Fast and Robust Visuomotor Riemannian Flow Matching Policy

Haoran Ding[1], Noémie Jaquier[2], Jan Peters[3], Leonel Rozo[1]

***Abstract*—Diffusion-based visuomotor policies excel at learning complex robotic tasks by effectively combining visual data with high-dimensional, multi-modal action distributions. However, diffusion models often suffer from slow inference due to costly denoising processes or require complex sequential training arising from recent distilling approaches. This paper introduces Riemannian Flow Matching Policy (RFMP), a model that inherits the easy training and fast inference capabilities of flow matching (FM). Moreover, RFMP inherently incorporates geometric constraints commonly found in realistic robotic applications, as the robot state resides on a Riemannian manifold. To enhance the robustness of RFMP, we propose Stable RFMP (SRFMP), which leverages LaSalle's invariance principle to equip the dynamics of FM with stability to the support of a target Riemannian distribution. Rigorous evaluation on ten simulated and real-world tasks show that RFMP successfully learns and synthesizes complex sensorimotor policies on Euclidean and Riemannian spaces with efficient training and inference phases, outperforming Diffusion Policies and Consistency Policies.**

***Index Terms*—Learning from demonstrations; Learning and adaptive systems; Deep learning in robotics and automation; Visuomotor policies; Riemannian flow matching**

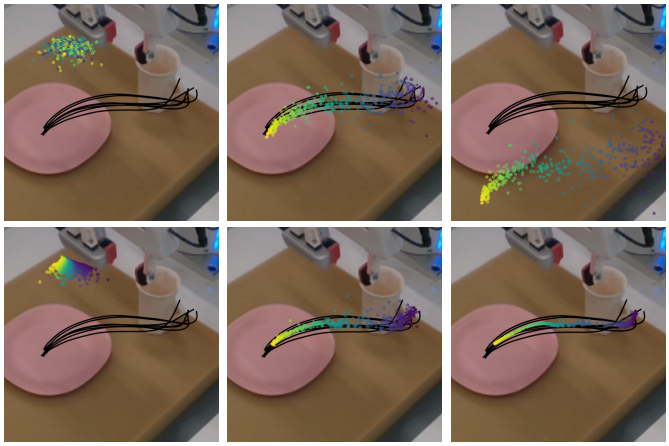


Fig. 1. Flows of the RFMP (**top**) and SRFMP (**bottom**) at times $t = \{0.0, 1.0, 1.5\}$. The policies are learned from pick-and-place demonstration (black) and conditioned on visual observations. Note that the flow of SRFMP is stable to the target distribution at $t > 1.0$, enhancing the policy robustness and inference time.

I. Introduction

DEEP generative models are revolutionizing robot skill learning due to their ability to handle high-dimensional multimodal action distributions and interface them with perception networks, enabling robots to learn sophisticated sensorimotor policies [1]. In particular, diffusion-based models such as diffusion policies (DP) [2]–[7] exhibit exceptional performance in imitation learning for a large variety of simulated and real-world robotic tasks, demonstrating a superior ability to learn multimodal action distributions compared to previous behavior cloning methods [8]–[10]. Nevertheless, these models are characterized by an expensive inference process as they often require to solve a stochastic differential equation, thus hindering their use in certain robotic settings [11], e.g., for highly reactive motion policies.

For instance, DP [2], typically based on a Denoising Diffusion Probabilistic Model (DDPM) [12], requires approximately 100 denoising steps to generate an action. This translates to roughly 1 second on a standard GPU. Even faster approaches such as Denoising Diffusion Implicit Models (DDIM) [13] still need 10 denoising steps, i.e., 0.1 second, per action [2]. Consistency policy [3] aims to accelerate the inference process by training a student model to mimic a DP teacher with larger denoising steps. Despite providing a more computationally-efficient inference, the CP training requires more computational resources and might be unstable due to the sequential training of the two models. Importantly, training these models becomes more computationally demanding when manipulating data with geometric constraints, e.g., robot end-effector orientations, as the computation of the score function of the diffusion process is not as simple as in the Euclidean case [14]. Furthermore, the inference process also incurs increasing computational complexity.

To overcome these limitations, we propose to learn visuomotor robot skills via a Riemannian flow matching policy (RFMP). Compared to DP, RFMP builds on another kind of generative model: Flow Matching (FM) [15], [16]. Intuitively, FM gradually transforms a simple prior distribution into a complex target distribution via a vector field, which is represented by a simple function. The beauty of FM lies in its simplicity, as the resulting flow, defined by an ordinary differential equation (ODE), is much easier to train and much faster to evaluate compared to the stochastic differential equations of diffusion models. However, as many visuomotor policies are represented in the robot's operational space, action representations must include both end-effector position and orientation. Thus, the policy must consider that orientations

[1]Bosch Center for Artificial Intelligence. Renningen, Germany. leonel.rozo@de.bosch.com

[2]Division of Robotics, Perception, and Learning, KTH Royal Institute of Technology, Stockholm, Sweden. jaquier@kth.se

[3] Computer Science Department of the Technische Universität Darmstadt, Darmstadt, Germany. peters@ias.tu-darmstadt.de

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

lie on either the $\mathcal{S}^3$ hypersphere or the $\mathrm{SO}(3)$ Lie group, depending on the chosen parametrization. To properly handle such data, we leverage Riemannian flow matching (RFM) [16], an extension of flow matching that accounts for the geometric constraints of data lying on Riemannian manifolds. In our previous work [17], we introduced the idea of leveraging flow matching in robot imitation learning and presented RFMP, which capitalizes on the easy training and fast inference of FM methods to learn and synthesize end-effector pose trajectories. However, our initial evaluation was limited to simple proof-of-concept experiments on the LASA dataset [18].

In this paper, we demonstrate the effectiveness of RFMP to learn complex real-world visuomotor policies and present a systematic evaluation of the performance of RFMP on both simulated and real-world manipulation tasks. Moreover, we propose Stable Riemannian Flow Matching Policy (SRFMP) to enhance the robustness of RFMP, as RFMP performance can be sensitive to the ODE integration horizon during inference (see Figures 1 and 2). SRFMP builds on stable flow matching (SFM) [19], [20], which leverages LaSalle's invariance principle [21] to equip the dynamics of FM with stability to the support of the target distribution. Unlike SFM, which is limited to Euclidean spaces, SRFMP generalizes this concept to Riemannian manifolds, guaranteeing the stability of the RFM dynamics to the support of a Riemannian target distribution. We systematically evaluate RFMP and SRFMP across 10 tasks in both simulation and real-world settings, with policies conditioned on both state and visual observations. Our experiments demonstrate that RFMP and SRFMP inherit the advantages from FM models, achieving comparable performance to DP with fewer evaluation steps (i.e., faster inference) and significantly shorter training times. Moreover, under the same training epochs, our methods outperform both DP and CP. Notably, SRFMP requires fewer ODE steps than RFMP to achieve an equivalent performance, resulting in even faster inference times.

In summary, beyond demonstrating the effectiveness of our early work on simulated and real robotic tasks, **the main contributions of this article** are threefold: *(1)* We introduce Stable Riemannian Flow Matching (SRFM) as an extension of SFM [20] to incorporate stability into RFM; *(2)* We propose stable Riemannian flow matching policy (SRFMP), which combines the easy training and fast inference of RFMP with stability guarantees to a Riemannian target action distribution; *(3)* We systematically evaluate both RFMP and SRFMP across 10 tasks from simulated benchmarks and real settings. Supplementary material is available on the paper website https://sites.google.com/view/rfmp.

II. Related work

As the literature on robot policy learning is vast, we here focus on approaches that design robot policies based on flow-based generative models.

Normalizing Flows are arguably the first broadly-used generative models in robot policy learning [22]. They were commonly employed as diffeomorphisms for learning stable dynamical systems in Euclidean spaces [23]–[25], with extensions to Lie groups [26] and Riemannian manifolds [27], similarly addressed in this paper. The main drawback of normalizing flows is their slow training, as the associated ODE needs to be integrated to calculate the model log-likelihood. Moreover, none of the foregoing works learned sensorimotor policies based on visual observations via imitation learning.

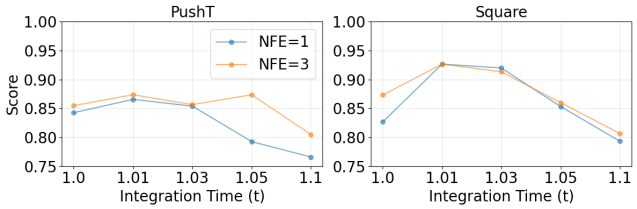


Fig. 2. RFMP performance when extending inference time beyond $t = 1$ on the Euclidean PUSHT and Robomimic SQUARE tasks.

Diffusion Models [28] recently became state-of-art in imitation learning due to their ability to learn multi-modal and high-dimensional action distributions. They have been primarily employed to learn motion planners [29], and complex control policies [2], [4], [30]. Recent extensions use 3D visual representations from sparse point clouds [6], and employ equivariant networks for learning policies that, by design, are invariant to changes in scale, rotation, and translation [7]. However, a major drawback of diffusion models is their slow inference process. In [2], DP requires 10 to 100 denoising steps, i.e., 0.1 to 1 second on a standard GPU, to generate each action. Consistency models (CM) [31] arise as a potential solution to overcome this drawback [3], [32], [33]. CM distills a student model from a pretrained diffusion model (i.e., a teacher), enabling faster inference by establishing direct connections between points along the probability path. Nevertheless, this sequential training process increases the overall complexity and time of the whole training phase.

Flow Matching [15] essentially trains a normalizing flow by regressing a conditional vector field instead of maximizing the likelihood of the model, thus avoiding to simulate the ODE of the flow. This leads to a significantly simplified training procedure compared to classical normalizing flows. Moreover, FM builds on simpler probability transfer paths than diffusion models, thus facilitating faster inference. Tong *et al.* [34] showed that several types of FM models can be obtained according to the choice of conditional vector field and source distributions, some of them leading to straighter probability paths, which ultimately result in faster inference. Rectified flows [35] is a similar simulation-free approach that designs the vector field by regressing against straight-line paths, thus speeding up inference. Note that rectified flows are a special case of FM, in which a Dirac distribution is associated to the probability path [34]. Due to their easy training and fast inference, FM models quickly became one of the de-facto generative models in machine learning and have been employed in a plethora of different applications [36]–[41].

In our previous work [17], we proposed to leverage RFM to learn sensorimotor robot policies represented by end-effector pose trajectories on Riemannian manifolds. Building on a similar idea, subsequent works have used FM along with an equivariant transformer to learn $\mathrm{SE}(3)$-equivariant

policies [42], for multi-support manipulation tasks with a humanoid robot [43], and for robot imitation learning with point cloud observations [44]. **In this paper**, we build upon our previous work, Riemannian Flow Matching Policy (RFMP) [17], to enable the learning of complex visuomotor policies on Riemannian manifolds. Unlike the aforementioned approaches, our work focuses on providing a fast and robust RFMP inference process. We achieve this by constructing the FM vector field using LaSalle's invariance principle, which not only enhances inference robustness with stability guarantees but also preserves the easy training and fast inference capabilities of RFMP.

III. Background

In this section, we provide a short background on Riemannian manifolds, and an overview of the flow matching framework with its extension to Riemannian manifolds.

A. Riemannian Manifolds

A smooth manifold $\mathcal{M}$ can be intuitively understood as a d-dimensional surface that locally, but not globally, resembles the Euclidean space $\mathbb{R}^d$ [45], [46]. The geometric structure of the manifold is described via the so-called charts, which are diffeomorphic maps between parts of $\mathcal{M}$ and $\mathbb{R}$. The collection of these charts is called an atlas. The smooth structure of $\mathcal{M}$ allows us to compute derivatives of curves on the manifolds, which are tangent vectors to $\mathcal{M}$ at a given point $\boldsymbol{x}$. For each point $\boldsymbol{x} \in \mathcal{M}$, the set of tangent vectors $\boldsymbol{u}$ of all curves that pass through $\boldsymbol{x}$ forms the tangent space $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$. The tangent space spans a d-dimensional affine subspace of $\mathbb{R}^d$, where d is the manifold dimension. The collections of all tangent spaces of $\mathcal{M}$ forms the tangent bundle $\mathcal{T}\mathcal{M} = \bigcup_{\boldsymbol{x}\in\mathcal{M}}\{(\boldsymbol{x}, \boldsymbol{u}) | \boldsymbol{u} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}\}$, which can be thought as the union of all tangent spaces paired with their corresponding points on $\mathcal{M}$.

Riemannian manifolds are smooth manifolds equipped with a smoothly-varying metric g, which is a family of inner products $g_{\boldsymbol{x}} : \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \times \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \to \mathbb{R}$. The norm associated with the metric is denoted as $\|\boldsymbol{v}\|_{g_{\boldsymbol{x}}}$ with $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$, and the distance between two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ is defined as the norm $\|\boldsymbol{u} - \boldsymbol{v}\|_{g_{\boldsymbol{x}}}$. With this metric, we can then define the length of curves on $\mathcal{M}$. The shortest curve on $\mathcal{M}$ connecting any two points $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{M}$ is called a geodesic. Intuitively, geodesics can be seen as the generalization of straight lines to Riemannian manifolds. To operate with Riemannian manifolds, a common way is to exploit its Euclidean tangent spaces $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ and back-and-forth maps between $\mathcal{M}$ and $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$, i.e., the exponential and logarithmic maps. Specifically, the exponential map $\mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{u}) : \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \to \mathcal{M}$ maps a point $\boldsymbol{u}$ on the tangent space of $\boldsymbol{x}$ to a point $\boldsymbol{y} \in \mathcal{M}$, so that the geodesic distance between $\boldsymbol{y} = \mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{u})$ and $\boldsymbol{x}$ satisfies $d_g(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{u}\|_{g_{\boldsymbol{x}}}$. The inverse operation is the logarithmic map $\mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y}) : \mathcal{M} \to \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$, which projects a point $\boldsymbol{y} \in \mathcal{M}$ to the tangent space $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ of $\boldsymbol{x}$. Finally, when optimizing functions of manifold-valued parameters, we need to compute the Riemannian gradient. Specifically, the Riemannian gradient of a scalar function $f : \mathcal{M} \to \mathbb{R}$ at $\boldsymbol{x} \in \mathcal{M}$ is a vector in the tangent space $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ [47], [48]. It is obtained via the identification $\mathcal{L}_{\boldsymbol{u}} f(\boldsymbol{x}) = \langle \nabla_{\boldsymbol{x}} f(\boldsymbol{x}), \boldsymbol{u} \rangle_{\boldsymbol{x}}$, where $\mathcal{L}_{\boldsymbol{u}} f(\boldsymbol{x})$ denotes the directional derivative of f in the direction $\boldsymbol{u} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$, and $\langle \cdot, \cdot \rangle_{\boldsymbol{x}}$ is the Riemannian inner product on $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$.

