Towards Safe Imitation Learning via Potential Field-Guided Flow Matching

Haoran Ding[1], Anqing Duan[1], Zezhou Sun[1], Leonel Rozo[2], Noémie Jaquier[3], Dezhen Song[1], Yoshihiko Nakamura[1]

***Abstract*— Deep generative models, particularly diffusion and flow matching models, have recently shown remarkable potential in learning complex policies through imitation learning. However, the safety of generated motions remains overlooked, particularly in complex environments with inherent obstacles. In this work, we address this critical gap by proposing Potential Field-Guided Flow Matching Policy (PF2MP), a novel approach that simultaneously learns task policies and extracts obstacle-related information, represented as a potential field, from the same set of successful demonstrations. During inference, PF2MP modulates the flow matching vector field via the learned potential field, enabling safe motion generation. By leveraging these complementary fields, our approach achieves improved safety without compromising task success across diverse environments, such as navigation tasks and robotic manipulation scenarios. We evaluate PF2MP in both simulation and real-world settings, demonstrating its effectiveness in task space and joint space control. Experimental results demonstrate that PF2MP enhances safety, achieving a significant reduction of collisions compared to baseline policies. This work paves the way for safer motion generation in unstructured and obstacle-rich environments.**

I. INTRODUCTION

Imitation learning (IL) [1] has seen transformative advancements with the introduction of generative models [2], [3], [4]. Among these, diffusion models [5], [6] and flow matching models [7], [8] have emerged as powerful tools, enabling scalable and efficient policy learning. By capturing complex, high-dimensional, and multi-modal distributions, these methods have established generative models as the foundation of next-generation IL frameworks [9].

Despite these advances, a critical aspect remains insufficiently explored: the safety of the generated policies [10]. Safety is a critical consideration in real-world applications such as autonomous driving [11] and robotic manipulation [12], where the consequences of unsafe actions can be catastrophic. Recent efforts have begun to incorporate safety considerations into generative models, with notable progress in diffusion-based IL. For example, some methods enhance safety by embedding gradients of safety objectives into the denoising process [13], [14]. However, similar advancements in Flow Matching-based IL are mainly absent. Current Flow Matching-based IL primarily focus on accurately modeling and reproducing expert demonstrations, often overlooking potential hazards or obstacles inherent in the environment. This gap presents a significant challenge for deploying these methods in safety-critical domains.

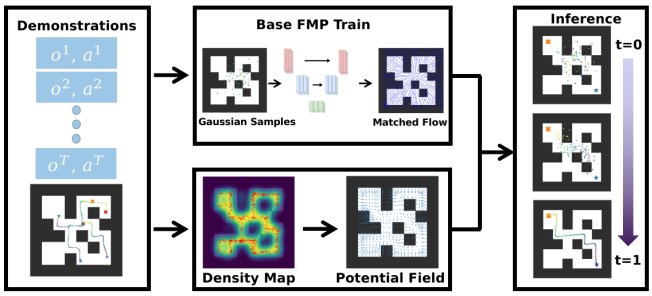


Fig. 1: Overview of the PF2MP Framework: The pipeline begins with demonstration data (left). The Base Flow Matching Policy (FMP) is trained (middle top) to model expert behaviors, while a potential field is simultaneously learned (middle bottom) using a density map. During the inference phase (right), the PF2MP framework integrates the learned vector field from FMP and the obstacle-aware potential field to generate safe trajectories.

To address this limitation, we propose a novel approach to enhance the safety of Flow Matching policies by incorporating obstacle-aware potential fields into the policy generation process. The key insight of our method is to leverage demonstration data not only to learn the desired policy but also to design a potential field derived from the density of the demonstrations estimated via Kernel Density Estimation (KDE). By guiding the flow matching inference process with the estimated potential field, our approach adjusts trajectories in response to environmental hazards encountered during demonstrations, effectively bridging efficient policy generation with safety assurance.

In summary, we enhance Flow Matching-based IL by incorporating an estimated potential field that accounts for the inherent obstacles in the environment, resulting in the **P**otential **F**ield-Guided **F**low **M**atching **P**olicy (**PF2MP**), illustrated in Fig 1. In contrast to existing diffusion model-based IL [13], [14], [15], which rely on manually-defined safety objective functions, our approach autonomously derives safety constraints directly from expert demonstrations. Through extensive experiments in both simulated and real-world environments, we demonstrate that our model effectively improves the safety of the learned policy while

[1]Haoran Ding, Anqing Duan, Zezhou Sun, Dezhen Song and Yoshihiko Nakamura are with the Department of Robotics, Mohamed bin Zayed University of Artificial Intelligence (MBZUAI), Abu Dhabi, UAE. {haoran.ding, anqing.duan, zezhou.sun, dezhen.song, yoshihiko.nakamura}@mbzuai.ac.ae

[2]Leonel Rozo is with Bosch Center for Artificial Intelligence, Renningen, Germany. leonel.rozo@de.bosch.com

[3]Noémie Jaquier is with the Division of Robotics, Perception, and Learning, KTH Royal Institute of Technology, Stockholm, Sweden. jaquier@kth.se

Manuscript 951 submitted to 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received February 26, 2025.

preserving task performance.

II. RELATED WORK

Ensuring safety is a critical challenge in robot policy learning, with most studies focusing on reinforcement learning [12], [16] and imitation learning [13], [17], [18], [19], [20]. Recently, generative models [5], [6], [7] have made transformative advancements to robot policy learning, particularly in the field of imitation learning. Here we review recent developments, categorizing them into two major approaches: Diffusion-Based Robot Policies and Flow Matching-Based Robot Policies.

