2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Riemannian Flow Matching Policy for Robot Motion Learning

Max Braun[1], Noémie Jaquier[1], Leonel Rozo[2], and Tamim Asfour[1]

***Abstract*— We introduce Riemannian Flow Matching Policies (RFMP), a novel model for learning and synthesizing robot visuomotor policies. RFMP leverages the efficient training and inference capabilities of flow matching methods. By design, RFMP inherits the strengths of flow matching: the ability to encode high-dimensional multimodal distributions, commonly encountered in robotic tasks, and a very simple and fast inference process. We demonstrate the applicability of RFMP to both state-based and vision-conditioned robot motion policies. Notably, as the robot state resides on a Riemannian manifold, RFMP inherently incorporates geometric awareness, which is crucial for realistic robotic tasks. To evaluate RFMP, we conduct two proof-of-concept experiments, comparing its performance against Diffusion Policies. Although both approaches successfully learn the considered tasks, our results show that RFMP provides smoother action trajectories with significantly lower inference times.**

I. INTRODUCTION

The problem of learning, synthesizing and adapting robot motions in unstructured environments has been recently disrupted by the rise of deep generative models. These models enable a robot to learn elaborated skills that may display high-dimensional multimodal action distributions. They can also be interfaced with deep multimodal perception networks, thus allowing a robot to learn sensorimotor policies. These models have been leveraged in both imitation and reinforcement settings [1]–[4], where diffusion processes [5], [6] have recently shown promising results in a plethora of real robotic tasks. Nevertheless, this type of models are characterized by expensive inference methods as they often require to solve a stochastic differential equation, which might hinder their use in some robotic settings [5], e.g., for reactive motion policies. Moreover, when learning diffusion models on Riemannian manifolds, the computation of the score function of the diffusion process is not as simple as in the Euclidean case [7], and the inference process incurs increasing computational complexity.

In contrast to diffusion models, flow matching (FM) [8] takes a different approach. Intuitively, FM defines a series of small transformations (flows) that can smoothly move samples from a base distribution towards the target data points (i.e., the demonstration dataset). Each flow is represented by simple function that takes a base distribution sample and pushes it slightly in a specific direction. By chaining these small flows together, FM gradually transforms the prior distribution into the target (demonstrations) distribution. The beauty of FM lies in its simplicity, as these flow function is much easier to train and evaluate compared to solving complex stochastic differential equations as in diffusion models. Motivated by the recent efficacy of flow matching (FM) methods [8] across various machine learning domains [9]–[11], we propose to learn sensorimotor robot skills via a Riemannian Flow Matching Policy (RFMP). RFMP capitalizes on the easy training and fast inference of FM methods to learn and synthesize robot movements represented by end-effector pose trajectories. Our main contributions are twofold: (1) we pioneer the application of FM methods within sensorimotor robot policies learning, and (2) we empirically validate their effectiveness on the established benchmark LASA dataset [12].

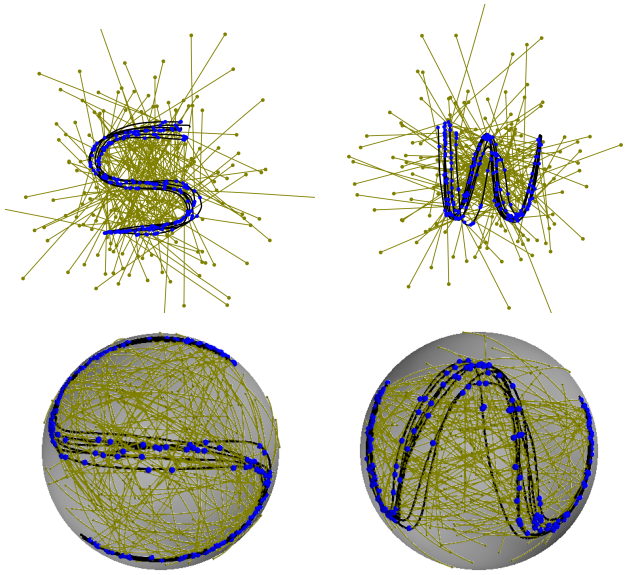


Fig. 1: Learned RFMP flows (—) from the base distribution (•) to the LASA datasets S and W (•) on both $\mathbb{R}^2$ (top) and $\mathcal{S}^2$ (bottom). The flow is conditioned on random observations $\boldsymbol{o}$ from the training dataset (—).

Related work: The literature on robot policy learning is vast, and therefore we focus on approaches that design policies based on flow-based generative models. Normalizing flows [13] are arguably the first models to be broadly adapted as robot policy representations. The most common approach was to employ them as diffeomorphisms for learning stable dynamical systems [14]–[16], with extensions to Lie groups [17], and Riemannian manifolds [18]. A shortcoming of normalizing flows is their slow training due to the integration of the associated ODE. More recently, diffusion

This work was supported by the Carl Zeiss Foundation under the project JuBot and the European Union's Horizon Europe Framework Programme under grant agreement No 101070596 (euROBIN).

[1]Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Karlsruhe, Germany. noemie.jaquier@kit.edu

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

models have dominated the robot learning scene due to their more stable training and their ability to learn complex data distributions more accurately [6]. They have been primarily employed to learn motion planners [1] and complex control policies [2]–[4]. In contrast to the aforementioned works, our work leverages flow matching [8] to model robot motion policies. This choice stems from the inherent advantages of FM: it avoids the complex training procedures of normalizing flows and the computationally expensive inference of diffusion models. Furthermore, our method also accounts for full-pose trajectories by leveraging the recently developed Riemannian extension of FM models presented in [19].