B. Flow Matching

Continuous normalizing flows (CNF) [49] form a class of deep generative models that transform a simple probability distribution into a more complex one. The continuous transformation of the samples is parametrized by an ODE, which describes the flow of the samples over time. Training CNF is achieved via maximum likelihood estimation, and thus involves solving (a.k.a. simulating) inverse ODEs, which is computationally expensive. Instead, flow matching (FM) [15] is a simulation-free generative model that efficiently trains CNF by directly mimicking a target vector field.

1) Euclidean Flow Matching: FM [15] reshapes a simple prior distribution p into a (more complicated) target distribution q via a probability density path p_t that satisfies $p_0 = p$ and $p_1 = q$. The path p_t is generated by push-forwarding p_0 along a flow ψ_t as,

$$p_t = [\psi_t]_* p_0, \tag{1}$$

where the push-forward operator $*$ is defined as,

$$[\psi_t]_* p_0(\boldsymbol{x}) = p_0(\psi_t^{-1}(\boldsymbol{x})) \det \left(\frac{\partial \psi_t^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right). \tag{2}$$

The flow ψ_t is defined via a vector field $u_t : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ by solving the ODE,

$$\frac{d\psi_t(\boldsymbol{x})}{dt} = u_t(\psi_t(\boldsymbol{x})), \quad \text{with} \quad \psi_0(\boldsymbol{x}) = \boldsymbol{x}. \tag{3}$$

Assuming that both the vector field $u_t(\boldsymbol{x})$ and probability density path p_t are known, one can regress a parametrized vector field $v_t(\boldsymbol{x}; \boldsymbol{\theta}) : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ to some target vector field u_t, which leads to the FM loss function,

$$\ell_{\text{FM}}(\boldsymbol{\theta}) = \mathbb{E}_{t, p_t(\boldsymbol{x})} \| v_t(\boldsymbol{x}; \boldsymbol{\theta}) - u_t(\boldsymbol{x}) \|_2^2, \tag{4}$$

where $\boldsymbol{\theta}$ are the learnable parameters, $t \sim \mathcal{U}[0, 1]$, and $\boldsymbol{x} \sim p_t(\boldsymbol{x})$. However, the loss (4) is intractable since we actually do not have prior knowledge about u_t and p_t. Instead, Lipman *et al.* [15] proposed to learn a conditional vector field $u_t(\boldsymbol{x}|\boldsymbol{x}_1)$ with $\boldsymbol{x}_1$ as a conditioning variable. This conditional vector field generates the conditional probability density path $p_t(\boldsymbol{x}|\boldsymbol{x}_1)$, which is related to the marginal probability path via $p_t(\boldsymbol{x}) = \int p_t(\boldsymbol{x}|\boldsymbol{x}_1) q(\boldsymbol{x}_1) d\boldsymbol{x}_1$, with q being the unknown data distribution. After reparametrization, this leads to the tractable conditional flow matching (CFM) loss function,

$$\ell_{\text{CFM}}(\boldsymbol{\theta}) = \mathbb{E}_{t, q(\boldsymbol{x}_1), p(\boldsymbol{x}_0)} \| v_t(\boldsymbol{x}_t; \boldsymbol{\theta}) - u_t(\boldsymbol{x}_t|\boldsymbol{x}_1) \|_2^2, \tag{5}$$

where $\boldsymbol{x}_t = \psi_t(\boldsymbol{x}_0|\boldsymbol{x}_1)$ denotes the conditional flow. Note that optimizing the CFM loss (5) is equivalent to optimizing the FM loss (4) as they have identical gradients [15]. The problem therefore boils down to design a conditional vector field $u_t(\boldsymbol{x}_t|\boldsymbol{x}_1)$ that generates a probability path p_t satisfying the boundary conditions $p_0 = p$, $q = p_1$. Intuitively, $u_t(\boldsymbol{x}_t|\boldsymbol{x}_1)$

should move a randomly-sampled point at $t = 0$ to a datapoint at $t = 1$. Lipman *et al.* [15] proposed the Gaussian CFM, which defines a probability path from a zero-mean normal distribution to a Gaussian distribution centered at $\boldsymbol{x}_1$ via the conditional vector field,

$$u_t(\boldsymbol{x}_t|\boldsymbol{x}_1) = \boldsymbol{x}_1 - (1-\sigma)\boldsymbol{x}_0, \tag{6}$$

which leads to the flow $\boldsymbol{x}_t = \psi_t(\boldsymbol{x}_0|\boldsymbol{x}_1) = (1-(1-\sigma)t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$. Note that a more general version of CFM is proposed by Tong *et al.* [34].

Finally, the inference process of CFM is straightforward and consists of the following steps: *(1)* Get a sample from p_0; and *(2)* Query the learned vector field $v_t(\boldsymbol{x};\boldsymbol{\theta})$ to solve the ODE (3) with off-the-shelf solvers, e.g., based on the Euler method [50].

2) Riemannian Flow Matching: In many robotics settings, data lies on Riemannian manifolds [51], [52]. For example, various tasks involve the rotation of the robot's end-effector. Therefore, the corresponding part of the state representation lies either on the Riemannian hypersphere $\mathcal{S}^3$ or the $\mathrm{SO}(3)$ group, depending on the choice of parametrization. To guarantee that the FM generative process satisfies manifold constraints, Chen and Lipman [16] extended CFM to Riemannian manifolds. The Riemannian conditional flow matching (RCFM) considers that the flow ψ_t evolves on a Riemannian manifold $\mathcal{M}$. Thus, for each point $\boldsymbol{x} \in \mathcal{M}$, the vector field associated to the flow ψ_t at this point lies on the tangent space of $\boldsymbol{x}$, i.e., $u_t(\boldsymbol{x}) \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$. The RCFM loss function resembles that of the CFM model but it is computed with respect to the Riemannian metric $g_{\boldsymbol{x}}$ as follows,

$$\ell_{\text{RCFM}} = \mathbb{E}_{t,q(\boldsymbol{x}_1),p(\boldsymbol{x}_0)}\|v_t(\boldsymbol{x}_t;\boldsymbol{\theta}) - u_t(\boldsymbol{x}_t|\boldsymbol{x}_1)\|^2_{g_{\boldsymbol{x}_t}}. \tag{7}$$

As in the Euclidean case, we need to design the flow ψ_t, its corresponding conditional vector field $u_t(\boldsymbol{x}_t|\boldsymbol{x}_1)$, and choose the base distribution. Following [16], [36], the most straightforward strategy is to exploit geodesic paths to design the flow ψ_t. For simple Riemannian manifolds such as the hypersphere, the hyperbolic manifold, and some matrix Lie groups, geodesics can be computed via closed-form solutions. We can then leverage the geodesic flow given by,

$$\boldsymbol{x}_t = \mathrm{Exp}_{\boldsymbol{x}_1}(t\,\mathrm{Log}_{\boldsymbol{x}_1}(\boldsymbol{x}_0)), \quad t \in [0,1]. \tag{8}$$

The conditional vector field can then be calculated as the time derivative of $\boldsymbol{x}_t$, i.e., $u_t(\boldsymbol{x}_t|\boldsymbol{x}_1) = \dot{\boldsymbol{x}}_t$. Notice that u_t boils down to the conditional vector field (6) with $\sigma = 0$ when $\mathcal{M} = \mathbb{R}^d$. Chen and Lipman [16] also provide a general formulation of $u_t(\boldsymbol{x}_t|\boldsymbol{x}_1)$ for cases where closed-form geodesics are not available. The prior distribution p_0 can be chosen as a uniform distribution on the manifold [16], [36], or as a Riemannian [53] or wrapped Gaussian [16], [54] distribution on $\mathcal{M}$. During inference, we solve the corresponding RCFM's ODE on the Riemannian manifold $\mathcal{M}$ via projection-based methods. Specifically, at each step, the integration is performed in the tangent space $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ and the resulting vector is projected onto the Riemannian manifold $\mathcal{M}$ with the exponential map.

C. Flow Matching vs. Diffusion and Consistency Models

Diffusion Policy (DP) [2] is primarily trained based on DDPM [12], which performs iterative denoising from an initial noise sample $\boldsymbol{x}_K$, where K denotes the total number of denoising steps. The denoising process follows,

$$\boldsymbol{x}_{k-1} = \alpha_t\big(\boldsymbol{x}_k - \gamma\epsilon_{\boldsymbol{\theta}}(\boldsymbol{x}_k, k) + \mathcal{N}(0, \sigma^2\boldsymbol{I})\big), \tag{9}$$

where α, γ and σ constitute the noise schedule that governs the denoising process, and $\epsilon_\theta(\boldsymbol{x}_k, k)$ is the noise prediction network that infers the noise at the k-th denoising step. The final sample $\boldsymbol{x}_0$ is the noise-free target output. The equivalent inference process in FM is governed by,

$$\frac{d\psi_t(\boldsymbol{x})}{dt} = v_t(\psi_t(\boldsymbol{x});\boldsymbol{\theta}), \quad \text{with} \quad \psi_0(\boldsymbol{x}) = \boldsymbol{x}, \tag{10}$$

where $v_t(\psi_t(\boldsymbol{x});\boldsymbol{\theta})$ is the learned vector field that mimics the target forward process (6). Two key differences can be identified: (1) FM requires to solve a simple ODE, and (2) The FM vector field induces straighter paths. Importantly, the DP denoising framework limits its inference efficiency, particularly in scenarios requiring faster predictions.

Consistency Policy (CP) [3] aims to speed up the inference process of DP by leveraging a consistency model that distills the DP as a teacher model. While CP adopts a similar denoising mechanism as DP, it enhances the process by incorporating both the current denoising step k and the target denoising step t as inputs to the denoising network, formalized as $\epsilon_\theta(\boldsymbol{x}_k, k, t)$. However, CP involves a two-stage training process. First, a DP teacher policy is trained. Next, a student policy is trained to mimic the denoising process of the teacher policy. This approach enables CP to achieve faster and more efficient inference while retaining the performance of its teacher model, at the cost of a more complex training process. In contrast, FM training is simulation-free, and features a single-phase training with a simple loss function.

IV. Fast and Robust RFMP

Our goal is to leverage the RCFM framework to learn a parameterized policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{o})$ that adheres to the target (expert) policy $\pi_e(\boldsymbol{a}|\boldsymbol{o})$, which generates a set of N demonstrations $D_n = \{\boldsymbol{o}^s, \boldsymbol{a}^s\}_{s=1}^T$, where $\boldsymbol{o}$ denotes an observation, $\boldsymbol{a}$ represents the corresponding action, and T denotes the length of n-th trajectory. In this section, we first introduce Riemaniann flow matching polices (RFMP) that leverage RCFM to achieve easy training and fast inference. Second, we propose Stable RFMP (SRFMP), an extension of RFMP that enhances its robustness and inference speed through stability to the target distribution.

A. Riemannian Flow Matching Policy

RFMP adapts RCFM to visuomotor robot policies by learning an observation-conditioned vector field $u_t(\boldsymbol{a}|\boldsymbol{o})$. Similar to DP [2], RFMP employs a receding horizon control strategy [55], by predicting a sequence of actions over a prediction horizon T_p. This strategy aims at providing temporal consistency and smoothness on the predicted actions. This means that the predicted action horizon vector is constructed

Algorithm 1: RFMP Training & Inference

Training

Input: Initial parameters $\boldsymbol{\theta}$, prior and target distribution p_0, p_1.

Output: Learned vector field parameters $\boldsymbol{\theta}$.

while *termination condition unsatisfied* **do**

- Sample flow time step t from the uniform distribution $\mathcal{U}[0, 1]$.
- Sample noise $\boldsymbol{a}_0 \sim p_0$.
- Jointly sample action sequence $\boldsymbol{a}_1 \sim p_1$ and corresponding observation vector $\boldsymbol{o}$.
- Compute conditional vector field $u_t(\boldsymbol{a}_t|\boldsymbol{a}_1)$ via the RCFM geodesic flow (8).
- Evaluate ℓ_{RFMP} as in (11).
- Update parameters $\boldsymbol{\theta}$.

Inference Step

Input: Predefined number of function evaluation N, learned vector field $v_{\boldsymbol{\theta}}$, observation vector $\boldsymbol{o}$, prior distribution p_0.

Sample $\boldsymbol{a}_0 \sim p_0$, and set $t = 0$.

while $t \leq 1$ **do**

- Integrate the learned Riemannian vector field $\boldsymbol{\xi}_{t+\Delta t} = \text{Exp}_{\boldsymbol{\xi}_t}(v_{\boldsymbol{\theta}}(\boldsymbol{\xi}_t, \boldsymbol{o})\Delta t)$.
- Update time $t = t + \Delta t$.

as $\boldsymbol{a} = [\boldsymbol{a}^s, \boldsymbol{a}^{s+1}, \ldots, \boldsymbol{a}^{s+T_p}]$, where $\boldsymbol{a}^i$ is the action at time step i, and T_p is the action prediction horizon. This implies that all samples $\boldsymbol{a}_1$, drawn from the target distribution p_1, have the form of the action horizon vector $\boldsymbol{a}$. Moreover, we define the base distribution p_0 such that samples $\boldsymbol{a}_{p_0} \sim p_0$ are of the form $\boldsymbol{a}_0 = [\boldsymbol{a}_b, \ldots, \boldsymbol{a}_b]$ with $\boldsymbol{a}_b$ sampled from an auxiliary distribution b. This structure contributes to the smoothness of the predicted action vector $\boldsymbol{a}$, as the flow of all its action components start from the same initial action $\boldsymbol{a}_b$.