A. Diffusion-Based Robot Policies

Diffusion models [21], which iteratively transform a Gaussian distribution into a complex target distribution through denoising steps, have achieved impressive success in various application domains, including image generation [5], video synthesis [22], and human motion modeling [23], among others. In robotics, Diffusion Policy (DP) [3], primarily trained via Denoising Diffusion Probabilistic Model (DDPM), demonstrated improved performance in imitation learning compared to earlier behavioral cloning methods [24], [25], [26] with an enhanced ability to capture multimodal action distributions. Consistency Policy (CP) [27] addresses the expensive inference process of DP by distilling a student policy from a teacher DP, effectively reducing computational overhead. Furthermore, diffusion models have demonstrated strong performance in complex robot policy learning tasks [28]. For example, 3D Diffusion Policy (DP3) [29] combines a compact 3D visual representation with diffusion models, enabling effective policy learning in more intricate environments.

Several works proposed enhanced diffusion-based policies to generate safer robot motions. Motion Planning Diffusion (MPD) [15] guides the diffusion process based on the gradients of multiple objective functions, thereby enhancing the smoothness and safety of the generated trajectories. Similarly, Ensemble-of-costs-guided Diffusion for Motion Planning (EDMP) [14] incorporates the gradients of multiple collision cost functions during denoising, significantly enhancing the safety of generated motions. Cold Diffusion with Replay Buffers (CDRP) [13] augments DP by recording the intermediate variables generated during the denoising process on the training dataset and storing them as a safe replay buffer. During testing, CDRP projects the current denoising intermediate variables onto this buffer, effectively anchoring the inference process to observed safe states. This approach enables CDRP to produce robotic trajectories with reduced collision rate.

B. Flow Matching-Based Robot Policies

Flow Matching (FM) [7], [30] is another class of generative models that transform a simple prior distribution (e.g., a Gaussian distribution) into a complex target distribution via a vector field. In contrast to diffusion models, the FM vector field induces straighter paths and its flow corresponds to an ordinary differential equation (ODE). Therefore, FM is easier to train and faster to query during inference when compared to diffusion models.

Capitalizing on these advantages, several works recently proposed to leverage FM in robot imitation learning. Braun *et al.* introduced Riemannian Flow Matching Policy (RFMP) [4] to learn sensorimotor policies represented by end-effector pose trajectories and ensuring geometric constraints in the generated policies. Stable Riemannian Flow Matching Policy (SRFMP) [31] enhances the robustness and inference speed of RFMP by stabilizing the convergence of the flow to the target distribution. ActionFlow [32] employs FM with a $\mathrm{SE}(3)$ flow and an equivariant transformer to learn $\mathrm{SE}(3)$-equivariant policies that effectively handle relative positions between the end-effector and objects. FM was also applied to multi-support manipulation tasks in [33], to control a bimanual robot interacting with multiple environmental supports. Additionally, Chisari *et al.* [34] proposed to employed 3D point clouds instead of images as observations, which resulted in improved FM policies performances.

While FM-based methods methods offer efficiency and flexibility, none of these works addressed the safety of the generated FM policies. However, ensuring safe policies is critical in real-world robotics applications, especially in environments with inherent obstacles and dynamic constraints. In this paper, we aim to extend the capabilities of FM-based robot policies by learning not only complex policies from expert demonstrations, but also potential fields to avoid unsafe regions and collisions with the obstacles observed in the demonstrations.

III. BACKGROUND

In this section, we provide a short background on Flow Matching and Kernel Density Estimation (KDE), which serve as the foundational frameworks for our method.

A. Flow Matching

Flow Matching (FM) [7] is a deep generative model that transforms a simple base distribution $p_0(\boldsymbol{x})$, typically a Gaussian distribution, into an unknown target distribution $q(\boldsymbol{x}_1)$ through a learned vector field. The probability path is generated by push-forwarding p_0 along the resulting flow, which is governed by an ordinary differential equation (ODE). While the exact form of the target distribution $q(\boldsymbol{x}_1)$ is unknown, it is assumed that samples $\boldsymbol{x}_1$ drawn from $q(\boldsymbol{x}_1)$ are accessible.

Conditional flow matching (CFM) [7] constructs a conditional probability path $p_t(\boldsymbol{x}_t|\boldsymbol{x}_1)$ linking a Gaussian base distribution $p_0(\boldsymbol{x})$ and the target distribution $q(\boldsymbol{x}_1)$ by conditioning on the available samples $\boldsymbol{x}_1$. The probability path is defined as,

$$p_t(\boldsymbol{x}|\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}|\mu_t(\boldsymbol{x}_1), \sigma_t(\boldsymbol{x}_1)^2\boldsymbol{I}), \tag{1}$$

where $\mu_t(\boldsymbol{x}_1)$ and $\sigma_t(\boldsymbol{x}_1)$ are time-dependent mean and scalar standard deviation, respectively. The flow ψ_t that

Manuscript 951 submitted to 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received February 26, 2025.

pushes the base distribution to the target distribution can be expressed as,

$$\psi_t(\boldsymbol{x}|\boldsymbol{x}_1) = \sigma_t(\boldsymbol{x}_1)\boldsymbol{x} + \mu_t(\boldsymbol{x}_1), \tag{2}$$

and describes the trajectory of the variable $\boldsymbol{x}_t$ as it moves from the base distribution to the target distribution. Lipman *et al.* [7] introduce a general formulation of the conditional vector field u_t that generates such flow ψ_t,

$$u_t(\boldsymbol{x}|\boldsymbol{x}_1) = \frac{\sigma_t'(\boldsymbol{x}_1)}{\sigma_t(\boldsymbol{x}_1)}(\boldsymbol{x} - \mu_t(\boldsymbol{x}_1)) + \mu_t'(\boldsymbol{x}_1), \tag{3}$$

where $\mu_t'(\boldsymbol{x}_1)$ and $\sigma_t'(\boldsymbol{x}_1)$ are the derivatives of the mean and standard deviation functions. By integrating the vector field (3) with initial points $\boldsymbol{x}_0$ under a predefined time boundary, the result aligns with the target distribution. The training process of CFM can be framed as a regression problem. A neural network is trained to mimic the conditional vector field $u_t(\boldsymbol{x}|\boldsymbol{x}_1)$ by minimizing a mean squared error loss,

$$\mathcal{L}_{\text{CFM}}(\boldsymbol{\theta}) = \mathbb{E}_{t,q(\boldsymbol{x}_1),p(\boldsymbol{x}_0)}||v_\theta(\boldsymbol{x}_t, t) - u_t(\boldsymbol{x}|\boldsymbol{x}_1)||^2, \tag{4}$$

where $v_{\boldsymbol{\theta}}$ is the learned vector field, $\boldsymbol{\theta}$ represents the parameters of the neural network and t is uniformly sampled within the time boundary.