II. BACKGROUND

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

A. Riemannian manifolds

Let us imagine a flexible and curved surface like a globe. A smooth manifold, mathematically denoted by $\mathcal{M}$, can be intuitively conceptualized as a smaller patch on that surface. Locally, this patch looks flat, therefore resembling the Euclidean space $\mathbb{R}^d$ [20], [21]. But unlike the entire globe, the whole manifold may be curved or twisted, preventing it from being entirely flat. The smoothness of the manifold allows us to define directions and rates of change at each point, leading to tangent vectors in $\mathbb{R}^d$. The set of tangent vectors of all curves at $\boldsymbol{x} \in \mathcal{M}$ forms a d-dimensional affine subspace of $\mathbb{R}^d$, known as the *tangent space* $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ of $\mathcal{M}$ at $\boldsymbol{x}$. The collection of all such tangent spaces is called the *tangent bundle* $\mathcal{TM} = \bigcup_{\boldsymbol{x}\in\mathcal{M}} \{(\boldsymbol{x}, \boldsymbol{u}) | \boldsymbol{u} \in \mathcal{T}_x\mathcal{M}\}$ It is possible to endow a smooth manifold $\mathcal{M}$ with a *Riemannian metric*, which is a family of inner products $g_{\boldsymbol{x}} : \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \times \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \to \mathbb{R}$ associated to each point $\boldsymbol{x} \in \mathcal{M}$. A *Riemannian manifold* $(\mathcal{M}, g)$ is a smooth manifold endowed with a Riemannian metric g, that is a family of inner products $g_{\boldsymbol{x}} : \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \times \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \to \mathbb{R}$ associated to each point $\boldsymbol{x} \in \mathcal{M}$ [21].

To operate with Riemannian manifolds, we can leverage their Euclidean tangent spaces and resort to mappings back and forth between $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ and $\mathcal{M}$, using 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 $\boldsymbol{u} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ to a point $\boldsymbol{y}$ on the manifold, so that it lies on the geodesic starting at $\boldsymbol{x}$ in the direction $\boldsymbol{u}$ and such that the geodesic distance $d_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{y}) = d_{\mathbb{R}}(\boldsymbol{x}, \boldsymbol{u})$. The inverse operation is the logarithmic map $\mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y}) : \mathcal{M} \to \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$. Finally, the parallel transport $\Gamma_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{u}) : \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \to \mathcal{T}_{\boldsymbol{y}}\mathcal{M}$ describes how elements of $\mathcal{M}$ can be transported along curves on $\mathcal{M}$ while maintaining their intrinsic geometric properties. This allows us to operate elements lying on different tangent spaces.

B. Flow Matching

Flow matching [8] is a simulation-free generative model that reshapes a simple base density $p_0 \in \mathbb{P}(\mathbb{R}^d)$ to a target (more complicated) distribution $p_1 \in \mathbb{P}(\mathbb{R}^d)$ via the push-forward of the prior $p_1 = \phi_\sharp p_0$, with ϕ denoting the *flow* and $\sharp$ being the push-forward operator. To design this flow, we can define a vector field $u_t : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ that represents the ODE,

$$\frac{d\phi_t(\boldsymbol{x})}{dt} = u_t(\phi_t(\boldsymbol{x})) \quad \text{with initial condition} \quad \phi_0(\boldsymbol{x}) = \boldsymbol{x}. \tag{1}$$

Loosely speaking, the vector field u_t defines how a sample $\boldsymbol{x}_0 \sim p_0$ is transformed over time (from t_0 to $t = 1$) to match a target sample from $\boldsymbol{x}_1 \sim p_1$. At a density level, the vector field defines a probability density path $p_t : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$, i.e. an interpolation in probability space, which is characterized by the continuity equation [8].

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

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

where $\boldsymbol{\theta}$ are the learnable parameters, $t \sim \mathcal{U}[0, 1]$, and $\boldsymbol{x} \sim p_t(\boldsymbol{x})$. Unfortunately, the objective in (2) is intractable since we actually do not have prior knowledge about p_t and u_t. Lipman *et al.* [8] showed that by defining a conditional probability path $p_t(\boldsymbol{x}|\boldsymbol{z})$ (and consequently a conditional vector field $u_t(\boldsymbol{x}|\boldsymbol{z})$, it is possible to obtain a tractable conditional flow matching (CFM) loss,

$$\mathcal{L}_{\text{CFM}}(\boldsymbol{\theta}) = \mathbb{E}_{t,q(\boldsymbol{z}),p_t(\boldsymbol{x}|\boldsymbol{z})} \|v_t(\boldsymbol{x}; \boldsymbol{\theta}) - u_t(\boldsymbol{x}|\boldsymbol{z})\|_2^2, \tag{3}$$

which happens to have identical gradients to the unconditional loss (2) w.r.t $\boldsymbol{\theta}$.

Tong *et al.* [22] showed that there exist several forms of CFM depending on how we design the prior $q(\boldsymbol{z})$, the conditional probability $p_t(\boldsymbol{x}|\boldsymbol{z})$, and the corresponding vector field $u(\boldsymbol{x}|\boldsymbol{z})$. For example, by considering the conditioning variable as $\boldsymbol{z} = \boldsymbol{x}_1 \sim p_1$, and by defining $p_t(\boldsymbol{x}|\boldsymbol{z})$ and $u(\boldsymbol{x}|\boldsymbol{z})$ as follows,

$$p_t(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{x}|t\boldsymbol{x}_1, (t\sigma - t + 1)^2\right), \tag{4}$$

$$u_t(\boldsymbol{x}|\boldsymbol{z}) = \frac{\boldsymbol{x}_1 - (1 - \sigma)\boldsymbol{x}}{1 - (1 - \sigma)t}, \tag{5}$$

one recovers the Gaussian CFM of Lipman *et al.* [8], which defines a probability path from a zero-mean normal distribution to a Gaussian distribution centered at $\boldsymbol{x}_1$, which is also the approach taken in this paper. Finally, the inference process boils down to: (1) Get a sample from p_0 and; (2) Use the vector field $v_t(\boldsymbol{x}; \boldsymbol{\theta})$ to solve the ODE (1) using off-the-shelf solvers.