In contrast to the action horizon vector, the observation vector $\boldsymbol{o}$ is not defined on a receding horizon but is constructed by randomly sampling only few observation vectors. Specifically, RFMP follows the sampling strategy proposed in [37] which uses: *(1)* A reference observation $\boldsymbol{o}^{s-1}$ at time step $s-1$; *(2)* A context observation $\boldsymbol{o}^c$ randomly sampled from an observation window with horizon T_o, i.e., c is uniformly sampled from $[s-T_o, s-2]$; and *(3)* The time gap $s-c$ between the context observation and reference observation. Therefore, the observation vector is defined as $\boldsymbol{o} = [\boldsymbol{o}^{s-1}, \boldsymbol{o}^c, s-c]$. Notice that, when $T_o = 2$, we disregard the time gap and the observation is $\boldsymbol{o} = [\boldsymbol{o}^{s-1}, \boldsymbol{o}^c]$. The aforementioned strategy leads to the following RFMP loss function,

$$\ell_{\text{RFMP}} = \mathbb{E}_{t,q(\boldsymbol{a}_1),p(\boldsymbol{a}_0)} \| v_t(\boldsymbol{a}_t|\boldsymbol{o};\boldsymbol{\theta}) - u_t(\boldsymbol{a}_t|\boldsymbol{a}_1) \|^2_{g_{\boldsymbol{a}_t}}. \quad (11)$$

Algorithm 1 summarizes the training process of RFMP. Note that our RFMP inherits most of the training framework of RCFM, the main difference being that the vector field learned in RFMP is conditioned on the observation vector $\boldsymbol{o}$. After training RFMP, the inference process, which essentially queries the policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{o})$, boils down to the following four steps: *(1)* Draw a sample $\boldsymbol{a}_0$ from the prior distribution p_0; *(2)* Construct the observation vector $\boldsymbol{o}$; *(3)* Employ an off-the-shelf ODE solver to integrate the learned vector field $v_t(\boldsymbol{a}|\boldsymbol{o};\boldsymbol{\theta})$ from $\boldsymbol{a}_0$ along the time interval $t = [0, 1]$, and get the generated action sequence $\boldsymbol{a} = [\boldsymbol{a}^s, ..., \boldsymbol{a}^{s+T_p}]$; and *(4)* Execute the first T_a actions of the sequence $\boldsymbol{a}$ with $T_a \leq T_p$. This last step allows the robot to quickly react to environment changes, while still providing smooth predicted actions.

Although RFM is theoretically designed to match a time-dependent vector field defined over the time horizon $t \in [0, 1]$, we observed that, in practice, its inference performance often improves when evaluating trajectories slightly beyond $t = 1$. To investigate this phenomenon, we conduct experiments on two tasks: The Euclidean PUSH-T task from [2] and the SQUARE task from the Robomimic robotic manipulation benchmark [10]. As shown in Figure 2, RFMP performance improves when the ODE integration horizon slightly exceeds $t = 1$, but it worsens and eventually gets unstable when integrating for longer time horizons. This observation motivates the development of SRFMP introduced next, which enhances RFMP by explicitly enforcing inference stability beyond the unit interval using the LaSalle's invariance principle. This approach stabilizes the policy inference, making it robust to the ODE integration horizon, as illustrated in Figures 1, 3 and 4.

B. Stable Riemannian Flow Matching Policy

Both CFM and RCFM train and integrate the learned vector field $v_t(\boldsymbol{x}|\boldsymbol{\theta})$ within the interval $t = [0, 1]$. However, they do not guarantee that the flow converges stably to the target distribution at $t = 1$. Besides, the associated vector field may even display strongly diverging behaviors when going beyond this upper boundary [20]. The aforementioned issues may arise due to numerical inaccuracies when training or integrating the vector field. To solve this problem, Sprague *et al.* [20] proposed Stable Autonomous Flow Matching (SFM), which equips the dynamics of FM with stability to the support of the target distribution. Here, we propose to improve RFMP with SFM, which we generalize to the Riemannian case, in order to guarantee that the flow stabilizes to the target policy at $t = 1$. Our experiments show that this approach not only enhances RFMP's robustness but also further reduces inference time.

1) Stable Euclidean FMP: We first summarize the Euclidean SFM [20], and then show how SFM can be integrated into RFMP. SFM leverages the stochastic LaSalle's invariance principle [56], [57] — a criterion from control theory used to characterize the asymptotic stability of stochastic autonomous dynamical systems — to design a stable vector field u. Sprague *et al.* [20, Thm 3.8] adapts this principle to the FM setting as follows.

Theorem 1. ***(Stochastic LaSalle's Invariance Principle)*** *If there exists a time-independent vector field u, a flow ψ generated by u, and a positive scalar function H such that,*

$$\mathcal{L}_{\boldsymbol{u}} H(\boldsymbol{x}) = \nabla_{\boldsymbol{x}} H(\boldsymbol{x}) u(\boldsymbol{x}) \leq 0, \quad (12)$$

where $\mathcal{L}_{\boldsymbol{u}} H(\boldsymbol{x})$ is the directional derivative of the scalar function H in the direction $\boldsymbol{u}$ and $\nabla_{\boldsymbol{x}} H$ is the gradient of

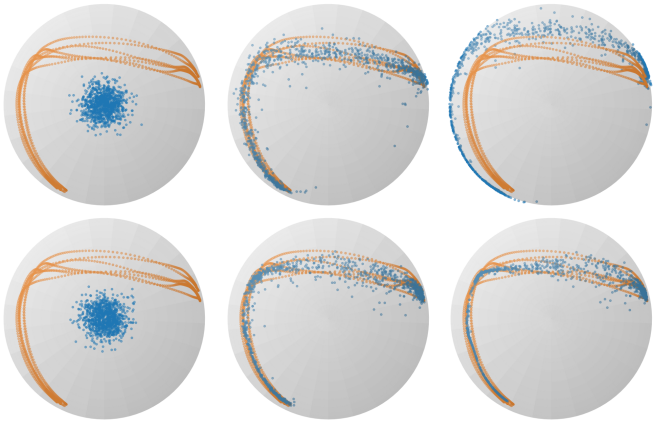


Fig. 3. Flows of the RFM (**top**) and SRFM (**bottom**) trained on the L-shape LASA dataset projected on the sphere manifold. Orange points represent the training dataset, while blue points are sampled from the generated probability path at different times $t = \{0.0, 1.0, 1.5\}$ across the three columns.

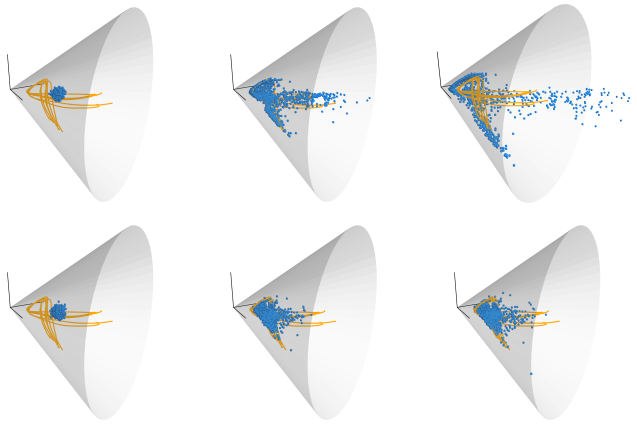


Fig. 4. Flows of the RFM (**top**) and SRFM (**bottom**) on the SPD manifold $\mathcal{S}^2_{++}$. Orange points represent the training dataset, while blue points correspond to sampled from the generated probability path at different times $t = \{0.0, 1.0, 1.5\}$ across the three columns.

the function H, then,

$$\lim_{t\to\infty} \psi(\boldsymbol{x}, t) \in \{\boldsymbol{x} \in \mathcal{X} | \mathcal{L}_u H(\boldsymbol{x}) = 0\},$$

almost surely with $\boldsymbol{x} \sim p(\boldsymbol{x}, 0)$.

Intuitively, Theorem 1 provides conditions for convergence to an invariant set even when $\mathcal{L}_u H(\boldsymbol{x})$ is not strictly negative, therefore accounting for stochastic fluctuations in the system. Theorem 1 notably holds if $u(\boldsymbol{x})$ is a gradient field of H, i.e., if $u(\boldsymbol{x}) = -\nabla_{\boldsymbol{x}} H(\boldsymbol{x})^\top$. In this case, the problem of finding a stable vector field boils down to defining an appropriate scalar function H. As LaSalle's invariance principle requires an autonomous, a.k.a time-independent, vector field, Sprague *et al.* [20] augment the FM state space $\boldsymbol{x}$ with an additional dimension τ, called temperature or pseudo time, so that the SFM augmented state space becomes $\boldsymbol{\xi} = [\boldsymbol{x}, \tau]$. The pair $(H, \boldsymbol{\xi})$ is then defined so that it satisfies (12) as,

$$H(\boldsymbol{\xi}|\boldsymbol{\xi}_1) = \frac{1}{2}(\boldsymbol{\xi} - \boldsymbol{\xi}_1)^\top \boldsymbol{A}(\boldsymbol{\xi} - \boldsymbol{\xi}_1), \tag{13}$$

$$u(\boldsymbol{\xi}|\boldsymbol{\xi}_1) = -\nabla_{\boldsymbol{\xi}} H(\boldsymbol{\xi}|\boldsymbol{\xi}_1)^\top = -\boldsymbol{A}(\boldsymbol{\xi} - \boldsymbol{\xi}_1), \tag{14}$$

where $\boldsymbol{A}$ is a positive-define matrix. To simplify the calculation, $\boldsymbol{A}$ is set as the diagonal matrix,

$$\boldsymbol{A} = \begin{bmatrix} \lambda_{\boldsymbol{x}} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \lambda_\tau \end{bmatrix}, \tag{15}$$

with $\lambda_{\boldsymbol{x}}, \lambda_\tau \in \mathbb{R}$. The vector field u and corresponding stable flow ψ_t are then given as,

$$u(\boldsymbol{\xi}_t|\boldsymbol{\xi}_1) = \begin{bmatrix} u_{\boldsymbol{x}}(\boldsymbol{x}_t|\boldsymbol{x}_1) \\ u_\tau(\tau_t|\tau_1) \end{bmatrix} = \begin{bmatrix} -\lambda_{\boldsymbol{x}}(\boldsymbol{x}_t - \boldsymbol{x}_1) \\ -\lambda_\tau(\tau_t - \tau_1) \end{bmatrix}, \tag{16}$$

$$\psi_t(\boldsymbol{\xi}_0|\boldsymbol{\xi}_1) = \begin{bmatrix} \psi_t(\boldsymbol{x}_0|\boldsymbol{x}_1) \\ \psi_t(\tau_0|\tau_1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_1 + e^{-\lambda_{\boldsymbol{x}} t}(\boldsymbol{x}_0 - \boldsymbol{x}_1) \\ \tau_1 + e^{-\lambda_\tau t}(\tau_0 - \tau_1) \end{bmatrix}. \tag{17}$$

The parameters τ_1 and τ_0 define the range of the τ flow, while the parameters $\lambda_{\boldsymbol{x}}$ and λ_τ determine its convergence. Specifically, the flow converges faster for higher values of $\lambda_{\boldsymbol{x}}$ and λ_τ. Moreover, the ratio between $\lambda_{\boldsymbol{x}}/\lambda_\tau$ determines the relative rate of convergence of the spatial and pseudo-time parts of the flow. We ablate the influence of $\lambda_{\boldsymbol{x}}$ and λ_τ on SRFMP in Section V-B.

We integrate SFM to RFMP by regressing an observation-conditioned vector field $v(\boldsymbol{\xi}|\boldsymbol{o}; \boldsymbol{\theta})$ to a stable target vector field $u(\boldsymbol{\xi}_t|\boldsymbol{\xi}_1)$, where $\boldsymbol{\xi} = [\boldsymbol{a}^s, ..., \boldsymbol{a}^{s+T_p}, \tau]$ is the augmented prediction horizon vector, and $\boldsymbol{o}$ is the observation vector.

2) Stable Riemannian FMP: Next, we introduce our extension, stable Riemannian flow matching (SRFM), which generalizes SFM [20] to Riemannian manifolds. Similarly as SFM, we define a time-invariant vector field u by augmenting the state space with the pseudo-time state τ, so that $\boldsymbol{\xi} = [\boldsymbol{x}, \tau]$. Notice that, in this case, $\boldsymbol{x} \in \mathcal{M}$ and thus $\boldsymbol{\xi}$ lies on the product of Riemannian manifolds $\mathcal{M} \times \mathbb{R}$. Importantly, Theorem 1 also holds for Riemannian autonomous systems, in which case $\nabla_{\boldsymbol{x}} H(\boldsymbol{x})$ denotes the Riemannian gradient of the positive scalar function H. We formulate H so that the pair $(H, \boldsymbol{\xi})$ satisfies (12) as,

$$H(\boldsymbol{\xi}|\boldsymbol{\xi}_1) = \frac{1}{2}\mathrm{Log}_{\boldsymbol{\xi}_1}(\boldsymbol{\xi})^\top \boldsymbol{A}\mathrm{Log}_{\boldsymbol{\xi}_1}(\boldsymbol{\xi}), \tag{18}$$

which leads to the Riemannian vector field,

$$u(\boldsymbol{\xi}|\boldsymbol{\xi}_1) = -\nabla_{\boldsymbol{\xi}} H(\boldsymbol{\xi}|\boldsymbol{\xi}_1)^\top = -\boldsymbol{A}\mathrm{Log}_{\boldsymbol{\xi}_1}(\boldsymbol{\xi}). \tag{19}$$

By setting the positive-definite matrix $\boldsymbol{A}$ as in (15), we obtain the Riemannian vector field,

$$u(\boldsymbol{\xi}_t|\boldsymbol{\xi}_1) = \begin{bmatrix} u_{\boldsymbol{x}}(\boldsymbol{x}_t|\boldsymbol{x}_1) \\ u_\tau(\tau_t|\tau_1) \end{bmatrix} = \begin{bmatrix} -\lambda_{\boldsymbol{x}}\mathrm{Log}_{\boldsymbol{x}_1}(\boldsymbol{x}_t) \\ -\lambda_\tau(\tau_t - \tau_1) \end{bmatrix}, \tag{20}$$

which generates the stable Riemannian flow,

$$\psi_t(\boldsymbol{\xi}_0|\boldsymbol{\xi}_1) = \begin{bmatrix} \psi_t(\boldsymbol{x}_0|\boldsymbol{x}_1) \\ \psi_t(\tau_0|\tau_1) \end{bmatrix} = \begin{bmatrix} \mathrm{Exp}_{\boldsymbol{x}_1}\left(e^{-\lambda_{\boldsymbol{x}} t}\mathrm{Log}_{\boldsymbol{x}_1}(\boldsymbol{x}_0)\right) \\ \tau_1 + e^{-\lambda_\tau t}(\tau_0 - \tau_1) \end{bmatrix}. \tag{21}$$

The parameters $\lambda_{\boldsymbol{x}}$ and λ_τ have the same influence as in the Euclidean case. Notice that the spatial part of the stable flow (21) closely resembles the geodesic flow (8) proposed in [16]. Figures 3 and 4 show examples of learned RFM and SRFM flows at times $t = \{0, 1, 1.5\}$ on the sphere

Algorithm 2: SRFMP Training & Inference

Training

Input: Initial parameters $\boldsymbol{\theta}$, prior and target distributions p_0, p_1.

