




# Learning Stable Vector Fields on Lie Groups

Julen Urain , Graduate Student Member, IEEE, Davide Tateo , and Jan Peters 

**Abstract**—Learning robot motions from demonstration requires models able to specify vector fields for the full robot pose when the task is defined in operational space. Recent advances in reactive motion generation have shown that learning adaptive, reactive, smooth, and stable vector fields is possible. However, these approaches define vector fields on a flat Euclidean manifold, while representing vector fields for orientations requires modeling the dynamics in non-Euclidean manifolds, such as Lie Groups. In this paper, we present a novel vector field model that can guarantee most of the properties of previous approaches i.e., stability, smoothness, and reactivity beyond the Euclidean space. In the experimental evaluation, we show the performance of our proposed vector field model to learn stable vector fields for full robot poses as SE(2) and SE(3) in both simulated and real robotics tasks.

**Index Terms**—Imitation learning, lie groups, learning from demonstration, machine learning for robot control, reactive motion generation.

## I. INTRODUCTION

**D**ATA-DRIVEN motion generation methods such as Imitation Learning (IL) [1], [2] bring the promise of teaching our robots the desired behavior from a set of demonstrations without further programming of the robot skill. Similar to the CNN networks in computer vision, choosing a good representation of the motion generator might help in the quality of the robot's performance, when learning a policy directly from data. During the last two decades, there has been vast research on learning policy architectures [3], [4], [5], [6], [7] that guarantee a set of desirable inductive biases. Popularized as Movement

Primitive (MP), the community explored a wide set of policy architectures with inductive biases such as Smoothness [6], Stability [3], [5] or cyclic performance [4], [8].

**Learning Movement Primitives for orientations requires additional insights in the architecture of the model.** There exist multiple representation forms for the orientation, such as Euler angles, rotation matrices, or quaternions. Euler angles have an intuitive representation, but the representation is not unique and might get stuck in singularities (i.e. gimbal lock). These properties make Euler angles undesirable for reactive motion generation [9]. Instead of Euler angles, rotation matrices and quaternions are preferred representations for reactive motion generation. Nevertheless, they require special treatment, given they are not defined in the Euclidean space. Rotation matrices are represented by the special orthogonal group, SO(3), while quaternions are represented in the 3-sphere,  $S^3$ . Thus, in the context of modeling orientation MP, there has been wide research integrating manifold constraints and MP. In [10], [11], [12], Dynamic Movement Primitives (DMP) [3] were adapted to learn orientation DMP, by representing DMP for quaternions [10], [11], [12] or rotation matrices [12]. More recently, orientation MP have been also considered to adapt Kernelized Movement Primitives (KMP) [13], [14], Task Parameterized GMM (TP-GMM) [7], [15] and Probabilistic Movement Primitives (ProMP) [6], [16]. Nevertheless, most of the MP are rather phase dependant or lack stability.

We propose to learn Stable Vector Field (SVF) [5], [17], [18] on position and orientations. SVF are a family of dynamic systems that are autonomous (i.e. don't have phase dependency, but only depend on the current state) and are inherently stable in terms of Lyapunov. In contrast with DMP [10] or KMP [14], SVF are **inherently reactive to disturbances** without the requirement of any phase adaptation. In contrast with TP-GMM [15], SVF are **inherently stable**, generating stable motions beyond the expert demonstrations. These properties makes SVF ideal for human-robot interaction or to combine them with other vector fields as in Riemannian Motion Policies (RMP) [19].

The contribution of this paper are: (1) We introduce a novel learnable SVF function that can generate stable motions on Lie Groups. Our proposed function generalizes Euclidean space *diffeomorphism-based SVF* [17], [18], [20] to arbitrary smooth manifolds such as Lie Groups. (2) To learn these SVF, we propose a neural network architecture that represents diffeomorphic functions in robotic-relevant Lie Groups such as SE(2) and SE(3). (3) Finally, we compare the performance of our proposed model w.r.t. learning the vector fields for Euler angles and learning the vector fields in the configuration space of the robot.

Manuscript received 9 August 2022; accepted 12 September 2022. Date of publication 2 November 2022; date of current version 14 November 2022. This letter was recommended for publication by Associate Editor A. M. Ghalamzan Esfahani and Editor D. Kulic upon evaluation of the reviewers' comments. This work was supported in part by European Union's Horizon 2020 Research and Innovation Programmes under Grant 820807 (SHAREWORK) and in part by the German Federal Ministry of Education and Research (BMBF) within a Subproject "Modeling and exploration of the operational area, design of the AI assistance and the Legal Aspects of the use of Technology" of the collaborative KIARA Project under Grant 13N16274. (Corresponding author: Julen Urain.)