The Riemannian case: In several robotic settings, the target data distribution may lie on a Riemannian manifold $\mathcal{M}$, as the desired movements for a robot's end-effector often include the orientation component. Therefore, the part of the robot state representation lie on either the $\mathcal{S}^3$ hypersphere or the $\mathrm{SO}(3)$ group, depending on the specific parametrization used. To properly handle this type of cases, Chen and Lipman [19] recently extended CFM to Riemannian manifolds (RCFM). Formally, this Riemannian formulation considers that $\boldsymbol{x} \in \mathcal{M}$, and therefore the vector field $u_t(\boldsymbol{x}) \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$

(i.e., it evolves on the tangent bundle $\mathcal{TM}$) generates a probability density path $p_t \in \mathbb{P}(\mathcal{M})$. As stated previously, a Riemannian manifold $\mathcal{M}$ is endowed with a Riemannian metric $g_{\boldsymbol{x}}$, which implies that the CFM loss (3) is now computed w.r.t such a metric, as follows,

$$\mathcal{L}_{\text{RCFM}}(\boldsymbol{\theta}) = \mathbb{E}_{t,q(\boldsymbol{z}),p_t(\boldsymbol{x}|\boldsymbol{z})} \|v_t(\boldsymbol{x};\boldsymbol{\theta}) - u_t(\boldsymbol{x}|\boldsymbol{z})\|^2_{g_{\boldsymbol{x}}}. \quad (6)$$

As in the Euclidean case, we need to design the vector field and choose the base distribution. Following [11], [19], the most straightforward strategy is to exploit geodesic paths to design the flow ϕ, i.e., we use the shortest path to connect $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$. Importantly, for many known geometries such as the $\mathcal{S}^d$ hypersphere, the $\mathrm{SO}(3)$ group, or the manifold of symmetric positive definite matrices $\mathcal{S}^d_{++}$, to mention a few, we have closed-form geodesics. Specifically, the geodesic flow connecting $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$ is given by,

$$\boldsymbol{x}_t = \mathrm{Exp}_{\boldsymbol{x}_0}\left(t \,\mathrm{Log}_{\boldsymbol{x}_0}(\boldsymbol{x}_1)\right), \quad t \in [0,1]. \quad (7)$$

We now can design the vector field $u_t(\boldsymbol{x}|\boldsymbol{z})$ by leveraging the ODE associated with the conditional flow $d\phi_t(\boldsymbol{x})/dt = \dot{\boldsymbol{x}}_t = u_t(\boldsymbol{x}|\boldsymbol{z})$. This means that computing the vector field $u_t(\boldsymbol{x}|\boldsymbol{z})$ corresponds to compute the time derivative of (7). Finally, the choice of the base distribution p_0 generally depends on the problem at hand. One could directly define the base density as a uniform distribution over $\mathcal{M}$ as in [11], [19], but it is also possible to use Riemannian or wrapped Gaussian distributions.

III. The Riemannian Flow Matching Policy

Given a set of trajectories $\mathcal{D} = \{\boldsymbol{o}_n, \boldsymbol{a}_n\}_{n=1}^N$, where $\boldsymbol{o}$ denotes the observation and $\boldsymbol{a}$ represents the corresponding action, our goal is to leverage the CFM framework to learn a Riemannian flow matching policy (RFMP) $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{o})$. This policy aims to generate action sequences that adhere to the target (expert) distribution π_e. Note that, in the general case, we assume that both $\boldsymbol{o}, \boldsymbol{a} \in \mathcal{M}$. We hereinafter explain how we leverage CFM to model, train, and use such a policy.

A. RFMP training

Firstly, we adapt RCFM to visuomotor policies by simply conditioning the parametrized vector field on the observation vector $\boldsymbol{o}_t$, that is $v_t(\boldsymbol{a}|\boldsymbol{o})$. Secondly, inspired by the diffusion policies framework [4], we employ a receding horizon to achieve temporal consistency and smoothness on the predicted actions. This means that our predicted action horizon vector is constructed as $\boldsymbol{a} = [\boldsymbol{a}_\tau, \boldsymbol{a}_{\tau+1}, \ldots, \boldsymbol{a}_{\tau+T_a}]$ for a T_a-steps prediction horizon. This implies that all samples $\boldsymbol{a}_1$ drawn from the target distribution have the form of the action horizon vector $\boldsymbol{a}$. Moreover, the samples $\boldsymbol{a}_0$ from the base distribution are constructed as $\boldsymbol{a}_0 = [\boldsymbol{a}_{p_0}, \ldots, \boldsymbol{a}_{p_0}]$ with $\boldsymbol{a}_{p_0} \sim p_0$. Nevertheless, instead of defining a similar receding horizon for the observations, we randomly sample only T_o observation vectors from the training dataset to construct the conditioning variable $\boldsymbol{o}$. To do so, we follow the sampling strategy proposed in [9], which uses: (1) A *reference observation* $\boldsymbol{o}_{\tau-1}$; (2) A *context observation* $\boldsymbol{o}_c$ with the index c uniformly sampled from $\{1, \ldots, \tau-2\}$; and (3) The distance $\tau - c$ between the prediction and the context observation. The combination of a reference and a context observation overcomes the fact that a single observation carries very little information and provides additional information about the direction of the motion. Therefore, the observation vector is defined as $\boldsymbol{o} = [\boldsymbol{o}_{\tau-1}, \boldsymbol{o}_c, \tau - c]$. The aforementioned strategy leads to the following RFMP loss,

$$\mathcal{L}_{\text{RFMP}}(\boldsymbol{\theta}) = \mathbb{E}_{t,q(\boldsymbol{a}_1),p_t(\boldsymbol{a}|\boldsymbol{a}_1)} \|v_t(\boldsymbol{a}|\boldsymbol{o};\boldsymbol{\theta}) - u_t(\boldsymbol{a}|\boldsymbol{a}_1)\|^2_{g_{\boldsymbol{a}}}. \quad (8)$$

Algorithm 8 summarizes the training procedure of RFMP.

Algorithm 1: Riemannian Flow Matching Policy

Input : Initial parameters $\boldsymbol{\theta}$, base and target distributions $q(\boldsymbol{a}_1), p(\boldsymbol{a}_1)$.
Output: Regressed vector field parameters $\boldsymbol{\theta}$.