Output: Learned vector field parameters $\boldsymbol{\theta}$.

while *termination condition unsatisfied* **do**

- Sample flow time step t from an uniform distribution $\mathcal{U}[0, 1]$.
- Sample noise $\boldsymbol{a}_0 \sim p_0$.
- Jointly sample action sequence $\boldsymbol{a}_1 \sim p_1$ and corresponding observation vector $\boldsymbol{o}$.
- Form the vectors $\boldsymbol{\xi}_1 = [\boldsymbol{a}_1, \tau_1]$ and $\boldsymbol{\xi}_0 = [\boldsymbol{a}_0, \tau_0]$.
- Compute conditional vector field $u_t(\boldsymbol{\xi}_t|\boldsymbol{\xi}_1)$ via (14).
- Evaluate the loss ℓ_{SRFMP} as defined in (22).
- Update parameters $\boldsymbol{\theta}$.

Inference Step

Input: Predefined number of function evaluation N, learned vector field $v_{\boldsymbol{\theta}}$, observation vector $\boldsymbol{o}$, prior distribution p_0.

Sample $\boldsymbol{a}_0 \sim p_0$, and set $k = 1$, $t = 0$, $\boldsymbol{\xi}_0 = [\boldsymbol{a}_0, \tau_0]$.

while $k \leq N$ **do**

- **if** $k = 1$ **then**
  - $\Delta t = \frac{1}{\lambda_{\boldsymbol{x}}}$
- **else**
  - $\Delta t = \varepsilon \leq \frac{1}{\lambda_{\boldsymbol{x}}}$
- Integrate the learned Riemannian vector field $\boldsymbol{\xi}_{t+\Delta t} = \text{Exp}_{\boldsymbol{\xi}_t}(v_{\boldsymbol{\theta}}(\boldsymbol{\xi}_t, \boldsymbol{o})\Delta t)$.
- Update time $t = t + \Delta t$.
- Update iteration $k = k + 1$.

and symmetric positive-definite (SPD) matrices manifold. The RFM flow diverges from the target distribution at time $t > 1$, while the SRFM flow is stable and adheres to the target distribution for $t \geq 1$.

Finally, the process to induce this stable behavior into the RFMP flow involves two main changes: *(1)* We define an augmented action horizon vector $\boldsymbol{\xi} = \left[\boldsymbol{a}^s, \ldots, \boldsymbol{a}^{s+T_p}, \tau\right]$, and; *(2)* We regress the observation-conditioned Riemannian vector field $v(\boldsymbol{\xi}|\boldsymbol{o}; \boldsymbol{\theta})$ against the stable Riemannian vector field $u(\boldsymbol{\xi}|\boldsymbol{\xi}_1)$ defined in (19), where $\boldsymbol{o}$ denotes the observation vector. The model is then trained to minimize the SRFMP loss,

$$\ell_{\text{SRFMP}} = \mathbb{E}_{t, q(\boldsymbol{a}_1), p(\boldsymbol{a}_0)} \| v(\boldsymbol{\xi}_t|\boldsymbol{o}; \boldsymbol{\theta}) - u(\boldsymbol{\xi}_t|\boldsymbol{\xi}_1) \|^2_{g_{\boldsymbol{a}_t}}. \quad (22)$$

This approach, hereinafter referred to as stable Riemannian flow matching policy (SRFMP), is summarized in Algorithm 2. The learned SRFMP vector field drives the flow to converge to the target distribution within a certain time horizon, ensuring that it remains within this distribution, as illustrated by the bottom row of Figures 1, 3, and 4 . In contrast, the RFMP vector field may drift the flow away from the target distribution at $t > 1$ (see the top row of Figures 1, 3, and 4). Therefore, SRFMPs provide flexibility and increased robustness in designing the generation process, while RFMPs are more sensitive to the integration process.

3) Solving the SRFMP ODE: As previously discussed, querying SRFMP policies involves integrating the learned vector field along the time interval $t = [0, T]$ with time boundary T. To do so, we use the projected Euler method, which integrates the vector field on the tangent space for one Euler step and then projects the resulting vector onto the manifold $\mathcal{M}$. Assuming an Euclidean setting and that $v(\boldsymbol{\xi}|\boldsymbol{o}; \boldsymbol{\theta})$ is perfectly learned, this corresponds to recursively applying,

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + v_{\boldsymbol{x}}(\boldsymbol{x}_t|\boldsymbol{o}; \boldsymbol{\theta})\Delta t \approx \boldsymbol{x}_t + \lambda_{\boldsymbol{x}}(\boldsymbol{x}_1 - \boldsymbol{x}_t)\Delta t, \quad (23)$$

with time step Δt and $v_{\boldsymbol{x}}$ following the same partitioning as $u_{\boldsymbol{x}}$ in (20). The time step Δt is typically set as $\Delta t = T/N$, where N is the total number of ODE steps. Here we propose to leverage the structure of SRFMP to choose the time step Δt in order to further speed up the inference time of RFMP. Specifically, we observe that the recursion (23) leads to,

$$\boldsymbol{x}_t = (1 - \lambda_{\boldsymbol{x}}\Delta t)^n(\boldsymbol{x}_0 - \boldsymbol{x}_1) + \boldsymbol{x}_1, \quad (24)$$

after n time steps. It is easy to see that $\boldsymbol{x}_t$ converges to $\boldsymbol{x}_1$ after a single time step when setting $\Delta t = 1/\lambda_{\boldsymbol{x}}$.

In the Riemannian case, assuming that $v(\boldsymbol{\xi}|\boldsymbol{o}; \boldsymbol{\theta})$ approximately equals the Riemannian vector field (20), we obtain,

$$\boldsymbol{x}_{t+1} = \text{Exp}_{\boldsymbol{x}_t}\left(v_{\boldsymbol{x}}(\boldsymbol{x}|\boldsymbol{o}; \boldsymbol{\theta})\Delta t\right) \approx \text{Exp}_{\boldsymbol{x}_t}\left(\lambda_{\boldsymbol{x}}\text{Log}_{\boldsymbol{x}_t}(\boldsymbol{x}_1)\Delta t\right). \quad (25)$$

Similarly, it is easy to see that the Riemannian flow converges to $\boldsymbol{x}_1$ after a single time step for $\Delta t = 1/\lambda_{\boldsymbol{x}}$. Importantly, this strategy assumes that the learned vector field is perfectly learned and thus equals the target vector field. However, this is often not the case in practice. However, our experiments show that the flow obtained solving the SRFMP ODE with a single time step $\Delta t = 1/\lambda_{\boldsymbol{x}}$ generally leads to the target distribution. In practice, we set Δt to $1/\lambda_{\boldsymbol{x}}$ for the first time step, and to a smaller value afterwards for refining the flow.

V. Experiments

We thoroughly evaluate the performance of RFMP and SRFMP on a set of eight simulation settings and two real-world tasks. The simulated benchmarks are: *(1)* The PUSH-T task from [2]; *(2)* A SPHERE PUSH-T task, which we introduce as a Riemannian benchmark; *(3)-(7)* Five tasks (LIFT, CAN, SQUARE, TOOL HANG, and TRANSPORT) from the large-scale robot manipulation benchmark Robomimic [10]; and *(8)* the FRANKA KITCHEN benchmark [58] featuring complex, long-horizon tasks. The real-world robot tasks correspond to: *(1)* A PICK & PLACE task; and (2) A MUG FLIPPING task. Collectively, these ten tasks serve as a benchmark to evaluate *(1)* the performance, *(2)* the training time, and *(3)* the inference time of RFMP and SRFMP with respect to state-of-the-art generative policies, i.e., DP and extensions thereof.

A. Implementation Details

To establish a consistent experimental framework, we first introduce the neural network architectures employed in RFMP and SRFMP across all tasks. We then describe the considered baselines, and our overall evaluation methodology.

TABLE I
HYPERPARAMETERS FOR ALL EXPERIMENTS: RESOLUTION OF ORIGINAL AND CROPPED IMAGE, NUMBER OF PARAMETERS FOR THE LEARNED VECTOR FIELD (VF) USING RFMP AND SRFMP, NUMBER OF RESNET PARAMETERS, TRAINING EPOCHS, AND BATCH SIZE.

Experiment	Image res.	Crop res.	RFMP VF # params	SRFMP VF # params	ResNet # params	Epochs	Batch size
PUSH-T Tasks							
Euclidean PUSH-T	96×96	84×84	8.0×10^7	8.14×10^7	1.12×10^7	300	256
SPHERE PUSH-T	100×100	84×84	8.0×10^7	8.14×10^7	1.12×10^7	300	256
State-based simulation tasks							
Robomimic LIFT	N.A.	N.A.	6.58×10^7	6.68×10^7	N.A.	50	256
Robomimic CAN	N.A.	N.A.	6.58×10^7	6.68×10^7	N.A.	50	256
Robomimic SQUARE	N.A.	N.A.	6.58×10^7	6.68×10^7	N.A.	50	512
Robomimic TOOL HANG	N.A.	N.A.	6.58×10^7	6.68×10^7	N.A.	100	512
FRANKA KITCHEN	N.A.	N.A.	6.69×10^7	6.89×10^7	N.A.	500	128
Vision-based simulation tasks							
Robomimic LIFT	$2 \times 84 \times 84$	$2 \times 76 \times 76$	9.48×10^7	9.69×10^7	$2 \times 1.12 \times 10^7$	100	256
Robomimic CAN	$2 \times 84 \times 84$	$2 \times 76 \times 76$	9.48×10^7	9.69×10^7	$2 \times 1.12 \times 10^7$	100	256
Robomimic SQUARE	$2 \times 84 \times 84$	$2 \times 76 \times 76$	9.48×10^7	9.69×10^7	$2 \times 1.12 \times 10^7$	100	512
Robomimic TRANSPORT	$4 \times 84 \times 84$	$4 \times 76 \times 76$	1.24×10^8	1.27×10^8	$4 \times 1.12 \times 10^7$	200	256
Real-world experiments							
PICK & PLACE	320×240	288×216	2.51×10^7	2.66×10^7	1.12×10^7	300	256
MUG FLIPPING	320×240	256×192	2.51×10^7	2.66×10^7	1.12×10^7	300	256

1) RFMP and SRFMP Implementation: Our RFMP implementation builds on the RFM framework from Chen and Lipman [16]. We parameterize the vector field $v_t(\boldsymbol{a}|\boldsymbol{o};\boldsymbol{\theta})$ using the UNet architecture employed in DP [2], which consists of 3 layers with downsampling dimensions of $(256, 512, 1024)$ and $(128, 256, 512)$ for simulated and real-world tasks. Each layer employs a 1-dimensional convolutional residual network as proposed in [29]. We implement a Feature-wise Linear Modulation (FiLM) [59] to incorporate the observation condition vector $\boldsymbol{o}$ and time step t into the UNet. Instead of directly feeding the FM time step t as a conditional variable, we first project it into a higher-dimensional space using a sinusoidal embedding module, similarly to DP. For tasks with image-based observations, we leverage the same vision perception backbone as in DP [2]. Namely, we use a standard ResNet-18 in which we replace: *(1)* the global average pooling with a spatial softmax pooling, and *(2)* BatchNorm with GroupNorm. Our SRFMP implementation builds on the SFM framework [20]. We implement the same UNet as RFMP to represent $v_{\boldsymbol{x}}$ by replacing the time step t by the temperature parameter τ. We introduce an additional Multi-Layer Perceptron (MLP) to learn v_τ. As for t in RFMP, we employed a sinusoidal embedding for the input τ. The boundaries τ_0 and τ_1 are set to 0 and 1 in all experiments.

We implement different prior distributions for different tasks. For the Euclidean PUSH-T and the Robomimic tasks, the action space is $\mathbb{R}^d$ and we thus define the prior distribution as a Euclidean Gaussian distribution for both RFMP and SRFMP. For the SPHERE PUSH-T, the action space is the hypersphere $\mathcal{S}^2$. In this case, we test two types of Riemannian prior distribution, namely a spherical uniform distribution, and a wrapped Gaussian distribution [54], [60], illustrated in Figure 5. For the FRANKA KITCHEN benchmark, we define the action space as the product of manifolds $\mathcal{M} = \mathbb{R}^3 \times \mathcal{S}^3 \times \mathbb{R}^2$, whose components represent the end-effector position, the end-effector orientation (encoded as quaternions), and the gripper fingers position. Regarding the real robot tasks, the action space is defined similarly as the product of manifolds $\mathcal{M} = \mathbb{R}^3 \times \mathcal{S}^3 \times \mathbb{R}^1$, whose components represent the position, orientation (encoded as quaternions), and opening of the gripper. For the FRANKA KITCHEN benchmark and the real robot tasks, the Euclidean and hypersphere parts employ Euclidean Gaussian distributions and a wrapped Gaussian distribution, respectively. Notice that this choice of prior distributions follows common practice in flow matching literature, where Gaussian and uniform priors are widely adopted due to their simplicity and effectiveness [16]. While more sophisticated approaches could be explored, e.g., using Gaussian Process as prior distributions [61], or learning task-dependent priors as in D-Flow [62], the choice depends on the problem at hand. We focus here on standard priors to maintain simplicity and consistency across tasks, while isolating their impact on our framework.