Liu *et al.* [30] proposed a more direct approach by simplifying the conditional vector field to

$$u_t(\boldsymbol{x}|\boldsymbol{z}) = \boldsymbol{x}_1 - \boldsymbol{x}_0, \tag{5}$$

where $\boldsymbol{z} = (\boldsymbol{x}_0, \boldsymbol{x}_1)$. This simplified vector field is conditioned not only on the sample $\boldsymbol{x}_1$ from the target distribution but also on the sample $\boldsymbol{x}_0$ from the base distribution. By connecting the two distributions with a straight path, this approach significantly reduces the computational complexity of the vector field calculation and accelerates the inference process [30]. PF2MP leverages [30] to train a base Flow Matching Policy to mimic the complex expert policy (see Section IV-A).

B. Kernel Density Estimation

Kernel Density Estimation (KDE) [35] is a non-parametric method for estimating the probability density function (PDF) of a random variable based on a finite sample of observations. Unlike parametric methods that assume that the data follows a specific distribution (e.g., Gaussian or exponential), KDE makes minimal assumptions about the underlying data distribution, making it a versatile and robust technique in statistical analysis and machine learning.

KDE seeks to estimate the density function $f(\boldsymbol{x})$ of a random variable $\boldsymbol{X}$, given M independent and identically distributed samples $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_M$. The KDE kernel estimator is expressed as,

$$\hat{f}(\boldsymbol{x}) = \frac{1}{Mh}\sum_{i=1}^{M} K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h}\right), \tag{6}$$

where $K(\cdot)$ is the kernel function, i.e., a symmetric, non-negative, and integrable function that determines the shape of the distribution estimate, h is a positive scalar representing bandwidth or smoothing parameter that controls the width of the kernel function, and thus the smoothness of the estimated density function [36]. KDE is employed to estimate the density of the demonstrations, which is subsequently used to construct the potential field in PF2MP (see Section IV-B).

Algorithm 1 Base Flow Matching Policy Training Process

Input: Initialized parameter $\boldsymbol{\theta}$, prior distribution p_0, demonstration $D = \{\boldsymbol{o}^\tau, \boldsymbol{a}^\tau\}_{\tau=1}^T$, training epochs N_T
Output: Parameter $\boldsymbol{\theta}$ of the learned vector field

```
for i = 1 : N_T do
    Sample a time step t from uniform distribution U[0, 1];
    Sample a noise A_0 from prior distribution p_0;
    Sample a target action series A_1 and the paired observation o from the demonstrations;
    Build the target vector field u_t (7);
    Calculate the loss function (8);
    Update the parameters θ;
end for
return θ
```

IV. POTENTIAL FIELD-GUIDED FLOW MATCHING POLICY

In this section, we introduce the **P**otential **F**ield-Guided **F**low **M**atching **P**olicy (**PF2MP**), illustrated in Fig 1. PF2MP is composed of two components trained from the same set of demonstrations: (1) a base Flow Matching Policy (Section IV-A); and (2) an estimated potential field (Section IV-B). By combining these two components during the inference process (Section IV-B), PF2MP generates safer policies that incorporate the environmental constraints observed in the demonstrations.

A. Base Flow Matching Policy

This section introduces Flow Matching Policy (FMP), as presented in prior works [4], [32], [33], [34], [31], [37], which constitutes the base policy of PF2MP.

We consider a set of demonstrations $D = \{\boldsymbol{o}^\tau, \boldsymbol{a}^\tau\}_{\tau=1}^T$, where $\boldsymbol{o}^\tau$ represents the environment observation, $\boldsymbol{a}^\tau$ denotes the corresponding action taken by the agent, and T is the total length of the trajectory. The objective of imitation learning is to train a policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}; \boldsymbol{o})$ that mimics the expert policy $\pi_e(\boldsymbol{a}; \boldsymbol{o})$, which generated the demonstrations.

Flow Matching Policy (FMP) leverages a flow matching model — specifically based on [30] in this paper — to learn an observation-conditioned vector field $u_t(\boldsymbol{A}|\boldsymbol{z}, \boldsymbol{o}^\tau)$, defined as

$$u_t(\boldsymbol{A}|\boldsymbol{z}, \boldsymbol{o}^\tau) = \boldsymbol{A}_1 - \boldsymbol{A}_0, \tag{7}$$

where $t \in [0, 1]$ represents the time step during the generation process, $\boldsymbol{A}$ denotes the intermediate variable during the generation process, the variable $\boldsymbol{z} = (\boldsymbol{A}_0, \boldsymbol{A}_1)$ includes a sample $\boldsymbol{A}_0$ from a prior distribution (typically a Gaussian distribution) with the same dimensionality as $\boldsymbol{A}_1$ and the target action series $\boldsymbol{A}_1 = \{\boldsymbol{a}^\tau, \boldsymbol{a}^{\tau+1}, ..., \boldsymbol{a}^{\tau+T_a-1}\}$ of length

Manuscript 951 submitted to 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received February 26, 2025.

T_a, and $\boldsymbol{o}^\tau$ is the corresponding observation at the begin of the target action series.[1]

The vector field network $v_\theta(\boldsymbol{A}, t; \boldsymbol{o})$ is learned using the following loss function,

$$\mathcal{L}_{\text{FMP}}(\boldsymbol{\theta}) = \mathrm{E}_{t, q(\boldsymbol{A}_1), p(\boldsymbol{A}_0)} ||v_\theta(\boldsymbol{A}, t; \boldsymbol{o}) - u_t(\boldsymbol{A}|\boldsymbol{z}, \boldsymbol{o})||^2, \quad (8)$$

where $q(\boldsymbol{A}_1)$ and $p(\boldsymbol{A}_0)$ are the target and prior distributions, respectively. The training process of FMP, summarized in Algorithm 1, consists of three steps: (1) Sample from the prior and target distributions; (2) Compute the target conditioned vector field using (7); (3) Update the neural network parameters $\boldsymbol{\theta}$ using the loss function (8).