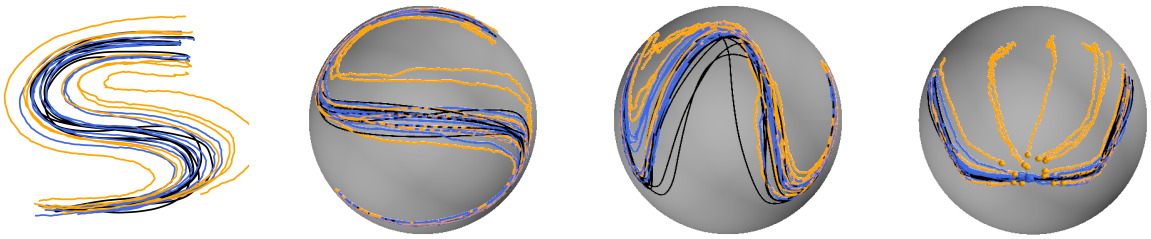
```
while termination condition do
    Sample time step t ~ U.
    Sample training example a_1 ~ p, and noise a_0 ~ q.
    Sample observation vector o.
    Compute target vector field ȧ_t = u_t(a|a_1) based on the
      geodesic flow (7).
    Evaluate L_RFMP(θ) (8).
    Update parameters θ.
end
```

B. RFMP inference

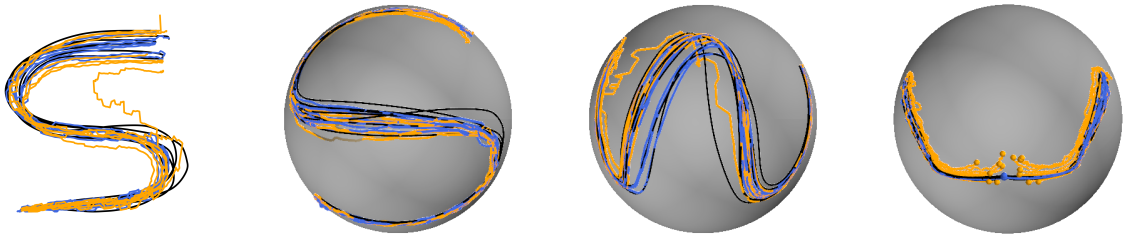
Once our RFMP is trained, the inference process, which corresponds to querying our policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{o})$, is carried out as follows: (1) Draw a sample $\boldsymbol{a}_0 \sim q$; (2) Employ an off-the-shelf ODE solver to integrate the learned vector field $v_t(\boldsymbol{a}|\boldsymbol{o};\boldsymbol{\theta})$ along the time interval $[0,1]$; (3) Execute only the first T_e actions $[\boldsymbol{a}_\tau, \boldsymbol{a}_{\tau+1}, \ldots, \boldsymbol{a}_{\tau+T_e}]$ with $T_e < T_a$, from the whole predicted action horizon $\boldsymbol{a}$. Note that the ODE solver queries the learned vector field $v_t(\boldsymbol{a}|\boldsymbol{o};\boldsymbol{\theta})$ using the observation vector $\boldsymbol{o} = [\boldsymbol{o}_{\tau-1}, \boldsymbol{o}_c, \tau - c]$ with $c \sim \mathcal{U}\{1, \ldots, \tau-2\}$. In the Euclidean case, we use the DOPRI ODE solver [23] implemented in torchdyn [24]. In the Riemannian case, we employ a Riemannian ODE solver based on the Euler method, as in [19].

C. RFMP implementation

Our RFMP implementation builds on the RFM framework from Chen and Lipman [19]. Specifically, we parameterized the vector field $v_t(\boldsymbol{a}|\boldsymbol{o};\boldsymbol{\theta})$ using a standard multilayer perceptron (MLP) with 64 hidden units and 5 layers for all experiments reported in the sequel. We used the Swish activation function [25] with a learnable parameter. The input to the MLP network is a vector formed as the concatenation of time and the observation vector. Under the aforementioned configuration, the resulting model has a total of 32K learnable parameters. We optimized the network parameters $\boldsymbol{\theta}$ using Adam with a learning rate of $1e{-}4$ and an exponential moving averaging on the weights [26] with a decay of 0.999. For all experiments, we split the data as 80% train, 10% validation, and 10% test. We trained the network for 200 epochs and selected the best model based on its performance on the validation set.



(a) RFMP trajectories.



(b) DP trajectories.

Fig. 2: Demonstrations (—) and learned trajectories on the LASA datasets S in $\mathbb{R}^2$ (*left*), on the LASA datasets S, W projected on $\mathcal{S}^2$ (*middle-left, middle-right*), and on a multimodal dataset made of mirrored datasets of the letter L projected on $\mathcal{S}^2$ (*right*). Reproductions start at the same initial observations as the demonstrations (—), or from randomly-sampled observations in the demonstration dataset neighborhood (—). Trajectory starts are depicted by dots in the multimodal case.

Concerning the base distribution, our RFMP uses a Gaussian distribution $p_0 = \mathcal{N}(\mathbf{0}, \sigma \boldsymbol{I})$, in the Euclidean case $\mathcal{M} = \mathbb{R}^2$, where we set $\sigma = 1$ for the experiments reported next. In the Riemannian setting, i.e., $\mathcal{M} = \mathcal{S}^2$, we define the base distribution as a wrapped Gaussian distribution [27], [28] centered at the manifold origin $\boldsymbol{e} = (0, \dots, 0, 1)^\mathsf{T} \in \mathcal{S}^d$, i.e., $p_0 = \mathcal{N}_{\mathcal{S}^2}\left(\boldsymbol{e}, \sigma \boldsymbol{I}\right)$ with $\sigma = 0.5$ for our experiments.

IV. EXPERIMENTS

We evaluate RFMP on the LASA dataset [12] in the Euclidean space $\mathbb{R}^2$ and on the same dataset but projected on the sphere $\mathcal{S}^2$. Using these datasets, we consider: (1) Trajectory-based policies, where observations are defined as current and past states along the trajectories; and (2) Visuomotor policies, where observations correspond to vector features extracted from grayscale images.