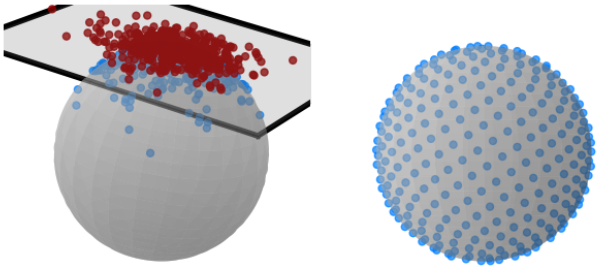


Fig. 5. 2D visualization of prior distributions on the sphere: wrapped Gaussian distribution (**left**) and sphere uniform distribution (**right**). The sphere Gaussian distribution is obtained by first sampling from a Euclidean Gaussian distribution on a tangent space of the sphere (the red points), followed by the exponential map, which projects the samples onto the sphere manifold $\mathcal{S}^d$ (blue points). The spherical uniform distribution is computed by normalizing samples from a zero-mean Euclidean Gaussian distribution.

For all experiments, we optimize the network parameters of RFMP and SRFMP using AdamW [63] with a learning rate of $\eta = 1 \times 10^{-4}$ and weight decay of $w_d = 0.001$ based on an exponential moving averaging (EMA) framework on the weights [64] with a decay of $w_{\text{EMA}} = 0.999$. We set the SRFMP parameters as $\lambda_{\boldsymbol{x}} = \lambda_\tau = 2.5$. Table I summarizes the image resolution, number of parameters, and number of training epochs used in each experiment. Note that the policies predicts a sequence of actions $\boldsymbol{a}$ over a given

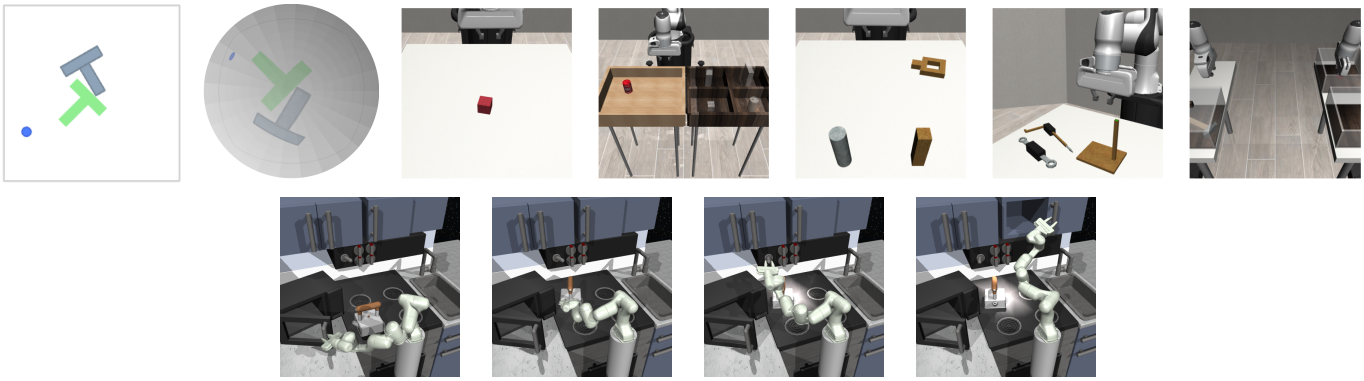


Fig. 6. Simulation benchmarks. **Top row**: Euclidean PUSH-T [2], SPHERE PUSH-T, and five Robomimic tasks [10]: LIFT, CAN, SQUARE, TOOL HANG, TRANSPORT. **Bottom row**: Tasks of the Franka kitchen benchmarks, i.e., open microwave, put kettle on top burner, switch on the light, slide cabinet.

time horizon T_p. Therefore, the total action space corresponds to the Cartesian product of manifolds computed over the prediction horizon defined for each task. Table II reports the total task dimensionality $\dim(\boldsymbol{a})$ as a function of the action space dimension $\dim(\boldsymbol{a}^s)$ and the prediction horizon T_p for all experiments considered in the paper. Notably, the FRANKA KITCHEN benchmark involves learning a policy that predicts a 1944-dimensional action sequence, enabling us to test the scalability of our method in high-dimensional, long-horizon settings. We use an action horizon $T_a = T_p/2$, and an observation horizon $T_o = 2$ for all tasks.

2) Baselines: In [2], DP is trained using either Denoising Diffusion Probabilistic Model (DDPM) [12] or Denoising Diffusion Implicit Model (DDIM) [13]. In this paper, we prioritize faster inference and thus employ DDIM-based DP for all our experiments. We train DDIM with 100 denoising steps. The prior distribution is a standard Gaussian distribution unless explicitly mentioned. During training we use the same noise scheduler as in [2], the optimizer AdamW with the same learning rate and weight decay as for RFMP and SRFMP. Note that DP does not handle data on Riemannian manifolds, and thus does not guarantee that the resulting trajectories lie on the manifold of interest for tasks with Riemannian action spaces, e.g., the SPHERE PUSH-T, the FRANKA KITCHEN benchmark, and the real-world robot experiments. In these cases, we post-process the trajectories obtained during inference and project them on the manifold. In the case of the hypersphere manifold, the projection corresponds to a unit-norm normalization.

We also compare RFMP and SRFMP against CP [3] on the Robomimic tasks with vision-based observations. For a fair comparison, we retrained CP from scratch using the code provided in [3]. We pretrain the DP teacher policy for 100 epochs and subsequently distilled the student policy for another 100 epochs, following the CP training procedure [3]. Notice that this increases the total training time, effectively doubling the number of epochs, compared to RFMP, SRFMP, and DP. The neural network architecture used in DP and CP to model the diffusion noise is identical to the one used in RFMP and SRFMP. This isolates the influence of the learning algorithm, eliminating the architectural aspects as confounding factors.

3) Evaluation methodology: We evaluate all policies using three key metrics: *(1)* The performance, computed as the average task-depending score across all trials, with 50 trials for each simulated task, and 10 trials for each real-world task; *(2)* The number of training epochs; and *(3)* The inference time. To provide a consistent measure of inference time across RFMP, SRFMP, and DP, we report it in terms of the number of function evaluations (NFE), which is proportional to the inference process time. Given that each function evaluation takes approximately the same time across all methods, inference time comparisons can be made directly based on NFE. For example, in real-world tasks, with an NFE of 2, DP requires around 0.0075s, while RFMP and SRFMP take approximately 0.0108s and 0.0113s, respectively. When NFE is increased to 5, DP takes around 0.0175s, while RFMP and SRFMP require about 0.021s and 0.0212s.

TABLE II
PREDICTION HORIZON AND ACTION SPACE DIMENSIONALITY.

Task	T_p	$\dim(\boldsymbol{a}^s)$	$\dim(\boldsymbol{a})$
Euclidean PUSH-T	16	2	32
SPHERE PUSH-T	16	2	32
Robomimic tasks	16	7	112
FRANKA KITCHEN	216	9	1944
Real-world experiments	16	7	112

B. Push-T Tasks

We first consider two simple PUSH-T tasks, namely the Euclidean PUSH-T proposed in [2], which was adapted from the Block Pushing task [8], and the SPHERE PUSH-T TASK, which we introduce shortly. The goal of the Euclidean PUSH-T task, illustrated in Figure 6, is to push a gray T-shaped object to the designated green target area with a blue circular agent. The agent's movement is constrained by a light gray square boundary. Each observation $\boldsymbol{o}$ is composed of the 96×96 RGB image of the current scene and the agent's state information. We introduce the SPHERE PUSH-T task, visualized in Figure 6, to evaluate the performance of our models on the sphere manifold. Its environment is obtained by projecting the Euclidean PUSH-T environment on one half of a 2-dimensional sphere $\mathcal{S}^2$ of radius of 1. This is achieved by projecting the environment, normalized to a range $[-1.5, 1.5]$, from the plane $z = 1$ to the sphere via a stereographic projection. The target area, the T-shaped object, and the agent

TABLE III
EUCLIDEAN PUSH-T: IMPACT OF NFE ON POLICIES.

Policy	NFE			
	1	3	5	10
RFMP	0.848	**0.855**	**0.923**	**0.891**
SRFMP	**0.875**	0.851	0.837	0.856
DP	0.109	0.79	0.838	0.862

TABLE IV
RFMP HYPERPARAMETERS ABLATION ON EUCLIDEAN PUSH-T.

Parameter	Values	Success rate				
NFE		1	3	5	10	100
T_o	**2**	**0.848**	**0.855**	**0.923**	**0.891**	**0.91**
	8	0.195	0.16	0.154	0.168	0.179
	16	0.135	0.143	0.14	0.133	0.135
T_p	8	0.754	0.835	0.827	0.839	0.85
	16	**0.848**	0.855	**0.923**	0.891	0.91
	32	0.799	**0.906**	0.878	**0.929**	**0.93**
η	$\mathbf{1 \times 10^{-4}}$	**0.848**	0.855	0.923	0.891	0.91
	5×10^{-5}	0.797	0.863	0.843	0.897	0.889
	1×10^{-5}	0.641	0.771	0.805	0.88	0.841
w_d	**0.001**	0.848	0.855	**0.923**	0.891	**0.91**
	0.005	0.846	**0.882**	0.875	0.866	0.856
	0.01	**0.868**	0.831	0.842	**0.927**	0.853

TABLE V
SRFMP HYPERPARAMETERS ABLATION ON EUCLIDEAN PUSH-T.

Parameter	Values	Success rate			
NFE		1	3	5	10
T_o	**2**	**0.875**	**0.851**	**0.837**	**0.856**
	8	0.124	0.139	0.147	0.13
	16	0.145	0.138	0.149	0.144
T_p	8	0.816	0.726	0.592	0.318
	16	**0.875**	0.851	0.837	**0.856**
	32	0.852	**0.861**	**0.881**	0.829
η	$\mathbf{1.0 \times 10^{-4}}$	**0.875**	**0.851**	**0.837**	**0.856**
	5×10^{-5}	0.754	0.621	0.456	0.334
	1×10^{-5}	0.602	0.56	0.443	0.288
w_d	**0.001**	**0.875**	**0.851**	**0.837**	**0.856**
	0.005	0.837	0.826	0.826	0.684
	0.01	0.733	0.75	0.768	0.571
$\lambda_{\boldsymbol{x}}$ λ_τ	1 0.2	0.74	0.614	0.494	0.457
	1 1	0.85	0.777	0.608	0.416
	1 6	0.87	0.768	0.753	0.812
	2.5 0.2	0.832	0.392	0.576	0.549
	2.5 2.5	**0.875**	**0.851**	0.837	**0.856**
	2.5 15	0.832	0.799	0.789	0.807
	5 1	0.796	0.741	0.614	0.513
	5 5	0.845	0.825	0.797	0.642
	5 30	0.822	0.830	0.772	0.743
	7.5 1.5	0.799	0.685	0.442	0.456
	7.5 7.5	0.787	0.8	0.809	0.633
	7.5 45	0.782	0.817	**0.844**	0.814

then lie and evolve on the sphere. As in the Euclidean case, each observation $\boldsymbol{o}$ is composed of the 96×96 RGB image of the current scene and the agent's state information on the manifold. All models (i.e., RFMP, SRFMP, DP) are trained for 300 epochs in both settings. During testing, we choose the best validation epoch and roll out 500 steps in the environment with an early stop rule terminating the execution when the coverage area is over 95% of the green target area. The score for both Euclidean and Sphere Push-T tasks is the maximum coverage ratio during execution. The tests are performed with 50 different initial states not present in the training set.

1) Euclidean Push-T: First, we evaluate the performance of RFMP and SRFMP in the Euclidean case for different number of function evaluations in the testing phase. The models are trained with the default parameters described in Section V-A1. Table III shows the success rate of RFMP and SRFMP for different NFEs. We observe that both RFMP and SRFMP achieve similar success rates overall. While SRFMP demonstrates superior performance with a single NFE, RFMP achieves higher success rates with more NFEs. We hypothesize that this behavior arises from the fact that, due to the equality $\lambda_{\boldsymbol{x}} = \lambda_\tau$, the SRFMP conditional probability path resembles the optimal transport map between the prior and target distributions as in [15]. This, along with the stability framework of SRFMP, allows us to automatically choose the time step during inference via (24), which enhances convergence in a single step. When comparing our approaches with DP, we observe that DP performs drastically worse than both RFMP and SRFMP for a single NFE, achieving a score of only 10.9%. Nevertheless, the performance of DP improves when increasing the NFE and matches that of our approaches for 10 NFE.

Next, we ablate the action prediction horizon T_p, observation horizon T_o, learning rate η, and weight decay w_d for RFMP and SRFMP. We consider 3 different values for each, while setting the other hyperparameters to their default values, and test the resulting models with 5 different NFE. For SRFMP, we additionally ablate the parameters $\lambda_{\boldsymbol{x}}$ and λ_τ for 3 different ratios $\lambda_{\boldsymbol{x}}/\lambda_\tau$ and 4 values for each ratio. Each setup is tested with 50 seeds, resulting in a total of 2250 and 4000 experiments for RFMP and SRFMP. The results are reported in Tables IV and V, respectively. We observe that a short observation horizon $T_o = 2$ leads to the best performance for both models. This is consistent with the task, as the current and previous images accurately provide the required information for the next pushing action, while the actions associated with past images rapidly become outdated. Moreover, we observe that an action prediction horizon $T_p = 16$ leads to the highest score. We hypothesize that this horizon allows the model to maintain temporal consistency, while providing frequent enough updates of the actions according to the current observations. Concerning SRFMP, we find that $\lambda_{\boldsymbol{x}} = \lambda_\tau = 2.5$ leads to the highest success rates. Interestingly, this choice leads to the ratio $\lambda_{\boldsymbol{x}}/\lambda_\tau = 1$, in which case the flow of $\boldsymbol{x}$ follows the Gaussian CFM (6) of [15] with $\sigma \to 0$ for $\tau = [0, 1]$, see [20, Cor 4.12]. In the next experiments, we use the default parameters resulting from our ablations, i.e., $T_p = 16$, $T_o = 2$, $\eta = 1 \times 10^{-4}$, $w_d = 0.001$, and $\lambda_{\boldsymbol{x}} = \lambda_\tau = 2.5$.