After training the observation-conditioned vector field, the policy inference process boils down to integrating the learned vector field, starting from samples drawn from the prior Gaussian distribution. As the analytical solution to this integration is generally intractable, numerical methods are commonly employed. Among these, the Euler method [38] provides a straightforward and effective approximation. The Euler method approximates the integration of the vector field by discretizing the integration into small time steps. Starting with an initial sample $\boldsymbol{A}_0$ from the prior Gaussian distribution and observation $\boldsymbol{o}$ of the environment, the integration process iteratively updates the state as follows,

$$\boldsymbol{A}_{t+\Delta t} = \boldsymbol{A}_t + v_\theta(\boldsymbol{A}, t; \boldsymbol{o}) \Delta t, \quad (9)$$

where Δt is the integration step size and related to total number of integration steps N via $\Delta t = 1/N$. The final $\boldsymbol{A}_1$, generated at $t = 1$ is the action series. Finally, each action is given sequentially to the robot to perform the task.

B. Safe Policy Generation with PF2MP

The base FMP primarily focuses on replicating expert demonstrations to reproduce observed behaviors. However, it overlooks critical considerations, such as the safety of the generated action sequences, which are essential for real-world deployment, especially in obstacle-rich environments.

Expert demonstrations inherently encode more than just the execution of complex policies to accomplish specific tasks; they also implicitly capture valuable information about the environment's constraints (e.g., obstacles). The regions explored by the expert policies represent safe areas, while areas not traversed by the demonstrations often correspond to unsafe or high-risk regions. By neglecting this implicit safety information, previous algorithms risk generating policies that, although behaviorally accurate, may lead to unsafe or infeasible actions when deployed in obstacle-rich environments.

To address this limitation, we propose the Potential Field-Guided Flow Matching Policy (PF2MP), which leverages the inherent constraint information embedded in the demonstration data. Specifically, our approach estimates the density $\hat{p}(\boldsymbol{a})$ of the demonstration trajectories via KDE and constructs a potential function $\phi(a)$ based on the estimated density as,

$$\phi(\boldsymbol{a}) = \log(\hat{p}(\boldsymbol{a})) + \alpha d(\boldsymbol{a}, \mathcal{H}), \quad (10)$$

which consists of 2 components, whose contributions are balanced by the hyperparameter α. The first term, $\log(\hat{p}(\boldsymbol{a}))$, represents the logarithm of the estimated density, while the second term, $d(\boldsymbol{a}, \mathcal{H})$, is a distance-based penalty that measures the proximity of an action $\boldsymbol{a}$ to the safer area $\mathcal{H}$. The safer area is defined as the region where the estimated density exceeds a predefined threshold, indicating regions covered by the expert demonstrations.

The resulting potential field $\Phi(A)$ is the gradient of the potential function across the action series, i.e.,

$$\Phi(\boldsymbol{A}) = \left[\frac{d\phi}{d\boldsymbol{a}^\tau}, \frac{d\phi}{d\boldsymbol{a}^{\tau+1}}, ..., \frac{d\phi}{d\boldsymbol{a}^{\tau+T_a-1}} \right]. \quad (11)$$

The policy inference process of PF2MP integrates the learned flow matching vector field with the designed potential field into a unified framework. As summarized in Algorithm 2, the procedure begins by sampling an initial point $\boldsymbol{A}_0$ from a prior Gaussian distribution, representing the initial state of the generated action series. At each subsequent integration step, the intermediate state $\boldsymbol{A}_t$ is updated by following the learned FM vector field $v_\theta(\boldsymbol{A}, t|\boldsymbol{o})$, which has been trained to mimic the expert demonstrations, augmented with the potential field $\Phi(\boldsymbol{a})$, constructed from the density estimation of the demonstration data. The potential field acts as a corrective term that incorporates safety constraints by steering the generated actions away from low-density (unsafe) regions and toward high-density (safe) regions. Mathematically, the update at each integration step is expressed as,

$$\boldsymbol{A}_{t+\Delta t} = \boldsymbol{A}_t + [v_\theta(\boldsymbol{A}_t, t; \boldsymbol{o}) + \lambda \Phi(\boldsymbol{A}_t)] \Delta t, \quad (12)$$

where λ is a hyperparameter controlling the trade-off between flow matching dynamics and potential field guidance, and Δt represents the integration step size of the Euler method. This iterative process is repeated N times, where N

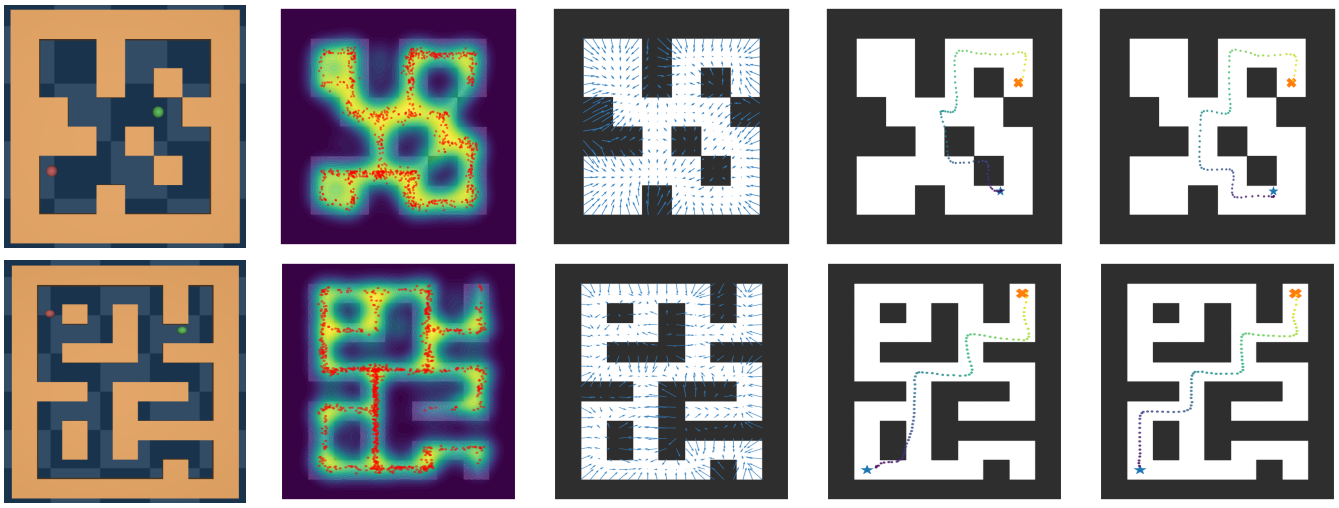
Algorithm 2 Potential Field-Guided Flow Matching Inference