Julen Urain and Davide Tateo are with the Computer Science Department, Institute for Intelligent Autonomous Systems, Technische Universität Darmstadt, 64289 Darmstadt, Germany (e-mail: urain@ias.informatik.tu-darmstadt.de; davide@robot-learning.de).

Jan Peters is with the Computer Science Department, Institute for Intelligent Autonomous Systems, Technische Universität Darmstadt, 64289 Darmstadt, Germany, also with the Research Department: Systems AI for Robot Learning, German Research Center for AI (DFKI), Germany, also with the Hessian Center for AI, Hessian.AI, Germany, and also with the Centre for Cognitive Science, Germany (e-mail: mail@jan-peters.net).

Videos and code are available at: <https://sites.google.com/view/svf-on-lie-groups/>

This letter has supplementary downloadable material available at <https://doi.org/10.1109/LRA.2022.3219019>, provided by the authors.

Digital Object Identifier 10.1109/LRA.2022.3219019

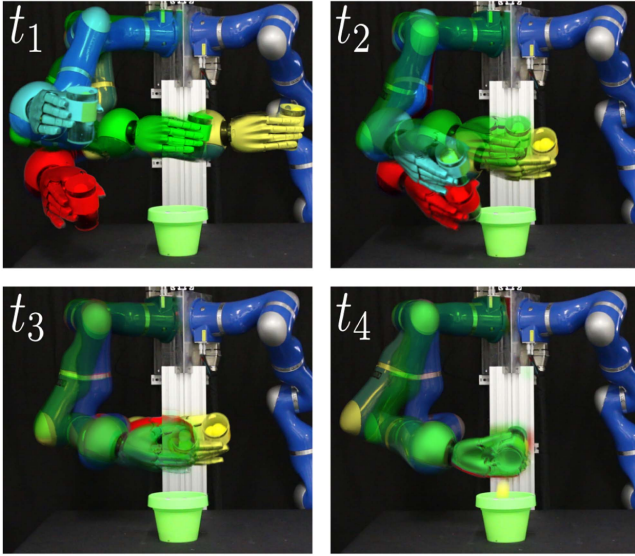


Fig. 1. Robot pouring trajectories generated by  $SE(3)$ -stable vector fields. Each color represents a trajectory starting from a different initial configuration. Given the stability properties, all the trajectories end up with the same orientation and position on the end effector.

### A. Background

A  $n$ -manifold  $\mathcal{M}$  is called *smooth* if it is locally diffeomorphic to an Euclidean space  $\mathbb{R}^n$  [21]. For each point  $\mathbf{x} \in \mathcal{M}$ , there exist a coordinate chart  $(U, \psi)$ , where  $U$  is an open subset in the manifold,  $U \subseteq \mathcal{M}$  and  $\psi : U \rightarrow \hat{U}$ , is a diffeomorphism from the subset  $U$  to a subset in the Euclidean space  $\hat{U} \subseteq \mathbb{R}^n$ . This chart allows us to represent a section of the manifold  $\mathcal{M}$  in a Euclidean space and do calculus.

For any point in the manifold,  $\mathbf{x} \in \mathcal{M}$ , we can attach a *tangent space*,  $T_{\mathbf{x}}\mathcal{M}$  that contains all the possible vectors that are tangential at  $\mathbf{x}$ . Intuitively, for any possible curve in  $\mathcal{M}$  passing through  $\mathbf{x}$ , the velocity vector of the curve at  $\mathbf{x}$  will belong to the tangent space,  $\mathbf{v} \in T_{\mathbf{x}}\mathcal{M}$ . Thus, a *vector field* in the manifold  $\mathcal{M}$  is a function that maps any point in the manifold to a vector in the tangent space,<sup>1</sup>  $\mathbf{g} : \mathcal{M} \rightarrow T\mathcal{M}$ . The *LogMap* is the map that moves a point in the manifold  $\mathcal{M}$  to the tangent space, and the *ExpMap* is the map that moves a point from the tangent space to the manifold.