2) Sphere Push-T: Next, we test the ability of RFMP and SRFMP to generate motions on non-Euclidean manifolds with the SPHERE PUSH-T task. We evaluate two types of Riemannian prior distributions for RFMP and SRFMP, namely a spherical uniform distribution and wrapped Gaussian distribution (see Figure 5). We additionally consider a Euclidean Gaussian distribution for DP. Notice that the actions generated by DP are normalized in a post-processing step to ensure that they belong to the sphere. The corresponding performance are reported in Table VI. Our results indicate that the choice of prior distribution significantly impacts the performance of both RFMP and SRFMP. Specifically, we observe that RFMP and SRFMP with a uniform sphere

TABLE VI
IMPACT OF THE PRIOR DISTRIBUTION ON SPHERE PUSH-T TASK.

Policy	NFE			
	1	3	5	10
RFMP sphere uniform	**0.871**	0.746	0.77	0.817
RFMP sphere Gaussian	0.587	0.724	0.748	0.733
SRFMP sphere uniform	0.772	0.736	0.796	0.829
SRFMP sphere Gaussian	0.707	0.706	0.735	0.707
DP sphere uniform	0.274	0.261	0.235	0.197
DP sphere Gaussian	0.170	0.162	0.231	0.227
DP euclidean Gaussian	0.227	**0.796**	**0.813**	**0.885**

TABLE VII
INFLUENCE OF INTEGRATION TIME ON RFMP AND SRFMP.

Euclidean PUSH-T	$t = 1.0$	$t = 1.2$	$t = 1.6$
RFMP	0.855	0.492	0.191
SRFMP	**0.862**	**0.851**	**0.829**
Sphere PUSH-T	$t = 1.0$	$t = 1.2$	$t = 1.6$
RFMP	**0.736**	0.574	0.264
SRFMP	0.727	**0.736**	**0.685**

distribution consistently outperform their counterparts with wrapped Gaussian distribution. We hypothesize that RFMP or SRFMP benefit from having samples that are close to the data support, which leads to simpler vector fields to learn. In other words, uniform distribution provides more samples around the data distribution, which potentially lead to simpler vector fields. DP exhibit poor performance with sphere-based prior distributions, suggesting its ineffectiveness in handling such priors. Instead, DP's performance drastically improves when using a Euclidean Gaussian distribution and higher NFE. Note that this high performance does not scale to higher dimensional settings as already evident in the real-world experiments reported in Section V-E, where the effect of ignoring the geometry of the parameters exacerbates, which is a known issue when naively operating with Riemannian data [65]. Importantly, SRFMP is consistently more robust to NFE and achieves high performance with a single NFE, leading to shorter inference times for similar performance compared to RFMP and DP.

3) Influence of Integration Time Boundary: We further assess the robustness of SRFMP to varying time boundaries on the PUSH-T tasks by increasing the time boundary during inference. The performance of both RFMP and SRFMP is summarized in Table VII with result presented for NFE $= 3$ under the time boundaries $t = 1$ and $t = 1.2$, as well as for NFE $= 4$ under the time boundaries $t = 1.6$. Our results show that the performance of RFMP is highly sensitive to the time boundary, gradually declining as the boundary increases. In contrast, SRFMP demonstrates remarkable robustness, with minimal variation across different time boundaries. As illustrated in Figure 7, the quality of action series generated by RFMP noticeably deteriorates with increasing time boundaries, whereas SRFMP consistently delivers high-quality action series regardless of the time boundary.

C. Robomimic Benchmark

Next, we evaluate RFMP and SRFMP on the well-known Robomimic robotic manipulation benchmark [10]. This benchmark consists of five tasks with varying difficulty levels for which it provides two types of demonstrations, namely proficient human (PH) high-quality teleoperated demonstrations, and mixed human (MH) demonstrations. Each demonstration contains multi-modal observations, including state information, images, and depth data. We report results on five tasks (LIFT, CAN, SQUARE, TOOL HANG, and TRANSPORT) from the Robomimic dataset with 200 PH demonstrations for training for both state- and vision-based observations. Note that the difficulty of the selected tasks becomes progressively more challenging. The score of each of the 50 trials is determined by whether the task is completed successfully after a given number of steps (300 for LIFT, 500 for CAN and SQUARE, 700 for TOOL HANG, and 500 for TRANSPORT). The performance is then the percentage of successful trials.

Fig. 7. Action series generated by RFMP (**first and third rows**) and SRFMP (**second and fourth rows**) trained on the PUSH-T tasks at integration times $t = \{0.8, 1.2, 1.6\}$.

1) State-based Observations: We first assess the training efficiency of RFMP and SRFMP and compare it against DP by analyzing their performance at different training stages. Figure 8 shows the success rate of the three policies as a function of the number of training epochs for the tasks LIFT, CAN, SQUARE, and TOOL HANG. All policies are evaluated with 3 NFE for LIFT, CAN, and SQUARE, and with 10 NFE for TOOL HANG. We observe that RFMP and SRFMP consistently outperform DP across all tasks, requiring fewer training epochs to achieve comparable or superior performance. For the easier tasks (LIFT and CAN),

TABLE VIII
SUCCESS RATE AS A FUNCTION OF DIFFERENT NFE VALUES ON THE STATE-BASED ROBOMIMIC TASKS.

			LIFT		
NFE	1	2	3	5	10
RFMP	0.992 ± 0.010	$\mathbf{0.992 \pm 0.010}$	$\mathbf{0.992 \pm 0.010}$	0.992 ± 0.010	0.992 ± 0.010
SRFMP	$\mathbf{1.000 \pm 0.000}$	$\mathbf{0.992 \pm 0.010}$	$\mathbf{0.992 \pm 0.010}$	$\mathbf{0.996 \pm 0.008}$	$\mathbf{1.000 \pm 0.000}$
DP	0.008 ± 0.010	0.756 ± 0.034	0.948 ± 0.016	0.956 ± 0.015	0.976 ± 0.015
			CAN		
NFE	1	2	3	5	10
RFMP	0.976 ± 0.023	$\mathbf{0.996 \pm 0.008}$	$\mathbf{0.996 \pm 0.008}$	$\mathbf{0.996 \pm 0.008}$	$\mathbf{0.996 \pm 0.008}$
SRFMP	$\mathbf{0.980 \pm 0.022}$	0.980 ± 0.013	$\mathbf{0.996 \pm 0.008}$	0.992 ± 0.010	0.968 ± 0.020
DP	0.004 ± 0.008	0.340 ± 0.013	0.836 ± 0.020	0.924 ± 0.015	0.908 ± 0.010
			SQUARE		
NFE	1	2	3	5	10
RFMP	$\mathbf{0.792 \pm 0.027}$	$\mathbf{0.848 \pm 0.020}$	$\mathbf{0.920 \pm 0.018}$	$\mathbf{0.896 \pm 0.041}$	$\mathbf{0.912 \pm 0.016}$
SRFMP	0.776 ± 0.029	0.800 ± 0.052	0.828 ± 0.016	0.824 ± 0.008	0.848 ± 0.030
DP	0.012 ± 0.010	0.384 ± 0.023	0.628 ± 0.016	0.672 ± 0.020	0.684 ± 0.015
			TOOL HANG		
NFE	1	2	3	5	10
RFMP	0.152 ± 0.020	$\mathbf{0.316 \pm 0.023}$	$\mathbf{0.368 \pm 0.032}$	$\mathbf{0.572 \pm 0.027}$	$\mathbf{0.716 \pm 0.023}$
SRFMP	$\mathbf{0.240 \pm 0.028}$	0.224 ± 0.020	0.308 ± 0.020	0.516 ± 0.023	0.568 ± 0.035
DP	0.000 ± 0.000	0.008 ± 0.010	0.008 ± 0.010	0.092 ± 0.024	0.092 ± 0.016

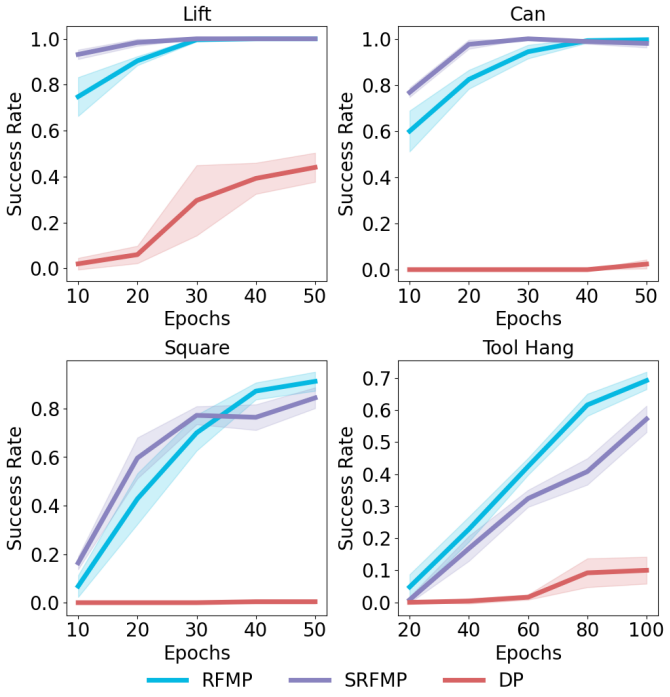


Fig. 8. Success rate (mean and standard deviation) on Robomimic tasks with state-based observations at different checkpoints. The models performance of LIFT, CAN, and SQUARE tasks is checked every 10 epochs throughout the 50-epoch training process using 3 NFE. For the TOOL HANG task, the models are trained over 100 epochs and checked every 20 epochs using 10 NFE.

both RFMP and SRFMP achieve high performance after just 20 training epochs, while the success rate of DP remains low after 50 epochs. This trend persists in the harder tasks (SQUARE and TOOL HANG), with RFMP and SRFMP reaching high success rates significantly faster than DP.

Next, we evaluate the performance of the policies for different NFE in the testing phase. For RFMP and SRFMP, we use the 50-epoch models for LIFT, CAN, and SQUARE, and the 100-epoch models for TOOL HANG. DP is further trained for a total of 300 epochs and we select the model at the best validation epoch. The results are reported in Table VIII. RFMP and SRFMP outperform DP for all tasks and all NFE, even though DP was trained for more epochs. Moreover, we observe that RFMP and SRFMP are generally more robust to low NFE than DP. They achieve 100% success rate at almost all NFE values for the easier LIFT and CAN tasks, while DP's performance drastically drops for 1 and 2 NFE. We observe a similar trend for the SQUARE task, where the performance of RFMP and SRFMP slightly improves when increasing the NFE. The performance of all models drops for TOOL HANG, which is the most complex of the four considered tasks. In this case, the performance of RFMP and SRFMP is limited for low NFE values and improves for higher NFE. DP performs poorly for all considered NFE values. Table IX reports the jerkiness as a measure of the smoothness of the trajectories generated by the different policies. We observe that RFMP and SRFMP produce arguably smoother trajectories than DP for low NFE, as indicated by the lower jerkiness values. The smoothness of the trajectories becomes comparable for higher NFE. In summary, both RFMP and SRFMP achieve high success rates and smooth action predictions with low NFE, enabling faster inference without compromising task completion.

2) Vision-based Observations: Next, we assess our models performance when the vector field is conditioned on visual observations. We consider the tasks LIFT, CAN, SQUARE, and TRANSPORT with the same policy settings and networks (see Table I)[1]. Each observation $\boldsymbol{o}^s$ corresponds to a visual embedding derived from camera images. For LIFT, CAN, and SQUARE, we use one over-the-shoulder and one in-hand camera. For TRANSPORT, which involves bimanual manipulation, we use two in-hand cameras and two over-the-shoulder cameras. We train the models for a total 100 epochs for LIFT, CAN, and SQUARE and for 200 epochs for TRANSPORT and use the best-performing checkpoint for evaluation.

The performance of different policies is reported in Table XI. RFMP and SRFMP consistently outperform DP and CP on all tasks, regardless of the NFE. As for the previous experiments, our models are remarkably robust to changes in NFE compared to DP. Importantly, SRFMP consistently outperforms RFMP for 1 and 2 NFE. Regarding CAN and SQUARE tasks, SRFMP with 1 NFE achieved performance on

[1]Note that we omit TOOL HANG due to its significant computational cost.

TABLE IX
JERKINESS OF PREDICTED ROBOT TRAJECTORIES FOR DIFFERENT NFE ON THE ROBOMIMIC TASKS WITH STATE-BASED OBSERVATIONS. ALL VALUES ARE EXPRESSED IN THOUSANDS, WHERE THE LOWER THE SMOOTHER THE PREDICTION.

			LIFT		
NFE	1	2	3	5	10
RFMP	$\mathbf{9.97 \pm 1.23}$	$\mathbf{8.80 \pm 0.19}$	9.06 ± 0.08	8.96 ± 0.08	8.49 ± 0.21
SRFMP	10.32 ± 0.41	9.90 ± 031	$\mathbf{8.00 \pm 0.14}$	9.39 ± 0.23	9.03 ± 0.19
DP	329.20 ± 47.44	13.96 ± 0.89	8.06 ± 0.42	$\mathbf{7.48 \pm 0.13}$	$\mathbf{5.9 \pm 0.23}$
			CAN		
NFE	1	2	3	5	10
RFMP	7.88 ± 0.14	$\mathbf{6.38 \pm 0.40}$	$\mathbf{6.37 \pm 0.17}$	6.40 ± 0.07	7.10 ± 1.30
SRFMP	$\mathbf{7.64 \pm 0.23}$	7.63 ± 0.18	6.69 ± 0.27	6.98 ± 0.13	6.94 ± 0.08
DP	526.00 ± 65.50	18.98 ± 2.82	6.68 ± 0.25	$\mathbf{6.28 \pm 0.35}$	$\mathbf{6.22 \pm 0.16}$
			SQUARE		
NFE	1	2	3	5	10
RFMP	$\mathbf{9.14 \pm 0.19}$	$\mathbf{6.22 \pm 0.29}$	$\mathbf{5.54 \pm 0.24}$	$\mathbf{5.17 \pm 0.16}$	$\mathbf{4.36 \pm 0.30}$
SRFMP	9.16 ± 0.31	9.43 ± 0.16	7.28 ± 0.97	7.70 ± 0.50	7.74 ± 0.26
DP	548.20 ± 54.77	20.26 ± 2.47	7.24 ± 0.54	10.02 ± 2.64	7.62 ± 1.44
			TOOL HANG		
NFE	1	2	3	5	10
RFMP	$\mathbf{4.49 \pm 0.35}$	$\mathbf{4.26 \pm 0.19}$	$\mathbf{4.65 \pm 0.32}$	$\mathbf{5.07 \pm 0.19}$	$\mathbf{4.66 \pm 0.31}$
SRFMP	5.24 ± 0.18	5.10 ± 0.20	5.23 ± 0.38	5.62 ± 0.42	5.62 ± 0.40
DP	699.40 ± 90.91	9.52 ± 0.30	8.79 ± 1.51	7.12 ± 1.94	6.96 ± 1.35

TABLE X
TRAINING TIME (IN SECONDS) PER EPOCH ON ROBOMIMIC VISION-BASED LIFT AND SQUARE.