Input: Trained vector field $v_\theta(\boldsymbol{A}, t; \boldsymbol{o})$, number of integration steps N, observation $\boldsymbol{o}$, prior distribution p_0, hyperparameter λ, potential field $\Phi(\boldsymbol{A})$

Output: Generated action series $\boldsymbol{A}_1$.

1: Compute the timestep $\Delta t = \frac{1}{N}$;
2: Sample from prior distribution $\boldsymbol{A}_0 \sim p_0$;
3: **for** $i = 1 : N$ **do**
4: Integrate the vector field learned by FMP as $\boldsymbol{A}_{t_i} = \boldsymbol{A}_{t_{i-1}} + \left[v_\theta(\boldsymbol{A}_{t_{i-1}}, t_{i-1}; \boldsymbol{o}) + \lambda \Phi(\boldsymbol{A}_{t_{i-1}}) \right] \Delta t$ (11)
5: **end for**
6: **return** $\boldsymbol{A}_1 = \left\{ \boldsymbol{a}^\tau, \boldsymbol{a}^{\tau+1}, ..., \boldsymbol{a}^{\tau+T_a-1} \right\}$

[1]In this paper, the subscript t represents the time step of the FM generation process, and the superscript τ indicates the real-world time step of the robot trajectory.



(a) Gym Environment (b) Density Map (c) Potential Field (d) FMP (e) PF2MP

Fig. 2: 2D maze navigation tasks: (a) Gym environment with the medium maze (top row) and large maze (bottom row). (b) Estimated density map, with red points representing the demonstration data. (c) Derived potential field, visualized with blue arrows, based on the estimated density. (d) Failure case of the baseline FMP, where collisions occur, with the blue star indicating the start point and the orange fork marking the goal. (e) Collision-free trajectory generated by the PF2MP under the same initial and target conditions as in (d).

is a hyperparameter determining the resolution of the integration. By integrating safety-aware potential fields into the flow matching framework, PF2MP generates action sequences that not only replicate expert behavior but also account for the obstacles present in the demonstrations, ensuring both effectiveness and safety in complex environments.

V. EXPERIMENT

We conducted four sets of experiments, including navigation and robot manipulation tasks in simulated environments, as well as a real-world robotic task. These experiments are designed to comprehensively evaluate the effectiveness of the proposed PF2FMP in enhancing safety with task execution controlled in task space or joint space.

A. Basic Settings

We implement a conditional 1D U-Net similar to [3], [31], as the base network for learning the vector field of the base FM policy. The U-Net consists of a three-layer structure with hidden dimensions [512, 256, 128], designed to efficiently model the observation-conditioned vector field. For comparison, we evaluate PF2MP against three baselines: base Flow Matching Policy (FMP), Diffusion Policy (DP), and DP with Potential Field Guidance (DPPF) as a modified baseline, where the same potential field guidance employed in PF2MP is integrated into the denoising process of DP.

To measure the performance of different policies, we consider the collision rate, given as the percentage of trajectories colliding with obstacles in the environment. For the 2D Maze (Section V-B) and Fetch (Section V-C) tasks, we also consider the task success rate, irrespectively of collisions during the execution.

B. 2D Maze Navigation Tasks

We begin by evaluating our approach on a navigation task in the 2D Maze environment provided by D4RL [39], [40],

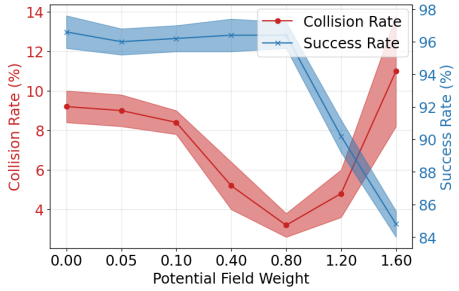


Fig. 3: Impact of the potential field weight λ on PF2MP performance in the large maze task.

which is built upon the Mujoco physics engine [41]. As illustrated in Fig 2, the environment consists of brown obstacles and boundaries. The controllable agent is represented by a green ball, whose objective is to navigate through the maze to reach a red ball. The training dataset, provided by D4RL, includes trajectories with random start and goal states within the safe region. To ensure trajectory smoothness, we resample the demonstrations to a fixed length of 80 steps, which is sufficient to complete the task effectively. D4RL provides mazes with varying levels of complexity. For our experiments, we select the "medium" and "large" maze configurations. In this task, we utilize PF2MP to predict the trajectory of the green ball, assuming the existence of a low-level controller capable of precisely reaching the predicted states. The observation $\boldsymbol{o}$ is the position of the green and red balls. To construct the potential field, we randomly sample $3,000$ points from the demonstrations instead of using the entire dataset to reduce computational overhead.

We evaluate PF2MP and the baseline methods using the same 500 pairs of initial and target states, repeating each test three times. All models are inferred with $N = 5$ integration steps. The weight λ for PF2MP and DPPF is set to 0.8.