A map  $\Phi : \mathcal{M} \rightarrow \mathcal{N}$  between smooth manifolds induces a linear map between their corresponding tangent spaces. For any point  $\mathbf{x} \in \mathcal{M}$ , the differential of  $\Phi$  at  $\mathbf{x}$  is a linear map,  $d\Phi_{\mathbf{x}} : T_{\mathbf{x}}\mathcal{M} \rightarrow T_{\Phi(\mathbf{x})}\mathcal{N}$ , from the tangent space at  $\mathbf{x} \in \mathcal{M}$  to the tangent space at  $\Phi(\mathbf{x}) \in \mathcal{N}$  (Fig. 2). The differential,  $d\Phi_{\mathbf{x}}$ , is used to map vectors between tangent spaces. The *pullback* operator is the linear operation  $d\Phi_{\mathbf{x}}^* : T_{\Phi(\mathbf{x})}\mathcal{N} \rightarrow T_{\mathbf{x}}\mathcal{M}$  that maps a vector from  $T_{\Phi(\mathbf{x})}\mathcal{N}$  to  $T_{\mathbf{x}}\mathcal{M}$ .

### B. Related Work

*a) Stable Vector Fields:* SVF models are powerful motion generators in robotics given they are robust to perturbations

<sup>1</sup>The precise term for  $T\mathcal{M}$  is tangent bundle. The tangent bundle is the disjoint set of all tangent spaces. The tangent space is defined at a certain point  $\mathbf{x}$ ,  $T_{\mathbf{x}}\mathcal{M}$ . For simplicity, with a slight terminology abuse, in this paper, we use the term tangent space.

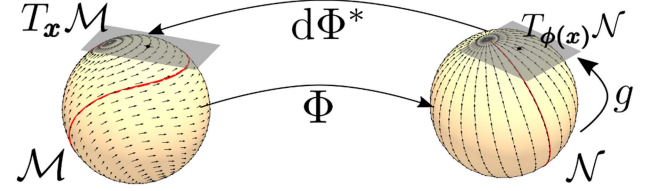


Fig. 2. In our work, we compute the vector field in  $\mathcal{M}$  by pulling back the vector field from the latent manifold  $\mathcal{N}$ . Given a point  $\mathbf{x} \in \mathcal{M}$ , we first map it to the latent manifold  $\mathbf{y} = \Phi(\mathbf{x})$  with  $\mathbf{y} \in \mathcal{N}$ . Then, we compute the vector in the latent manifold. Given a vector field  $\mathbf{g} : \mathcal{N} \rightarrow T\mathcal{N}$ , we compute  $\dot{\mathbf{y}} \in T_{\Phi(\mathbf{x})}\mathcal{N}$ . Finally, we apply the pullback linear operator to compute  $\dot{\mathbf{x}} = d\Phi_{\mathbf{x}}^*(\dot{\mathbf{y}})$  in the tangent space of  $\mathcal{M}$ ,  $\dot{\mathbf{x}} \in T_{\mathbf{x}}\mathcal{M}$ . As we can observe, the diffeomorphic function  $\Phi$  will deform the space and a trajectory (red line) or a vector field in the manifold  $\mathcal{N}$  will be deformed in  $\mathcal{M}$ .

and generalize the motion generation beyond the demonstrated trajectories. After the seminal work by Khansari et al. [5], several works [18], [20], [22], [23], [24] have proposed novel SVF models covering a wider family of solutions. Our work is particularly close to diffeomorphism based SVF models [18], [20], [22], [23]. Specifically, in this paper, we extend the class of solutions to non-Euclidean manifolds, such as Lie groups.

*b) Invertible Neural Networks (INN) in Smooth Manifolds:* Invertible Neural Network (INN) are a family of neural networks that guarantee to represent bijective functions. The study of modeling INN for smooth manifolds has been mainly developed for density estimation. A set of previous works [25], [26], [27] have proposed INN for specific manifolds, such as Tori or Sphere manifolds. A more recent work [28] proposes a manifold agnostic approach, on which Neural ODE [29], [30] are adapted to manifolds. In [31], INN are proposed for Lie Groups. Similar to our work, they also exploit the Lie algebra to learn expressive diffeomorphisms, but the proposed model is limited to density estimation.