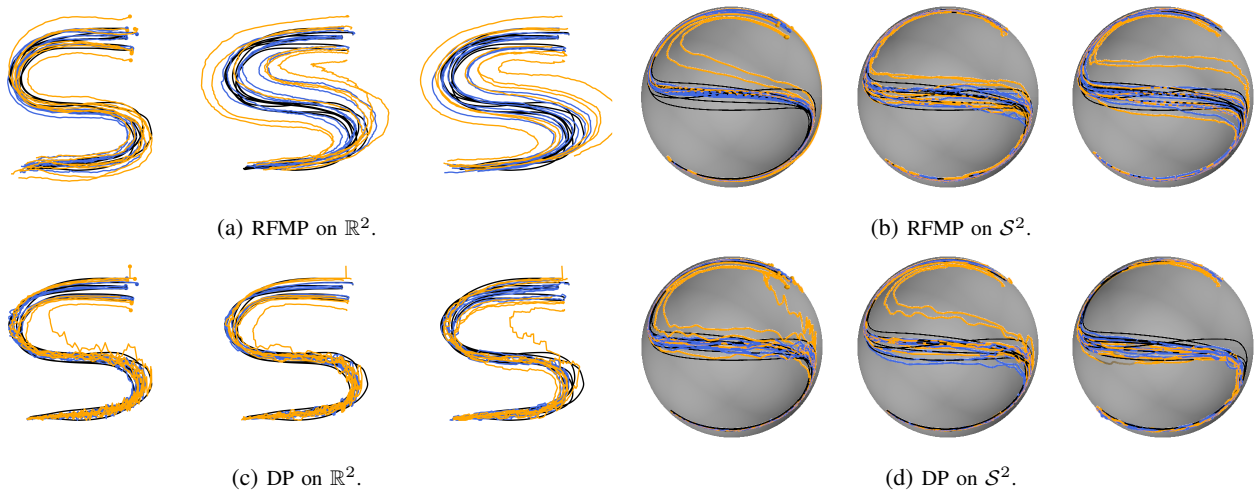
A. Trajectory-based policies

To learn the vector field v_t, we use a dataset $\{\{\{\boldsymbol{a}_{m,\tau}, \boldsymbol{o}_{m,\tau}\}_{c=1}^{\tau-2}\}_{\tau=2}^{T_m}\}_{m=1}^{M}$ of $M = 7$ demonstrations containing $T_m = 200$ timesteps each, where $\boldsymbol{a}_{m,\tau} = [\boldsymbol{a}_{m,\tau}, \dots, \boldsymbol{a}_{m,\tau+T_a}]$ and $\boldsymbol{o}_{m,\tau} = [\boldsymbol{o}_{m,\tau-1}, \boldsymbol{o}_{m,c}, \tau - c]$ are the action and observation vectors of the τ-th step of the m-th demonstration. All actions and observations are normalized and projected onto the manifold $\mathcal{M}$ of interest. In this experiment, both actions $\boldsymbol{a}_\tau \in \mathcal{M}$ and observations $\boldsymbol{o}_\tau \in \mathcal{M}$ are represented in the position space of the manifold $\mathcal{M}$. The variance of the base distribution p_0 is selected such that the distribution roughly spans half of the sphere containing the data. We use a prediction horizon $T_a = 8$ and an execution horizon $T_e = T_a/2$.

Figure 1 shows the learned RFMP flows conditioned on observations from the training dataset on the manifolds $\mathbb{R}^2$ and $\mathcal{S}^2$, for the letters S and W. We observe that the distributions reconstructed by RFMP closely match the original demonstrations. Figure 2a displays the trajectories reconstructed by sequentially executing the actions inferred by RFMP. The obtained trajectories closely follow the demonstrations when initialized with the same initial observations. Notably, when tested on initial conditions that are randomly-sampled on the neighborhood of the demonstrations support, RFMP generates trajectories that closely follow the demonstrations pattern. This kind of generalization is desired, for example, when the task demands to reproduce trajectories that closely resemble the demonstration style.

We compare RFMP against diffusion policies (DP) [4]. To do so, we employ the CNN-based diffusion network with 256M parameters provided by the authors. As in [4], we used the iDDPM algorithm [29] with the same 100 denoising diffusion iterations for both training and inference. Moreover, we used the same prediction and execution horizons as for RFMP. Figure 2b shows the trajectories obtained by sequentially executing the actions inferred by DP. Similarly to RFMP, the trajectories closely match the demonstrations when initialized with the same initial observations (blue curves). Interestingly, unlike RFMP, the trajectories starting at randomly-sampled initial conditions close to the demonstrations (orange curves), tend to rejoin the demonstration data support, resulting in less variance across reproductions. This behavior might be partly explained by a high memorization of the training data [30], although this requires further investigation.

Importantly, the trajectories obtained by DP tend to be more jerky than those obtained with RFMP. We hypothesize that such jerky trajectories are a result of the inherent stochasticity of diffusion models during inference. These



(a) RFMP on $\mathbb{R}^2$.

(b) RFMP on $\mathcal{S}^2$.

(c) DP on $\mathbb{R}^2$.

(d) DP on $\mathcal{S}^2$.

Fig. 3: Demonstrations (—) and learned trajectories on the LASA datasets S and W with different prediction horizons $T_a = \{2, 4, 8\}$ (from left to right). Reproductions start at the same initial observations as the demonstrations (—), or from randomly-sampled observations in their neighborhood (—).

observations are supported by quantitative measures. Table I shows the dynamic time warping distance (DTWD) as a measure of reproduction accuracy for trajectories initialized with the same initial observations as the demonstrations, and the jerkiness as a measure of the trajectories smoothness [31]. We observe that DP produces trajectories that display a similar or lower DTWD than RFMP. This can be explained by the tendency of DP to generate trajectories within the demonstrations support, while RFMP displays an increased variance across reproductions. As observed qualitatively, RFMP produces arguably smoother trajectories than DP, as indicated by the lower jerkiness values reported in Table I.