The performance of all models are reported in Table I. In the medium maze, all models exhibit strong performance,

TABLE I: Task performance of different models on 2D maze navigation and Fetch robot manipulation tasks.

Tasks	Success Rate (%)				Collision Rate (%)			
	FMP	**PF2MP**	DP	DPPF	FMP	**PF2MP**	DP	DPPF
Maze Medium	$\mathbf{99.7 \pm 0.1}$	99.6 ± 0.2	99.6 ± 0.2	99.6 ± 0.2	0.33 ± 0.12	$\mathbf{0.07 \pm 0.07}$	0.47 ± 0.12	0.27 ± 0.12
Maze Large	96.6 ± 1.1	96.4 ± 1.2	$\mathbf{96.7 \pm 0.8}$	96.4 ± 1.2	9.2 ± 0.8	$\mathbf{3.2 \pm 0.5}$	8.8 ± 0.7	7.6 ± 0.5
Fetch Reach	$\mathbf{85.0 \pm 4.4}$	84.0 ± 1.7	84.0 ± 3.5	83.0 ± 2.0	34.0 ± 5.6	$\mathbf{18.0 \pm 4.4}$	35.0 ± 4.4	33.0 ± 3.0
Fetch Push	50.0 ± 4.6	$\mathbf{51.7 \pm 2.5}$	48.3 ± 2.5	47.0 ± 3.6	30.0 ± 3.6	$\mathbf{25.0 \pm 2.6}$	33.3 ± 5.7	30.3 ± 4.7

with collision rates below 0.4% and success rates exceeding 99%. PF2MP achieves a marginally lower collision rate compared to FMP while maintaining an equivalent success rate. A similar trend is observed with DPPF compared to DP, where a slight reduction in collision rate is evident. In the large maze, PF2MP demonstrates a significant reduction in collision rate compared to FMP, achieving the lowest collision rate ($\sim 3\%$) among all models, while maintaining a comparable success rate to other baselines. These results indicate that the potential field guidance enhances the safety of the generated trajectories without adversely affecting task performance. This improvement is consistently observed across both FMP and DP, underscoring the general effectiveness of the potential field guidance.

To further analyze the effect of the potential field guidance, we conducted an ablation study on the potential field weight λ using the large maze setting (see results in Fig 3). When λ is set too low ($\lambda < 0.1$), the performance of PF2MP is nearly identical to the baseline FMP. As λ increases to values such as 0.4 and 0.8, the collision rate significantly decreases, and the task success rate remains stable. However, when λ is further increased, the collision rate rises dramatically, and the success rate decreases accordingly. We hypothesize that this occurs because a high λ leads to too-narrow safe area: As the agent tries to follow the demonstrated trajectory, it overshoots and collides with the opposite obstacle. This phenomenon suggests that while PF2MP can enhance safety in action generation, careful selection of the λ parameter is crucial for optimal performance.

C. Simulated Robotics Tasks in Task Space

We evaluate PF2MP on robot manipulation tasks by considering two benchmark tasks — Reach and Push — performed using the Fetch robot simulator from Gymnasium Robotics [42]. To evaluate the safety of the proposed method, we augment the original Fetch environment by introducing obstacles, as shown in Fig 4. This setup is inspired by and adapted from the experimental framework presented in [13]. In the Reach task, a thick red wall is placed as an obstacle in the middle of the platform. The robot's end-effector is initialized on the right side of the wall, while the target goal, represented by a red ball, is randomly initialized on the left side of the wall. The objective is to control the robot to navigate around the obstacle and reach the goal point. In the Push task, a similar thick red wall is present in the middle of the platform, but with a reduced height compared to the one used in the Reach task. The objective is to control the robot to push a black block to a target position marked by a red ball. Both the black block and the red ball are initialized on the surface of the platform.

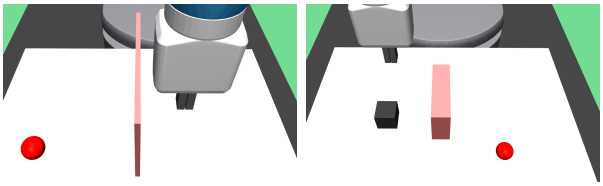


Fig. 4: Fetch Environments. Left: Reach task. Right: Push task.

To collect expert demonstrations, we employ the Stable-Baselines3, a Python library [43] for Reinforcement Learning. For the Reach task, we train a reinforcement learning agent using Proximal Policy Optimization (PPO) [44]. For the Push task, we employ Soft Actor-Critic (SAC) [45] combined with Hindsight Experience Replay (HER) [46] as the replay buffer. Both tasks are trained for a total of 1×10^6 time steps. After training, the RL agents are used to collect 1000 expert demonstrations for each task under varying initial conditions. To standardize the dataset, the length of each demonstration is resampled to 48 steps, which is sufficient to successfully complete both tasks.

The performance of our proposed method, PF2MP, along with other baseline models, is summarized in Table I. For the Reach task, all models achieve an average success rate of approximately 84%. However, PF2MP stands out as the only method with a collision rate below 20%, specifically 18%, which is nearly half the collision rate of the baseline FMP at 34%. For the DP model, the potential field guidance does not demonstrate significant effectiveness compared to the 2D Maze scenario, with DPPF outperforming DP by only 2%. For the Push task, which is simplified as a 2-points reach task, the overall performance of all models is less robust, achieving an average success rate of only around 50%, with collision rates hovering at approximately 30%. PF2MP shows a modest improvement in collision rate, reducing it by approximately 5%. This may be attributed to the low quality of the demonstration data collected by the RL agent, which tends to favor trajectories that closely follow obstacles. Additionally, the potential field guidance contributes a slight increase in success rate for this task, achieving a gain of about 1.7%.

D. Simulated Robotics Tasks in Joint Space

To validate the effectiveness of our proposed approach to learn policies in joint space, we designed a handwriting experiment using the Genesis simulator [47]. In this experiment, a Franka Panda robotic arm was controlled to draw letters on a planar surface while avoiding predefined restricted regions, as illustrated in Fig 5. Unlike our previous

Manuscript 951 submitted to 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received February 26, 2025.