Task	RFMP	SRFMP	DP	CP
LIFT	17.49 ± 0.03	17.63 ± 0.15	16.07 ± 0.20	31.44 ± 0.27
SQUARE	57.62 ± 0.30	56.67 ± 0.18	53.16 ± 0.16	102.86 ± 0.61

par with RFMP using 3 NFE. This efficiency gain showcases the benefits of enhancing the policies with stability to the target distribution for reducing their inference time.

In contrast to CP, RFMP and SRFMP rely on a simple, single-stage training pipeline, thus featuring easier and faster training in addition to fast inference. The training time per epoch of each policy on task LIFT and SQUARE tasks is summarized in Table X. All experiments were conducted on an NVIDIA RTX 4060Ti GPU using the same batch size across policies. The result indicate that RFMP, SRFMP, and DP have comparable per-epoch training times. In contrast, CP requires approximately twice as much time per epoch, primarily due to its training procedure involving distillation from a teacher policy. When accounting for the additional time required to pretrain DP the total training time per epoch for CP becomes roughly three times that of the other methods.

D. Franka Kitchen Benchmark

We evaluate the capability of RFMP and SRFMP in handling more complex, long-horizon manipulation tasks on the widely-used FRANKA KITCHEN benchmark [58]. We use the dataset from [66], which comprises 19 expert demonstrations totaling 4209 time steps and involves a sequential execution of four tasks: open the microwave, put the kettle on the top burner, switch on the light and slide the cabinet. The original dataset considers actions in Euclidean space as joint angular velocities. To evaluate the performance of our Riemannian policy framework, we instead consider end-effector trajectories obtained from joint configurations via forward kinematics. Therefore, each data point is composed of the end-effector position and orientation (as a unit quaternion), and the gripper state. Consequently, the predicted action sequence lies on the product of manifolds $\mathbb{R}^3 \times \mathcal{S}^3 \times \mathbb{R}^2$.

Each model is trained for 500 epochs with a batch size of 32, and evaluated using the best validation checkpoint over 50 test episodes with randomized initial states. The performance of different policies is reported in Table XII. SRFMP exhibits consistently high performance across all NFE, indicating strong robustness, and significantly outperforms RFMP and DP. The low success rate of RFMP and DP is due to the fact that their trajectories often exhibit local inaccuracies, leading to failing at least one subtask. This is then reported as a failure for the long-horizon task, leading to low success rates overall. In contrast, the trajectories produced by SRFMP are more precise, leading to the successful completion of all subtasks.

E. Real Robotic Experiments

Finally, we evaluate RFMP and SRFMP on two real-world tasks, namely PICK & PLACE and MUG FLIPPING, with a 7-DoF robotic manipulator.

1) Experimental Setup: Figures 9-11 show our experimental setup. The tasks are performed on a Franka Emika Panda robot arm. We collect the demonstrations via a teleoperation system made of two robot twins. Demonstrations are collected by an expert guiding the source robot. The target robot reads the end-effector pose of the source robot and reproduces it via a Cartesian impedance controller. The demonstration data was smoothed to mitigate the noise arising from the teleoperation setup. Each observation $\boldsymbol{o}^s$ is composed of the end-effector position and of the image embedding obtained from the ResNet vision backbone that processes the images from an over-the-shoulder camera. The policies are trained to generate 8-dimensional actions composed of the position, orientation, and gripper state lying on the product of manifolds $\mathbb{R}^3 \times \mathcal{S}^3 \times \mathbb{R}$.

2) Pick & Place: The goal of this task is to test the ability of RFMP and SRFMP to learn Euclidean policies in real-world settings. The task consists of approaching and picking up a white mug, and place it on a pink plate, as shown in Figure 10. Note that the robot end-effector points downwards during the entire task, so that its orientation remains almost constant.

TABLE XI
SUCCESS RATE AS A FUNCTION OF NFE ON VISION-BASED ROBOMIMIC TASKS.

Task	**LIFT**					**CAN**					**SQUARE**					**TRANSPORT**				
NFE	1	2	3	5	10	1	2	3	5	10	1	2	3	5	10	1	2	3	5	10
RFMP	**1**	**1**	**1**	**1**	**1**	0.78	0.82	**0.9**	**0.96**	**0.94**	0.56	0.74	**0.9**	**0.9**	**0.9**	0.6	**0.82**	0.78	**0.78**	0.8
SRFMP	**1**	**1**	**1**	**1**	**1**	**0.88**	**0.88**	**0.9**	0.9	0.86	**0.86**	**0.82**	**0.9**	0.88	**0.9**	**0.8**	**0.82**	**0.82**	**0.78**	0.78
DP	0	0.7	0.96	0.98	0.98	0	0.38	0.66	0.68	0.66	0	0.04	0.16	0.26	0.12	0	0.54	0.66	0.78	**0.82**
CP	0.5	0.44	0.46	0.46	0.4	0.82	0.82	0.84	0.84	0.82	0.32	0.52	0.54	0.48	0.52	0.4	0.36	0.44	0.38	0.36

TABLE XII
SUCCESS RATE AS A FUNCTION OF NFE ON THE FRANKA KITCHEN BENCHMARK.

Policy	**NFE**				
	1	**2**	**3**	**5**	**10**
RFMP	0.04	0.08	0.12	0.14	0.14
SRFMP	**1**	**1**	**1**	**1**	**1**
DP	0	0	0	0.02	0.02

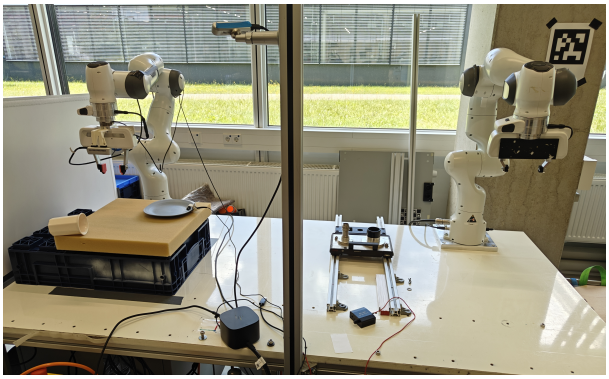


Fig. 9. Robotic experimental setup consisting of two Franka Emika Panda robot arms and an over-the-shoulder camera (Realsense d435). The left arm is the *target* robot, while the right one acts as the *source*. During the teaching phase, a human expert controls the source arm to teleoperate the target robot. During testing, only the target arm is operational.

We collect 100 demonstrations where the white mug is randomly placed on the yellow mat, while the pink plate position and end-effector initial position are slightly varied. We split our demonstration data to use 90 demonstrations for training and 10 for validation. All models are trained for 300 epochs with the same training hyperparameters as reported in Table I. As in previous experiments, we use the best-performing checkpoints of each model for evaluation.

During testing, we systematically place the white mug at 10 different locations on a semi-grid covering the surface of the yellow sponge. We evaluate the performance of RFMP, SRFMP, and DP as a function of different NFE values, under two metrics: Success rate and prediction smoothness. Figure 12 shows the increased robustness of RFMP and SRFMP to NFE compared to DP. Notably, DP requires more NFEs to achieve a success rate competitive to RFMP and SRFMP, which display high performance with only 2 NFE. Moreover, DP generated highly jerky predictions when using 2 NFE. In contrast, RFMP and SRFMP consistently retrieve smooth trajectories, regardless of the NFE.

Notice that the experiments involved natural variations in background and lighting conditions, further exposing the policies to realistic sensor noise. These slight variations in background and lighting conditions (e.g., cloudy and sunny days), had minimal impact on the policies performance. However, consistent failures were observed when the mug was initially positioned on the right side of the yellow mat, where the robot often obstructs the external camera view when approaching the mug. A multi-camera setting may improve performance on such occlusion cases.

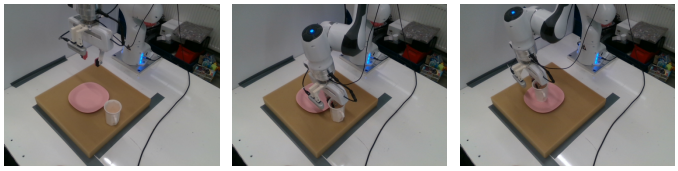


Fig. 10. PICK & PLACE: First, the robot end-effector approaches and grasps the white mug. Then, it lifts the mug and places it upright on the pink plate.

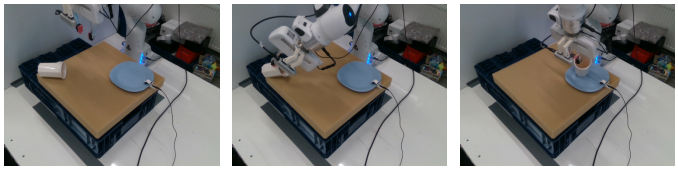


Fig. 11. MUG FLIPPING: The robot rotates its end-effector to align with the white mug's orientation and subsequently grasps it. It then place the mug upright on the blue plate.

3) Mug Flipping: In this task, a white mug is initially positioned horizontally on a yellow sponge, as shown in Figure 11. The task consists of two stages: First, the robot locates the white mug and grasps it by rotating its end-effector to align with the mug orientation. The robot then places the mug upright on the blue plate. Note that this task demands the robot to execute elaborated rotation trajectories for both grasping and placing. For this task, we collect 50 demonstrations with the white mug randomly positioned and rotated on the left side of a yellow sponge. Note that we use only the left side as the task requires the robot to operate near its workspace limits, which are prohibitive when the mug is placed on the right side. Furthermore, the end-effector initial pose was also slightly varied across the demonstrations. The policy hyperparameters for this task are provided in Table I.

Figure 12 shows the results of evaluating the different considered policies using the same two metrics as the PICK & PLACE task, namely success rate and trajectory smoothness. Both RFMP and SRFMP are significantly more robust to different NFE in terms of success rate when compared to DP. While the smoothness of RFMP and SRFMP is slightly affected by NFE in this particular task, both methods still outperform DP in this regard. Similarly to the PICK & PLACE,

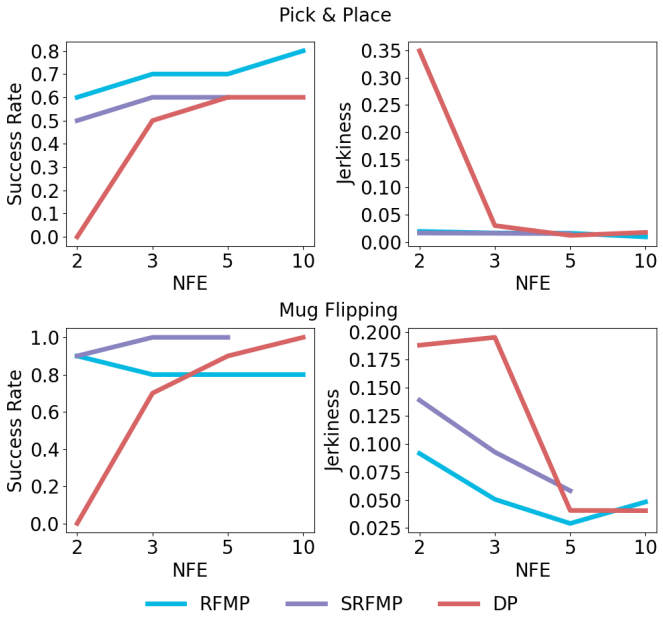


Fig. 12. Success rate and predicted actions jerkiness as a function of NFE on the PICK & PLACE and MUG FLIPPING tasks.

slight variations on lightning and background had a negligible effect on the performance of the tested policies.

F. Experiments Summary

We conducted comprehensive evaluations across diverse benchmarks in simulation, namely the Euclidean and SPHERE PUSH-T tasks, the Robomimic benchmark under both state- and vision-based observation, the long-horizon FRANKA KITCHEN benchmark, as well as two real-world manipulation tasks. These tasks display various action spaces with different types of geometric constraints. While the Euclidean PUSH-T and Robomimic tasks are defined on standard Euclidean spaces — a special type of Riemannian manifolds — the SPHERE PUSH-T task involves an action space constrained to the hypersphere $\mathcal{S}^2$, and both the FRANKA KITCHEN and real-world experiments operate on product Euclidean and sphere manifolds, accounting for position, orientation, and gripper state. Through their Riemannian formulation, RFMP and SRFMP are designed to naturally handle Euclidean and non-Euclidean action spaces, thus satisfying constraints such as the unit-norm of quaternions. It is important to emphasize that the Riemannian formulation is necessary not only to handle geometric constraints but also to guarantee stability in SRFMP. Naive approximations, e.g., unit-norm normalization as a post-processing step for quaternions, would break the stability guarantees in SRFMP.

Our findings from both simulated and real-world tasks show that RFMP and SRFMP offer significant advantages over DP. In particular, RFMP and SRFMP achieve faster inference by using fewer NFE without compromising success rates regardless of the observation type. This translates into highly-robust visuomotor policies. Importantly, these advantages do not come at the cost of elaborated training strategies like those used in consistency-based models. In fact, our models outperformed CP, while being notably easier and more efficient to train. Regarding the difference between SRFMP and RFMP, the results did not show significant performance gains in terms of success rate and prediction smoothness, at the exception of the long-horizon task of the FRANKA KITCHEN benchmark, where SRFMP significantly outperformed RFMP and DP. Nevertheless, SRFMP shows to be easier to train, achieving higher success rate than RFMP for fewer training epochs (e.g., in LIFT, CAN, SQUARE tasks).