Note that diffusion policies are not adapted to handle data on Riemannian manifolds, and thus do not provide any guarantees that the resulting trajectories lie on the manifold of interest. This can be observed, e.g., for the S dataset on $\mathcal{S}^2$, where some trajectories enter the sphere in the middle part of the S trajectories. This means that a post-processing step would be required to ensure that the trajectories lie on the manifold of interest. Although possible, such post-processing steps are known to produce highly-inaccurate predictions as they disregard the intrinsic geometry of the data, as discussed in [32]. A more technically-sound solution would involve to adapt diffusion policies using Riemannian formulations of diffusion models [7], [33].

We also tested the capabilities of both RFMP and DP to learn multimodal policies on Riemannian manifolds. The rightmost plots in Fig. 2 show the resulting trajectories for initial conditions matching those of the demonstrations dataset (blue curves) and for initial points drawn from a region close to the demonstrations (orange curves). Although both RFMP and DP are able to learn the multimodal pattern, their generalization behavior is different when tested on initial conditions that are different from the training dataset. We can again observe that the DP trajectories tend to move back to the data support. Interestingly, in the multimodal case, RFMP outperforms DP in terms of both DTWD and smoothness (see Table I).

Next, we ablate the prediction horizon T_a for both RFMP and DP. Figure 3 shows the trajectories obtained for both policy types with $T_a = \{2, 4, 8\}$ and $T_e = T_a/2$. The corresponding quantitative measures (DTWD and jerkiness) are given in Table II. Interestingly, the RFMP trajectories remain smooth despite the reduction of the prediction horizon, even for $T_a = 2$. In contrast, DP exhibits jerkier behaviors with shorter prediction horizons. This trend is especially pronounced for trajectories starting from initial observations that are different from the training dataset (see orange curves in Fig. 3b-*left* and Fig. 3d-*left*). We hypothesize that the reduction of the prediction horizon has a stronger impact on the inference process of DP than on RFMP due to the inherent stochasticity of diffusion models.

Finally, we compare the inference time of RFMP and DP in Table III. All computations were performed on a laptop with 2.60GHz $\times$12 CPU, a Nvidia Quatro T200 GPU, and 31 GB RAM. In the Euclidean case, we observe a reduction of $\sim 30\%$ ($\sim$ 350ms) for the inference time of RFMP compared to DP. This is due to the fact that RFMP employs ODE solvers, which are generally much more efficient that the SDE solvers required for diffusion models. This result is an inherent strength of flow matching compared to diffusion models, as thoroughly analyzed in the flow matching literature [8], [22]. The inference time of RFMP on the sphere $\mathcal{S}^2$ exceeds that of DP. However, this is due to the fact that RFMP intrinsically handles data on Riemannian manifolds and therefore uses ODE solvers that are specific to this type of spaces. Instead, DP disregards the geometry of the data and thus employs Euclidean SDE solvers. A fair comparison of the inference times of RFMP and DP on the sphere would involve the adaptation of diffusion policies to data on Riemannian manifolds and leveraging Riemannian SDE solvers for inference.

	DTWD				Jerkiness			
Dataset	S, $\mathbb{R}^2$	S, $\mathcal{S}^2$	W, $\mathcal{S}^2$	multi-L, $\mathcal{S}^2$	S, $\mathbb{R}^2$	S, $\mathcal{S}^2$	W, $\mathcal{S}^2$	multi-L, $\mathcal{S}^2$
RFMP	1.87 ± 0.94	0.95 ± 0.32	1.64 ± 0.84	$\mathbf{6.14 \pm 6.56}$	$\mathbf{2120 \pm 273}$	$\mathbf{4077 \pm 900}$	4198 ± 560	$\mathbf{2161 \pm 640}$
DP	$\mathbf{0.98 \pm 0.22}$	$\mathbf{0.80 \pm 0.21}$	$\mathbf{0.90 \pm 0.35}$	7.06 ± 7.73	8172 ± 747	7612 ± 543	$\mathbf{2944 \pm 1399}$	2201 ± 744

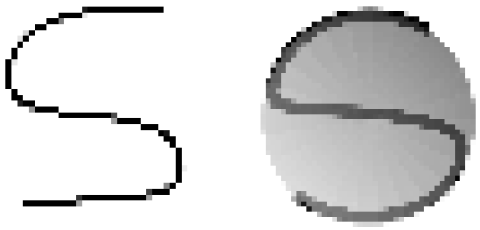
TABLE I: Average dynamic time warping distance (DTWD) and jerkiness (a.k.a smoothness) for trajectory-based RFMP and DP. DTWD is computed between the demonstrations and the reproductions initialized as the demonstrations, while the smoothness is averaged over all the reproductions displayed in Fig. 2.

	DTWD, $\mathbb{R}^2$			Jerkiness, $\mathbb{R}^2$			DTWD, $\mathcal{S}^2$			Jerkiness, $\mathcal{S}^2$		
T_a	2	4	8	2	4	8	2	4	8	2	4	8
RFMP	1.13 ± 0.29	2.31 ± 1.25	1.87 ± 0.94	$\mathbf{3454 \pm 276}$	$\mathbf{3729 \pm 475}$	$\mathbf{2120 \pm 273}$	$\mathbf{0.52 \pm 0.18}$	0.90 ± 0.34	0.95 ± 0.32	$\mathbf{2845 \pm 335}$	6169 ± 556	$\mathbf{4077 \pm 900}$
DP	$\mathbf{0.70 \pm 0.07}$	$\mathbf{0.76 \pm 0.25}$	$\mathbf{0.98 \pm 0.22}$	11905 ± 1962	7289 ± 939	8172 ± 747	0.60 ± 0.16	$\mathbf{0.70 \pm 0.33}$	$\mathbf{0.80 \pm 0.21}$	6586 ± 3140	$\mathbf{4239 \pm 1306}$	7612 ± 543

TABLE II: Average dynamic time warping distance (DTWD) and jerkiness (a.k.a smoothness) for RFMP and DP with different prediction horizons $T_a = \{2, 4, 8\}$. DTWD is computed between the demonstrations and the reproductions initialized as the demonstrations, while the smoothness is averaged over all the reproductions displayed in Fig. 3.