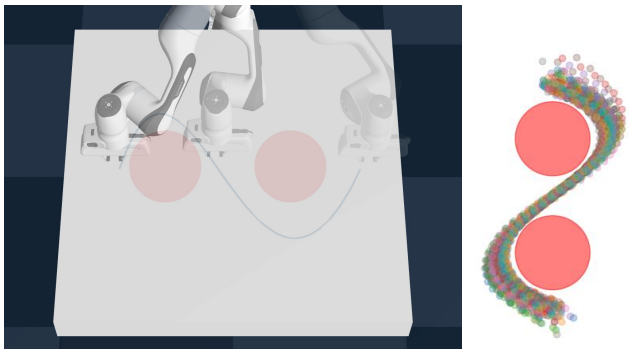


Fig. 5: Genesis Handwriting Task. Left: Simulation environment in Genesis, featuring a Franka Panda robotic arm, a white tabletop, and an "S"-shaped trajectory that avoids the red obstacle regions. Right: 2D end-effector trajectories obtained from the joint-space demonstrations.

TABLE II: Handwriting Tasks Collision Rate

Policies	**Collision Rate**	
	Genesis Simulator	Unitree Z1 Real Robot
FMP	$25.33 \pm 2.52(\%)$	$4/10$
PF2MP	$0.67 \pm 0.58(\%)$	$0/10$

experiments, this task involves direct control of the robot's joint space, encompassing 9 degrees of freedom (7 for the revolute joints, 2 for the gripper). The experimental setup highlights the adaptability of our method in generating safe and feasible trajectories under constraints, even in complex and high-dimensional task spaces.

To train the policies, we collected 20 demonstrations with the corresponding end-effector trajectories illustrated in Fig 5. The observation $\boldsymbol{o}$ in this task is the state of the robot arm in joint space. During testing, all policies were also inferred with $N = 5$ integration steps. The PF2MP weight parameter λ was set to 0.5. Every policy was tested using the same set of 100 prior samples, and this evaluation was repeated three times for statistical robustness. The results, summarized in Table II, demonstrate a clear advantage of our proposed PF2MP method in generating safer trajectories. Specifically, the baseline FMP approach exhibited a collision rate of approximately 25%, whereas our method achieved a substantial reduction in the collision rate to just 0.67%. This significant improvement demonstrates the ability of PF2MP to directly predict safe trajectories in the joint space.

E. Real Robotics Tasks

Finally, we evaluate PF2MP in a real-world scenario for a task executed on the Unitree Z1 robot arm mounted on a Unitree B2 base, as shown in Fig 6. This experiment replicates the Genesis Handwriting task but planning in task space, where the robot arm is tasked with plotting the letter "S" on a planar surface while avoiding a predefined constraint region, indicated by the pink circle in Fig 6. For this experiment, we collected 10 demonstrations with varying start and end points of the trajectory, which were generated via teleoperation using a keyboard interface provided by Unitree's official software package. We employed a smaller UNet with down dimension $[16, 32, 64]$, training it for 100 epochs. The observation $\boldsymbol{o}$ in this task is the state of robot end effector, and the prediction is the trajectory of the robot end effector.

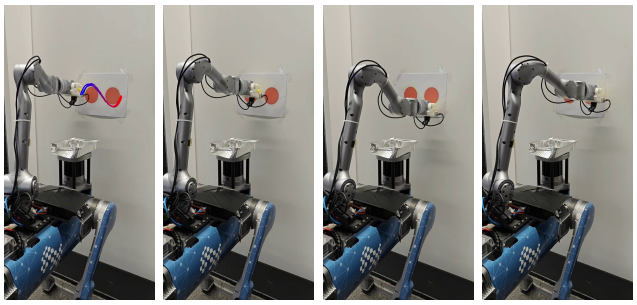


Fig. 6: Real-World Handwriting Task. A Unitree Z1 robotic arm mounted on a Unitree B2 robot dog draws the letter "S" on a wall-mounted sheet of paper, while ensuring that the trajectory avoids the predefined pink constrained area on the paper.

During testing, the hyperparameter λ for PF2MP was set to 0.5. Both policies were tested 10 times under identical initial conditions. The results, presented in Table II, clearly demonstrate the safety advantages of PF2MP. For the FMP baseline, approximately 4 out of the 10 trials resulted in trajectories that crossed the pink constraint area. In contrast, trajectories generated by PF2MP consistently avoided the constraint area in all trials, demonstrating its superior ability to generate safe trajectories under real-world conditions.

VI. CONCLUSIONS

In this work, we introduced Potential Field-Guided Flow Matching Policy (PF2MP), a novel approach that jointly learns a complex policy and an associated potential field from the same set of demonstrations. By extending the capabilities of existing FM-based imitation learning methods, PF2MP generates safer policies that significantly reduce collision rates. Extensive evaluations across diverse simulated and real-world robotics tasks demonstrated that our approach leads to a substantial reduction in collisions.

Despite its promising performance, PF2MP exhibits sensitivity to potential field weight, which requires careful tuning to balance safety and task success. Additionally, our experiments are currently restricted to static environments with stationary obstacles. Addressing these limitations, future work could focus on designing a framework that incorporates dynamic obstacle handling by learning a dynamic potential field, paving the way for broader applicability and improved robustness in real-world scenarios.

REFERENCES

[1] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[2] M. Reuss, M. Li, X. Jia, and R. Lioutikov, "Goal conditioned imitation learning using score-based diffusion policies," in *Robotics: Science and Systems (R:SS)*, 2023.

[3] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, 2024.

[4] M. Braun, N. Jaquier, L. Rozo, and T. Asfour, "Riemannian flow matching policy for robot motion learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 5144–5151.

Manuscript 951 submitted to 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received February 26, 2025.

[5] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 6840–6851, 2020.

[6] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *International Conference on Learning Representations (ICLR)*, 2021.