VI. CONCLUSION

This paper introduced Stable Riemannian Flow Matching Policy (SRFMP), a novel framework that combines the easy training of flow matching with stability-based robustness properties for visuomotor policy learning. SRFMP builds on our extension of stable flow matching to Riemannian manifolds, providing stable convergence of the learned flow to the support of Riemannian target distributions. Our simulated and real-world experiments show that both our previous work on RFMP and its stable counterpart SRFMP outperform diffusion policies and distillation-based extensions, while offering advantages in terms of inference speed, ease of training, and robust performance even with limited NFEs and training epochs. Future work will focus on exploring equivariant policy structures to potentially reduce the number of required demonstrations and to improve generalization. Additionally, we aim to investigate multi-modal perception backbones for tackling contact-rich tasks.

REFERENCES

[1] 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.

[2] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Robotics: Science and Systems (R:SS)*, 2023.

[3] 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.

[4] 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.

[5] S.-F. Chen, H.-C. Wang, M.-H. Hsu, C.-M. Lai, and S.-H. Sun, "Diffusion model-augmented behavioral cloning," in *Intl. Conf. on Machine Learning (ICML)*, 2024.

[6] 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.

[7] J. Yang, Z. Cao, C. Deng, R. Antonova, S. Song, and J. Bohg, "Equibot: SIM(3)-equivariant diffusion policy for generalizable and data efficient learning," in *Conference on Robot Learning (CoRL)*, 2024.

[8] 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)*, 2022.

[9] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," in *Neural Information Processing Systems (NeurIPS)*, 2022.

[10] 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)*, 2022.

[11] C. Luo, "Understanding diffusion models: A unified perspective," *arXiv preprint arXiv2208.11970*, 2022.

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

[13] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *Intl. Conf. on Learning Representations (ICLR)*, 2020.

[14] C.-W. Huang, M. Aghajohari, J. Bose, P. Panangaden, and A. Courville, "Riemannian diffusion models," in *Neural Information Processing Systems (NeurIPS)*, 2022.

[15] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *Intl. Conf. on Learning Representations (ICLR)*, 2022.

[16] R. T. Chen and Y. Lipman, "Riemannian flow matching on general geometries," in *Intl. Conf. on Learning Representations (ICLR)*, 2023.

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

[18] A. Lemme, Y. Meirovitch, M. Khansari-Zadeh, T. Flash, A. Billard, and J. J. Steil, "Open-source benchmarking for learned reaching motion generation in robotics," *Paladyn, Journal of Behavioral Robotics*, vol. 6, no. 1, 2015.

[19] C. I. Sprague, A. Elofsson, and H. Azizpour, "Stable autonomous flow matching," *arXiv preprint arXiv2402.05774*, 2024.

[20] ——, "Incorporating stability into flow matching," in *ICML Workshop on Structured Probabilistic Inference & Generative Modeling*, 2024.

[21] J. LaSalle, "Some extensions of Liapunov's second method," *IRE Transactions on Circuit Theory*, vol. 7, no. 4, pp. 520–527, 1960.

[22] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *Journal of Machine Learning Research*, vol. 22, no. 57, pp. 1–64, 2021.

[23] M. A. Rana, A. Li, D. Fox, B. Boots, F. Ramos, and N. Ratliff, "Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems," in *Learning for Dynamics and Control (L4DC)*. PMLR, 2020, pp. 630–639.

[24] S. A. Khader, H. Yin, P. Falco, and D. Kragic, "Learning stable normalizing-flow control for robotic manipulation," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2021, pp. 1644–1650.

[25] J. Urain, M. Ginesi, D. Tateo, and J. Peters, "Imitationflow: Learning deep stable stochastic dynamic systems by normalizing flows," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2020, pp. 5231–5237.

[26] J. Urain, D. Tateo, and J. Peters, "Learning stable vector fields on lie groups," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 569–12 576, 2022.

[27] J. Zhang, H. Beik Mohammadi, and L. Rozo, "Learning Riemannian stable dynamical systems via diffeomorphisms," in *Conference on Robot Learning (CoRL)*, 2022.

[28] 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 Comput. Surv.*, vol. 56, no. 4, 2023.

[29] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *Intl. Conf. on Machine Learning (ICML)*, 2022.

[30] Z. Wang, J. J. Hunt, and M. Zhou, "Diffusion policies as an expressive policy class for offline reinforcement learning," in *Intl. Conf. on Learning Representations (ICLR)*, 2023.

[31] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency models," in *Intl. Conf. on Machine Learning (ICML)*, 2023.

[32] G. Lu, Z. Gao, T. Chen, W. Dai, Z. Wang, and Y. Tang, "Manicm: Real-time 3d diffusion policy via consistency model for robotic manipulation," *arXiv preprint arXiv:2406.01586*, 2024.

[33] Z. Wang, Z. Li, A. Mandlekar, Z. Xu, J. Fan, Y. Narang, L. Fan, Y. Zhu, Y. Balaji, M. Zhou *et al.*, "One-step diffusion policy: Fast visuomotor policies via diffusion distillation," in *Intl. Conf. on Learning Representations (ICLR)*, 2025.

[34] A. Tong, K. Fatras, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, G. Wolf, and Y. Bengio, "Improving and generalizing flow-based generative models with minibatch optimal transport," *Trans. on Machine Learning Research (TMLR)*, 2024.

[35] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," in *Intl. Conf. on Learning Representations (ICLR)*, 2022.

[36] A. J. Bose, T. Akhound-Sadegh, K. Fatras, G. Huguet, J. Rector-Brooks, C.-H. Liu, A. C. Nica, M. Korablyov, M. Bronstein, and A. Tong, "SE(3)-stochastic flow matching for protein backbone generation," in *Intl. Conf. on Learning Representations (ICLR)*, 2023.

[37] A. Davtyan, S. Sameni, and P. Favaro, "Efficient video prediction via sparsely conditioned flow matching," in *Intl. Conf. on Computer Vision (ICCV)*, 2023, pp. 23 263–23 274.

[38] V. T. Hu, W. Yin, P. Ma, Y. Chen, B. Fernando, Y. M. Asano, E. Gavves, P. Mettes, B. Ommer, and C. G. M. Snoek, "Motion flow matching for human motion synthesis and editing," *arXiv preprint arXiv:2312.08895*, 2023.

[39] X. Zhang, Y. Pu, Y. Kawamura, A. Loza, Y. Bengio, D. L. Shung, and A. Tong, "Trajectory flow matching with applications to clinical time series modeling," in *Neural Information Processing Systems (NeurIPS)*, 2024.

[40] H. Lin, O. Zhang, H. Zhao, D. Jiang, L. Wu, Z. Liu, Y. Huang, and S. Z. Li, "PPFlow: Target-aware peptide design with torsional flow matching," in *Intl. Conf. on Machine Learning (ICML)*, 2024.

[41] A. H. Liu, M. Le, A. Vyas, B. Shi, A. Tjandra, and W.-N. Hsu, "Generative pre-training for speech with flow matching," in *Intl. Conf. on Learning Representations (ICLR)*, 2024.

[42] N. Funk, J. Urain, J. Carvalho, V. Prasad, G. Chalvatzaki, and J. Peters, "Actionflow: Efficient, accurate, and fast policies with spatially symmetric flow matching," in *R:SS workshop: Structural Priors as Inductive Biases for Learning Robot Dynamics*, 2024.

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

[44] 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.

[45] M. do Carmo, *Riemannian Geometry*. Birkhäuser Basel, 1992.

[46] J. M. Lee, *Introduction to Riemannian manifolds*. Springer, 2018, vol. 2.

[47] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2007.

[48] N. Boumal, *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023.

[49] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Neural Information Processing Systems (NeurIPS)*, 2018.

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

[51] J. M. Selig, *Geometric fundamentals of robotics*. Springer Science & Business Media, 2007.

[52] F. Merat, "Introduction to robotics: Mechanics and control," *IEEE Journal on Robotics and Automation*, vol. 3, no. 2, pp. 166–166, 1987.

[53] X. Pennec, "Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements," *Journal of Mathematical Imaging and Vision*, vol. 25, pp. 127–154, 2006.

[54] F. Galaz-Garcia, M. Papamichalis, K. Turnbull, S. Lunagomez, and E. Airoldi, "Wrapped distributions on homogeneous Riemannian manifolds," *arXiv preprint arXiv:2204.09790*, 2022.

[55] D. Q. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," in *IEEE Conference on Decision and Control (CDC)*, 1988, pp. 464–465.

[56] J. P. La Salle, "An invariance principle in the theory of stability," Tech. Rep., 1966.

[57] X. Mao, "Stochastic versions of the LaSalle theorem," *Journal of differential equations*, vol. 153, no. 1, pp. 175–195, 1999.

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

[59] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *AAAI Conf. on Artificial Intelligence*, vol. 32, no. 1, 2018.

[60] K. V. Mardia and P. E. Jupp, *Distributions on Spheres*. John Wiley and Sons, Ltd, 1999, ch. 9, pp. 159–192.

[61] M. Kollovieh, M. Lienen, D. Lüdke, L. Schwinn, and S. Günnemann, "Flow matching with Gaussian process priors for probabilistic time series forecasting," in *Intl. Conf. on Learning Representations (ICLR)*, 2025.

[62] H. Ben-Hamu, O. Puny, I. Gat, B. Karrer, U. Singer, and Y. Lipman, "D-flow: Differentiating through flows for controlled generation," in *Intl. Conf. on Machine Learning (ICML)*, 2024.

[63] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Intl. Conf. on Learning Representations (ICLR)*, 2019.

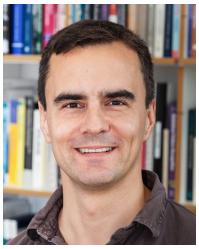
[64] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM journal on control and optimization*, vol. 30, no. 4, pp. 838–855, 1992.

[65] N. Jaquier, L. Rozo, and T. Asfour, "Unraveling the single tangent space fallacy: An analysis and clarification for applying Riemannian geometry in robot learning," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2024, pp. 242–249.

[66] O. G. Younis, R. Perez-Vicente, J. U. Balis, W. Dudley, A. Davey, and J. K. Terry, "Minari," Sep. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.13767625

Haoran Ding received the B.S. degree in Engineering from Tongji University, Shanghai, China, in 2021, and the M.Sc. degree in Computational Engineering from the Technical University of Darmstadt, Germany, in 2024. During his master's studies, he worked on the robot air hockey project at the Intelligent Autonomous Systems (IAS) Group and completed a one-year research internship at the Bosch Center for Artificial Intelligence (BCAI), where he conducted his master thesis. He is currently pursuing a Ph.D. degree in the Robotics Department at Mohamed bin Zayed University of Artificial Intelligence (MBZUAI), Abu Dhabi, United Arab Emirates. His research interests include visuomotor policy learning, generative models and imitation learning.

Noémie Jaquier is an assistant professor at the KTH Royal Institute of Technology, where she heads the Geometric Robot (GeoRob) Lab at the Division of Robotics, Perception and Learning. She received a B.Sc. degree on Microengineering, a M.Sc. degree in Robotics and Autonomous Systems from the Ecole Polytechnique Fédérale de Lausanne (EPFL) in 2014 and 2016. She received her PhD degree from EPFL in 2020 for her thesis "Robot Skills Learning with Riemannian Manifolds: Leveraging Geometry-awareness in Robot Learning, Optimization and Control". Prior to joining KTH, Noémie Jaquier was a postdoctoral researcher in the High Performance Humanoid Technologies Lab (H²T) at the Karlsruhe Institute of Technology (KIT) and a visiting postdoctoral scholar at the Stanford Robotics Lab. Her research investigates data-efficient and theoretically-sound learning algorithms that leverage differential geometry- and physics-based inductive bias to endow robots with close-to-human learning and adaptation capabilities. Personal webpage: http://njaquier.ch.



Jan Peters is a full professor (W3) for Intelligent Autonomous Systems at the Computer Science Department of the Technische Universitaet Darmstadt since 2011, and, at the same time, he is the dept head of the research department on Systems AI for Robot Learning (SAIROL) at the German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) since 2022. He is also is a founding research faculty member of the Hessian Center for Artificial Intelligence. Jan Peters has received the Dick Volz Best 2007 US PhD Thesis Runner-Up Award, the Robotics: Science & Systems - Early Career Spotlight, the INNS Young Investigator Award, and the IEEE Robotics & Automation Society's Early Career Award as well as numerous best paper awards. In 2015, he received an ERC Starting Grant and in 2019, he was appointed IEEE Fellow, in 2020 ELLIS fellow and in 2021 AAIA fellow. Jan Peters has studied Computer Science, Electrical, Mechanical and Control Engineering at TU Munich and FernUni Hagen in Germany, at the National University of Singapore (NUS) and the University of Southern California (USC). He has received four Master's degrees in these disciplines as well as a Computer Science PhD from USC. Jan Peters has performed research in Germany at DLR, TU Munich and the Max Planck Institute for Biological Cybernetics (in addition to the institutions above), in Japan at the Advanced Telecommunication Research Center (ATR), at USC and at both NUS and Siemens Advanced Engineering in Singapore. He has led research groups on Machine Learning for Robotics at the Max Planck Institutes for Biological Cybernetics (2007-2010) and Intelligent Systems (2010-2021).

Leonel Rozo is a lead research scientist at the Bosch Center for Artificial Intelligence (BCAI), Germany. He received a Bachelor degree in Mechatronics Engineering from the "Nueva Granada" Military University in Bogotá, Colombia in 2005, and a Master degree in Automatic Control and Robotics from the Polytechnical University of Catalonia, Barcelona, Spain, in 2007. He carried out his PhD research at the Institut de Robòtica i Informàtica Industrial under the supervision of Prof. Carme Torras and Dr. Pablo Jiménez, and received a Ph.D in Robotics from the Polytechnical University of Catalonia in 2013. Prior to joining BCAI in 2018, Leonel Rozo led the Learning and Interaction Group at the department of Advanced Robotics in the Italian Institute of Technology (IIT) from 2016 to 2018. He previously joined IIT in 2012, first as a research fellow and then as postdoctoral researcher in 2013. In 2017 he was awarded an individual Marie Skłodowska-Curie fellowship for his project proposal DRAPer. His research has been mainly focused on exploiting machine learning techniques, optimal control, and Riemannian manifold theory to learn robot motion skills via human demonstrations and refinement methods such as reinforcement learning with applications to (dual-arm) manipulation tasks and human-robot collaboration. Personal webpage: https://leonelrozo.weebly.com.