## II. PROBLEM STATEMENT

We aim to solve the problem of modeling SVF on Lie Groups. In particular, we model our SVF by diffeomorphisms. Diffeomorphism-based SVF represent the vector field in the observation space as the deformed vector field of a certain latent space [17], [18], [20], [23]. These models assume there exist a **stable vector field in a latent space**  $\mathbf{g} : \mathcal{N} \rightarrow T\mathcal{N}$ . Then, given a **parameterized diffeomorphic mapping**  $\Phi$ , that maps any point in observation space  $\mathcal{M}$  to the latent space  $\mathcal{N}$ ,  $\Phi : \mathcal{M} \rightarrow \mathcal{N}$ , we can represent the dynamics in the observation space

$$\dot{\mathbf{x}} = d\Phi_{\mathbf{x}}^* \circ \mathbf{g} \circ \Phi(\mathbf{x}), \quad (1)$$

in terms of the latent dynamics  $\mathbf{g}$  and the diffeomorphism  $\Phi$ .  $d\Phi_{\mathbf{x}}^*$  is the **pullback operator** that maps a velocity vector from the latent space to the observation space. Intuitively, as shown in Fig. 2, the diffeomorphic function  $\Phi$  deforms the space changing the direction of the vector field in the observation space. The stability guarantees of diffeomorphism-based SVF have been previously proven in terms of Lyapunov [17], [18].

Previous *diffeomorphism-based* SVF are limited to Euclidean spaces, without representing motion policies in the orientation.

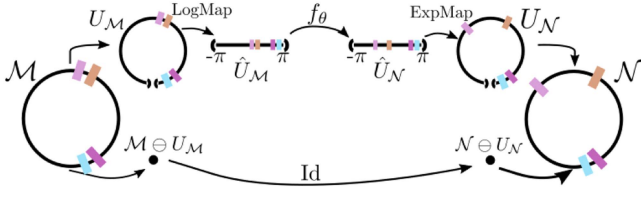


Fig. 3. A visual representation of the  $\Phi$  function for 1-sphere ( $S^1$ ). The points in  $S^1$  are split into two groups. For the points in  $U_M$ , the diffeomorphism is composed by first, mapping the points to the first-cover  $\hat{U}_M$  by the LogMap, then applying a bounded Euclidean diffeomorphism between  $\hat{U}_M$  and  $\hat{U}_N$  and mapping the points back to the manifold, by the ExpMap. For the points not belonging to  $U_M$ , we simply apply the identity map. If  $f_\theta$  is the identity map close to the boundaries  $-\pi$  and  $\pi$ ; the map is diffeomorphic for the whole  $S^1$ . We add a few markers to represent the space deformation along the mappings.

Euclidean SVF assumes (i) that  $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$  defines a bijective mapping between Euclidean spaces, (ii) in Euclidean spaces, the tangent space and the manifold are in the same space, and then, the latent dynamics are  $g : \mathbb{R}^n \rightarrow \mathbb{R}^n$  and, (iii) given  $\Phi$  defines a mapping between Euclidean spaces, the pullback operator is represented by the Jacobian pseudoinverse of  $\Phi$ ,  $d\Phi_x^* = J_\Phi^\dagger$ .

In our work, given we are required to model the SVF on Lie Groups, we need to (i) model a  $\Phi$  function that is bijective between Lie Groups, (ii) investigate how to model stable latent dynamics for Lie Groups and (iii) investigate how to model the pullback operator given the diffeomorphism  $\Phi$ .

### III. STABLE VECTOR FIELDS ON LIE GROUPS

As introduced in Section II, modelling *diffeomorphism-based* SVF on Lie Groups requires additional insights in the modelling of the three main elements  $\Phi$ ,  $g$  and  $d\Phi^*$ . In the following, we introduce our proposed models to represent each of these elements and we add a control block diagram on Fig. 4 to provide intuition on how to use the proposed SVF in practice.

#### A. Diffeomorphic Mapping $\Phi$

We introduce our proposed function to learn diffeomorphisms between Lie Groups,  $\Phi : \mathcal{M} \rightarrow \mathcal{N}$ . Both  $\mathcal{M}$  and  $\mathcal{N}$  are manifolds for the same Lie group, with  $\mathcal{M}$  representing the Lie group in the observation space and  $\mathcal{N}$ , the Lie group in the latent space. A simple example of  $\Phi$  is given by the rotation function. Given  $X \in \mathcal{M} = SO(3)$  and  $Y \in \mathcal{N} = SO(3)$ , the rotation function  $Y = \Phi(X) = RX$ , applies a linear diffeomorphic mapping between  $\mathcal{M}$  and  $\mathcal{N}$ .