[7] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *International Conference on Learning Representations (ICLR)*, 2023.

[8] R. T. Chen and Y. Lipman, "Flow matching on general geometries," in *International Conference on Learning Representations (ICLR)*, 2024.

[9] J. Urain, A. Mandlekar, Y. Du, M. Shafiullah, D. Xu, K. Fragkiadaki, G. Chalvatzaki, and J. Peters, "Deep generative models in robotics: A survey on learning from multimodal demonstrations," *arXiv preprint arXiv:2408.04380*, 2024.

[10] T. Ubukata, J. Li, and K. Tei, "Diffusion model for planning: A systematic literature review," *arXiv preprint arXiv:2408.10266*, 2024.

[11] K. Muhammad, A. Ullah, J. Lloret, J. Del Ser, and V. H. C. de Albuquerque, "Deep learning for safe autonomous driving: Current challenges and future directions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4316–4336, 2020.

[12] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022.

[13] Z. Wang, T. Oba, T. Yoneda, R. Shen, M. Walter, and B. C. Stadie, "Cold diffusion on the replay buffer: Learning to plan from known good states," in *Conference on Robot Learning (CoRL)*. PMLR, 2023, pp. 3277–3291.

[14] K. Saha, V. Mandadi, J. Reddy, A. Srikanth, A. Agarwal, B. Sen, A. Singh, and M. Krishna, "EDMP: Ensemble-of-costs-guided diffusion for motion planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 10 351–10 358.

[15] 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, pp. 1916–1923.

[16] A. Duan, C. Yang, J. Zhao, S. Huo, P. Zhou, W. Ma, Y. Zheng, and D. Navarro-Alarcon, "Safe learning by constraint-aware policy optimization for robotic ultrasound imaging," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 2349–2360, 2025.

[17] Y. U. Ciftci, D. Chiu, Z. Feng, G. S. Sukhatme, and S. Bansal, "Safe-gil: Safety guided imitation learning for robotic systems," *arXiv preprint arXiv:2404.05249*, 2024.

[18] A. Duan, R. Camoriano, D. Ferigo, D. Calandriello, L. Rosasco, and D. Pucci, "Constrained dmps for feasible skill learning on humanoid robots," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, 2018, pp. 1–6.

[19] K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer, "Ensembledagger: A bayesian approach to safe imitation learning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 5041–5048.

[20] H. Yin, P. Seiler, M. Jin, and M. Arcak, "Imitation learning with stability and safety guarantees," *IEEE Control Systems Letters*, vol. 6, pp. 409–414, 2021.

[21] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications," *ACM Computing Surveys*, vol. 56, no. 4, pp. 1–39, 2023.

[22] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet, "Video diffusion models," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 8633–8646, 2022.

[23] G. Tevet, S. Raab, B. Gordon, Y. Shafir, D. Cohen-or, and A. H. Bermano, "Human motion diffusion model," in *International Conference on Learning Representations (ICLR)*, 2023.

[24] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," in *Conference on Robot Learning (CoRL)*. PMLR, 2022, pp. 1678–1690.

[25] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conference on Robot Learning (CoRL)*. PMLR, 2022, pp. 158–168.

[26] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 22 955–22 968, 2022.

[27] A. Prasad, K. Lin, J. Wu, L. Zhou, and J. Bohg, "Consistency policy: Accelerated visuomotor policies via consistency distillation," in *Robotics: Science and Systems (R:SS)*, 2024.

[28] S. Yan, Z. Zhang, M. Han, Z. Wang, Q. Xie, Z. Li, Z. Li, H. Liu, X. Wang, and S.-C. Zhu, "M2Diffuser: Diffusion-based trajectory optimization for mobile manipulation in 3D scenes," *arXiv preprint arXiv:2410.11402*, 2024.

[29] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3D diffusion policy: Generalizable visuomotor policy learning via simple 3D representations," in *Robotics: Science and Systems (R:SS)*, 2024.

[30] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," in *International Conference on Learning Representations (ICLR)*, 2023.

[31] H. Ding, N. Jaquier, J. Peters, and L. Rozo, "Fast and robust visuomotor Riemannian flow matching policy," *arXiv preprint arXiv:2412.10855*, 2024.

[32] N. Funk, J. Urain, J. Carvalho, V. Prasad, G. Chalvatzaki, and J. Peters, "ActionFlow: Equivariant, accurate, and efficient policies with spatially symmetric flow matching," *arXiv preprint arXiv:2409.04576*, 2024.

[33] Q. Rouxel, A. Ferrari, S. Ivaldi, and J.-B. Mouret, "Flow matching imitation learning for multi-support manipulation," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2024, pp. 528–535.

[34] E. Chisari, N. Heppert, M. Argus, T. Welschehold, T. Brox, and A. Valada, "Learning robotic manipulation policies from point clouds with conditional flow matching," in *Conference on Robot Learning (CoRL)*, 2024.

[35] B. W. Silverman, *Density estimation for statistics and data analysis*. Routledge, 2018.

[36] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015.

[37] Q. Zhang, Z. Liu, H. Fan, G. Liu, B. Zeng, and S. Liu, "FlowPolicy: Enabling fast and robust 3D flow-based policy via consistency flow matching for robot manipulation," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2025.

[38] K. Atkinson, *An introduction to numerical analysis*. John wiley & sons, 1991.

[39] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4RL: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[40] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, A. Perez-Vicente, Rodrigo Pierré, S. Schulhoff, J. Jet Tai, H. Tan, and O. G. Younis, "Gymnasium: A standard interface for reinforcement learning environments," *arXiv preprint arXiv:2407.17032*, 2024.

[41] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5026–5033.

[42] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder *et al.*, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," *arXiv preprint arXiv:1802.09464*, 2018.

[43] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

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

[45] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 1861–1870.

[46] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[47] Genesis Authors, "Genesis: A universal and generative physics engine for robotics and beyond," December 2024. [Online]. Available: https://github.com/Genesis-Embodied-AI/Genesis