	Trajectory-based		Visuomotor	
Dataset	S, $\mathbb{R}^2$	S, $\mathcal{S}^2$	S, $\mathbb{R}^2$	S, $\mathcal{S}^2$
RFMP	$\mathbf{803 \pm 55}$	1539 ± 23	$\mathbf{1355 \pm 110}$	$\mathbf{2351 \pm 88}$
DP	1142 ± 17	$\mathbf{1147 \pm 26}$	2462 ± 141	2662 ± 541

TABLE III: Inference times (in milliseconds) per prediction step for RFMP and DP. These are averaged across the 50 prediction steps with $T_a = 8$, for both the 14 reproductions displayed in Fig. 2 for trajectory-based policies, and the 7 reproductions displayed in Fig. 5 for visuomotor policies.



(a) $\mathbb{R}^2$ (b) $\mathcal{S}^2$

Fig. 4: Examples of visual observations at the end of a demonstration of the LASA dataset S.

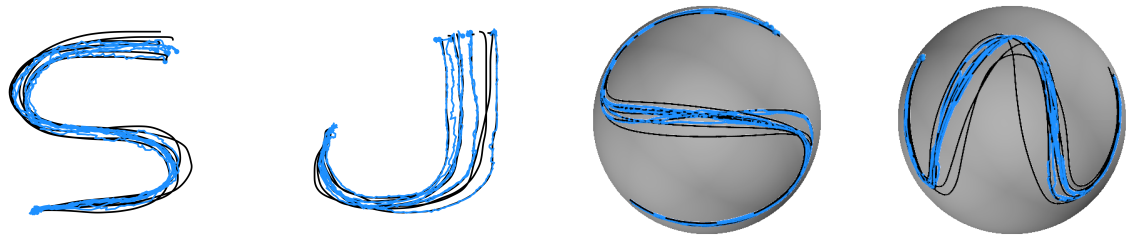
B. Towards visuomotor policies

In this section, we study the case where the RFMP vector field is conditioned on visual observations, thus resembling a visuomotor diffusion policy. Similarly to Section IV-A, we use a dataset defined as $\{\{\{\boldsymbol{a}_{m,\tau}, \boldsymbol{o}_{m,\tau}\}_{c=1}^{\tau-2}\}_{\tau=2}^{T_m}\}_{m=1}^{M}$ of $M = 7$ demonstrations containing $T_m = 200$ timesteps each, where $\boldsymbol{a}_{m,\tau} = [\boldsymbol{a}_{m,\tau}, \dots, \boldsymbol{a}_{m,\tau+T_a}]$ and $\boldsymbol{o}_{m,\tau} = [\boldsymbol{o}_{m,\tau-1}, \boldsymbol{o}_{m,c}, \tau - c]$ are the action and observation vectors of the τ-th step of the m-th demonstration. Again, all action and observation vectors are normalized and projected onto the manifold $\mathcal{M}$ of interest. However, in this case, the observations $\boldsymbol{o}_\tau \in \mathcal{M}$ are given by the latent encodings (a.k.a feature vectors) of 48×48 raw grayscale images depicting the temporal progress of the task. Examples of such images are shown in Fig. 4. Specifically, our vision perception backbone, which maps raw grayscale images to observation vectors, is exactly the same used in DP [4]. Namely, we used a standard ResNet-18 in which we replaced: (1) the global average pooling with a spatial softmax pooling, and (2) BatchNorm with GroupNorm. The former modification maintains spatial information [34], while the latter stabilizes the training [35].

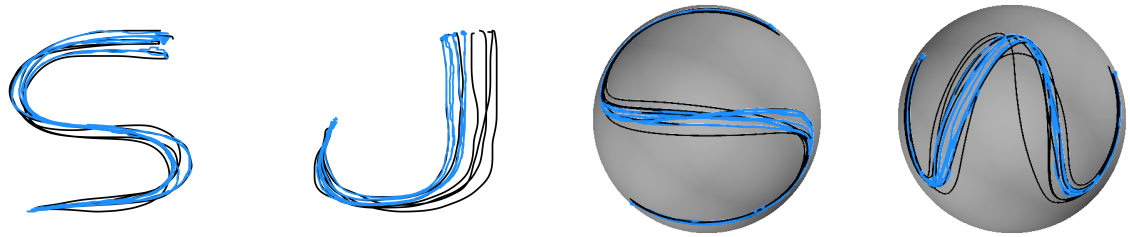
We trained the visual encoder end-to-end with our RFMP, for which we used the same base distributions p_0 and prediction horizon $T_a = 8$ as in Section IV-A. In this visuomotor RFMP, we empirically observed that shortening the observations horizon used to sample the context observation increases temporal consistency and improves the smoothness on the predicted actions. Therefore, in the following experiments, we sample $c \sim \mathcal{U}\{\tau - w, \dots, \tau - 2\}$ with $w = 50$. Figure 5a shows the demonstrations and the reproduced trajectories of the learned visuomotor RFMP. For both demonstrations and reproductions, the initial observations correspond to blank images for policies trained in $\mathbb{R}^2$, and an empty grayscale sphere for policies in $\mathcal{S}^2$. Similarly to the trajectory-based policies, the visuomotor RFMP successfully reproduces trajectories that match the demonstrations pattern in both the Euclidean and the Riemannian settings.

We compare visuomotor RFMP against visuomotor DP. As in [4], and similarly to the visuomotor RFMP, we train the vision perception backbone (modified ResNet-18) end-to-end with the CNN-based diffusion network described in Section IV-A. Figure 5b shows the trajectories obtained by sequentially executing the actions inferred by the visuomotor DP. Similarly to RFMP, the trajectories closely match the demonstrations. Interestingly, the visuomotor RFMP competitively performs when compared to the visuomotor DP in terms of the DTWD metric, as shown in Table IV, despite having a simpler architecture parametrizing the RFMP vector field. Moreover, as observed in Section IV-A for the trajectory-based case, visuomotor RFMP leads to smooth trajectories, especially for policies on $\mathcal{S}^2$ as indicated by the low jerkiness values in Table IV. Let us emphasize once more that RFMP ensures that the predicted actions lie on the manifold of interest, as opposed to DP which does not provide such guarantees.

