022-08-19.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Max Braun, Noémie Jaquier, Leonel Rozo, and Tamim Asfour. Riemannian flow matching policy for robot motion learning. *arXiv preprint arXiv:2403.10672*, 2024.

J. Carvalho, A.T. Le, M. Baierl, D. Koert, and J. Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

Kaiqi Chen, Eugene Lim, Lin Kelvin, Yiyang Chen, and Harold Soh. Don't Start From Scratch: Behavioral Refinement via Interpolant-based Policy Diffusion. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.122.

Lili Chen, Shikhar Bahl, and Deepak Pathak. Playfusion: Skill acquisition via diffusion from language-annotated play. In *CoRL*, 2023a.

Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84): eadc9244, 2023b. doi: 10.1126/scirobotics.adc9244.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems 2023*. Robotics: Science and Systems Foundation. ISBN 978-0-9923747-9-2. doi: 10.15607/RSS.2023.XIX.026. URL http://www.roboticsproceedings.org/rss19/p026.pdf.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023a.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023b. doi: 10.15607/RSS.2023.XIX.026.

Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal Manipulation Interface: In-The-Wild Robot Teaching Without In-The-Wild Robots. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.045.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *CoRL*, 2022.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long horizon tasks via imitation and reinforcement learning. *Conference on Robot Learning (CoRL)*, 2019.

Huy Ha, Pete Florence, and Shuran Song. Scaling up and distilling down: Language-guided robot skill acquisition. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 3766–3777. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/ha23a.html.

Xixi Hu, Bo Liu, Xingchao Liu, and Qiang Liu. Adaflow: Imitation learning with variance-adaptive flow-based policies. *arXiv preprint arXiv:2402.04292*, 2024.

Siyuan Huang, Zan Wang, Puhao Li, Baoxiong Jia, Tengyu Liu, Yixin Zhu, Wei Liang, and Song-Chun Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Sigmund H. Høeg, Yilun Du, and Olav Egeland. Streaming diffusion policy: Fast policy synthesis with variable noise diffusion models, 2024. URL https://arxiv.org/abs/2406.04806.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, pages 9902–9915. PMLR, 2022. URL https://proceedings.mlr.press/v162/janner22a.html. ISSN: 2640-3498.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. 35:26565–26577. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/a98846e9d9cc01cfb87eb694d946ce6b-Abstract-Conference.html.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ymjI8feDTD.

Kun Lei, Zhengmao He, Chenhao Lu, Kaizhe Hu, Yang Gao, and Huazhe Xu. Uni-o4: Unifying online and offline deep reinforcement learning with multi-step on-policy optimization. *arXiv*, 2023.

Xiang Li, Varun Belagali, Jinghuan Shang, and Michael S. Ryoo. Crossway diffusion: Improving diffusion-based visuomotor policy via self-supervised learning. URL http://arxiv.org/abs/2307.01849.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023a.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=XVjTT1nw5z.

Yunhao Luo, Chen Sun, Joshua B. Tenenbaum, and Yilun Du. Potential based diffusion motion planning. In *Forty-first International Conference on Machine Learning*, 2024.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv*, 2021.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *5th Annual Conference on Robot Learning*, 2021.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating human behaviour with diffusion models. In *The Eleventh International Conference on Learning Representations*, 2023.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation, 2024. URL http://arxiv.org/abs/2405.07503.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

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. *arXiv*, 2017.

Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. In *Robotics: Science and Systems 2023*. Robotics: Science and Systems Foundation. ISBN 978-0-9923747-9-2. doi: 10.15607/RSS.2023.XIX.028. URL http://www.roboticsproceedings.org/rss19/p028.pdf.

Moritz Reuss, Ömer Erdinç Yağmurlu, Fabian Wenzel, and Rudolf Lioutikov. Multimodal Diffusion Transformer: Learning Versatile Behavior from Multimodal Goals. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.121.

David Ruhe, Jonathan Heek, Tim Salimans, and Emiel Hoogeboom. Rolling diffusion models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 42818–42835. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/ruhe24a.html.

Kallol Saha, Vishal Mandadi, Jayaram Reddy, Ajit Srikanth, Aditya Agarwal, Bipasha Sen, Arun Singh, and Madhava Krishna. Edmp: Ensemble-of-costs-guided diffusion for motion planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10351–10358, 2024. doi: 10.1109/ICRA57147.2024.10610519.

Paul Maria Scheikl, Nicolas Schreiber, Christoph Haas, Niklas Freymuth, Gerhard Neumann, Rudolf Lioutikov, and Franziska Mathis-Ullrich. Movement primitive diffusion: Learning gentle robotic manipulation of deformable objects. *IEEE Robotics and Automation Letters*, 9(6): 5338–5345, 2024. doi: 10.1109/LRA.2024.3382529.

Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. In *CoRL*, 2023.

Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=agTr-vRQsa.

Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. URL http://arxiv.org/abs/2305.16317.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. a. URL https://openreview.net/forum?id=St1giarCHLP.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 32211–32252. PMLR, b. URL https://proceedings.mlr.press/v202/song23a.html. ISSN: 2640-3498.

Yang Song, Jascha Sohl-dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

Kaustubh Sridhar, Souradeep Dutta, Dinesh Jayaraman, James Weimer, and Insup Lee. Memory-consistent neural networks for imitation learning, 2023.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, 2012.

Julen Urain, Niklas Funk, Jan Peters, and Georgia Chalvatzaki. Se(3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion. *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

Che Wang, Xufang Luo, Keith Ross, and Dongsheng Li. Vrl3: A data-driven framework for visual deep reinforcement learning. *Advances in Neural Information Processing Systems*, 2022.

Lirui Wang, Jialiang Zhao, Yilun Du, Edward Adelson, and Russ Tedrake. PoCo: Policy Composition from and for Heterogeneous Robot Learning. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.127.

Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12156–12163. IEEE, 2024.

Zhou Xian, Nikolaos Gkanatsios, Theophile Gervet, Tsung-Wei Ke, and Katerina Fragkiadaki. Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2323–2339. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/xian23a.html.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *CVPR*, 2020.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*, 2020.

Yanjie Ze, Nicklas Hansen, Yinbo Chen, Mohit Jain, and Xiaolong Wang. Visual reinforcement learning with self-supervised 3d representations. *IEEE Robotics and Automation Letters*, 2023.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

Edwin Zhang, Yujie Lu, William Wang, and Amy Zhang. Lad: Language control diffusion: efficiently scaling through space, time, and tasks. *arXiv preprint arXiv:2210.15629*, 2023.

Zihan Zhang, Richard Liu, Rana Hanocka, and Kfir Aberman. Tedi: Temporally-entangled diffusion for long-term motion synthesis. In *ACM SIGGRAPH 2024 Conference Papers*, SIGGRAPH '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400705250. doi: 10.1145/3641519.3657515. URL https://doi.org/10.1145/3641519.3657515.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. volume 19. ISBN 978-0-9923747-9-2. URL https://www.roboticsproceedings.org/rss19/p016.html.

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 *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=gvdXE7ikHI.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.