Finally, we compare the inference time of visuomotor RFMP and DP in Table III. We observe a reduction of $\sim 45\%$ ($\sim$ 900ms) for the inference time of RFMP compared to DP. Interestingly, this reduction is greater for visuomotor policies compared to the trajectory-based case. Moreover, the inference time of RFMP on the sphere $\mathcal{S}^2$ is similar to that of DP, despite that RFMP uses Riemannian-specific



(a) RFMP trajectories.



(b) DP trajectories.

Fig. 5: Demonstrations (—) and trajectories reproduced by the visuomotor RFMP and DP (—) on the LASA datasets S and J in $\mathbb{R}^2$ (*left*) and on the LASA datasets S and W projected on $\mathcal{S}^2$ (*right*).

	DTWD				Jerkiness			
Dataset	S, $\mathbb{R}^2$	J, $\mathbb{R}^2$	S, $\mathcal{S}^2$	W, $\mathcal{S}^2$	S, $\mathbb{R}^2$	J, $\mathbb{R}^2$	S, $\mathcal{S}^2$	W, $\mathcal{S}^2$
RFMP	$\mathbf{1.22 \pm 0.44}$	$\mathbf{1.82 \pm 0.93}$	0.76 ± 0.27	$\mathbf{0.84 \pm 0.48}$	10543 ± 612	7655 ± 537	$\mathbf{3590 \pm 353}$	$\mathbf{4455 \pm 306}$
DP	1.29 ± 0.49	2.35 ± 1.66	$\mathbf{0.67 \pm 0.24}$	0.93 ± 0.48	$\mathbf{6198 \pm 755}$	$\mathbf{5588 \pm 801}$	5903 ± 170	5042 ± 136

TABLE IV: Average dynamic time warping distance (DTWD) and jerkiness (a.k.a smoothness) for visuomotor RFMP and DP. DTWD is computed between the demonstrations and the reproductions displayed in Fig. 5 and the smoothness is averaged over the same reproductions.

ODE solvers which are computationally more expensive than Euclidean ODE solvers. The reported findings indicate comparable performance between RFMP and DP in terms of task completion. However, RFMP exhibits a clear advantage in generating smoother action predictions, and this advantage remains consistent regardless of the prediction horizon. Furthermore, RFMP boasts significantly faster inference times compared to DP. These attributes make RFMP a compelling choice for real-time applications in various robotic domains.

V. CONCLUSION

We introduced Riemannian Flow Matching Policies (RFMP), a novel learning framework that leverages the simplicity and fast inference of flow matching models to model visuomotor robot policies on Riemannian manifolds. We evaluated RFMP on both trajectory-based and vision-based settings using the LASA dataset. Our results demonstrated that RFMP successfully learns policies that reproduce the demonstration patterns even for initial conditions outside the training data. Compared to Diffusion Policies (DP), RFMP generates smoother predicted trajectories with significantly lower inference times. Interestingly, RFMP exhibited less performance degradation with decreasing action prediction horizons. Notably, RFMP achieved this competitive performance using a simple MLP architecture for its vector field, in contrast to the more powerful CNN architecture employed by DP in score matching. Our proof-of-concept experiments showed the potential of RFMP for learning complex visuomotor policies in real-world robotic applications. Future work will evaluate the performance of RFMP in real-world robotics applications. Moreover, we will explore more powerful representations for the RFMP vector field and more informative prior models.

REFERENCES

[1] M. Janner, Y. Du, J. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," in *Intl. Conf. on Machine Learning (ICML)*, 2022, pp. 9902–9915.

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

[3] 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.

[4] 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.

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

[6] 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.

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

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

[9] A. Davtyan, S. Sameni, and P. Favaro, "Efficient video prediction via sparsely conditioned flow matching," in *Intl. Conf. on Computer Vision (ICCV)*, 2023, pp. 23 206–23 217.

[10] 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.

[11] J. Bose, T. Akhound-Sadegh, K. FATRAS, G. Huguet, J. Rector-Brooks, C.-H. Liu, A. C. Nica, M. Korablyov, M. M. Bronstein, and A. Tong, "SE(3)-stochastic flow matching for protein backbone generation," in *Intl. Conf. on Learning Representations (ICLR)*, 2024.

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] J. Zhang, H. B. Mohammadi, and L. Rozo, "Learning Riemannian stable dynamical systems via diffeomorphisms," in *Conference on Robot Learning (CoRL)*, 2023, pp. 1211–1221.

[19] R. T. Q. Chen and Y. Lipman, "Flow matching on general geometries," in *Intl. Conf. on Learning Representations (ICLR)*, 2024.

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

[21] J. M. Lee, *Introduction to Riemannian Manifolds*. Springer, 2018.

[22] 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," *Transactions on Machine Learning Research (TMLR)*, 2024.

[23] J. Dormand and P. Prince, "A family of embedded runge-kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[24] M. Poli, S. Massaroli, A. Yamashita, H. Asama, J. Park, and S. Ermon, "TorchDyn: Implicit models and neural numerical methods in PyTorch," *arXiv preprint arXiv:2009.09346*, 2020.

[25] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[26] 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.

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

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

[29] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *Intl. Conf. on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 139, 2021, pp. 8162–8171.

[30] T. Yoon, J. Y. Choi, S. Kwon, and E. K. Ryu, "Diffusion probabilistic models generalize when they fail to memorize," in *ICML 2023 Workshop on Structured Probabilistic Inference & Generative Modeling*, 2023.

[31] S. Balasubramanian, A. Melendez-Calderon, A. Roby-Brami, and E. Burdet, "On the analysis of movement smoothness," *Journal of NeuroEngineering and Rehabilitation*, vol. 12, no. 112, 2015.

[32] 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.

[33] A. Lou, M. Xu, A. Farris, and S. Ermon, "Scaling Riemannian diffusion models," in *Neural Information Processing Systems (NeurIPS)*, 2023.

[34] 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)*, ser. Proceedings of Machine Learning Research, vol. 164, 2022, pp. 1678–1690.

[35] Y. Wu and K. He, "Group normalization," in *European conference on computer vision (ECCV)*, 2018, pp. 3–19.