Nevertheless, representing nonlinear diffeomorphic mappings for Lie groups is challenging. In our work, we propose to exploit the tangent space to learn these mappings. In contrast with the manifold, the tangent space is a Euclidean space, making it easier to model nonlinear diffeomorphic functions.

The topology of the Lie groups and their Lie algebra are not the same. Then, it is impossible to define a single diffeomorphic function  $\Phi$  that maps all the points in the group to the Lie algebra. To make proper use of the Lie algebra and still guarantee the diffeomorphism for the whole Lie group, we propose to model the diffeomorphism by parts. We visualize an example of the proposed function in Fig. 3. The points in the Lie Group are

split into two sets. We consider a coordinate chart  $U_M \subseteq \mathcal{M}$  that defines a set of almost all the points in the Lie group. Then, we group all the points not belonging to the set  $U_M$  in a different set,  $x \in \mathcal{M} \ominus U_M$ . For example, in the example on Fig. 3, we group all the points except the antipodal point in  $U_M$  and put the antipodal point in the set  $\mathcal{M} \ominus U_M$ . The points in the set  $U_M$  are mapped to a set in the latent manifold,  $U_N \subseteq \mathcal{N}$ . The points in the set  $\mathcal{M} \ominus U_M$  are mapped to the latent space set  $\mathcal{N} \ominus U_N$ . Given that  $\mathcal{M}$  and  $\mathcal{N}$  are represented in the same Lie Group, the sets in the observation space and the latent space are also the same.

$$\Phi(x) = \begin{cases} \text{ExpMap} \circ f_\theta \circ \text{LogMap}(x) & \text{if } x \in U_M \\ x & \text{if } x \in \mathcal{M} \ominus U_M. \end{cases} \quad (2)$$

For any element in the coordinate chart  $x \in U_M$ , we define the map from  $U_M$  to  $U_N$ , through the tangent space,  $\Phi : \text{ExpMap} \circ f_\theta \circ \text{LogMap}$ . The function first maps a point in the Lie group to the Lie algebra by the LogMap. For any point  $x \in U_M$ , it will map to a point in a subset of the tangent space,  $\hat{x} \in \hat{U}_M \subseteq T_{x_H} \mathcal{M}$ . We call **first cover of the tangent space** to  $\hat{U}_M$ . The map between  $U_M$  and  $\hat{U}_M$  is guaranteed to be diffeomorphic given the LogMap properties [21]. Then, we apply a Euclidean diffeomorphism  $f_\theta$  between the first covers of the observation space  $\hat{U}_M$  and the first covers of the latent space,  $\hat{U}_N$ . We introduce our proposed  $f_\theta$  in Section IV. Finally, we can map the points  $\hat{y} \in \hat{U}_N$  back to  $y \in U_N \subseteq \mathcal{N}$  by the ExpMap and represent it in the Lie Group. Given the three steps are diffeomorphic, we can guarantee that  $\Phi$  applies a diffeomorphism between  $U_M$  and  $U_N$ . For the points not belonging to the set  $U_M$ , we apply the identity map. The identity map is also diffeomorphic.

Even if each part in (2) is diffeomorphic in itself, to guarantee the function  $\Phi$  is diffeomorphic in the whole Lie group, we require to guarantee the function is continuous and differentiable in the boundaries between  $U_M$  and  $\mathcal{M} \ominus U_M$ . To do so, we impose structurally  $f_\theta$  to become the identity map  $f_\theta(\hat{x}) = \hat{x}$  when approaching to the boundaries of the set  $\hat{U}_M$ . Thus,

$$\begin{aligned} \Phi(x) &= \text{ExpMap} \circ f_\theta \circ \text{LogMap}(x) \\ &= \text{ExpMap} \circ \text{LogMap}(x) = x \end{aligned} \quad (3)$$

when  $x$  is close to the boundaries of  $U_M$ .

1) *An Intuitive Example for 1-Sphere ( $S^1$ ) Manifold*: The 1-sphere manifold is composed by all the points in a circle of radius  $r$ ,  $S^1 : \{x \in \mathbb{R}^2 : \|x\| = r\}$ . We visualize this manifold in Fig. 3. To model a diffeomorphic transformation between  $\mathcal{M}$  and  $\mathcal{N}$ , we propose to split the manifold in two sets: the set  $U_M$  considers all the points in the manifold except the point in the south  $x_S = (0, r)$ ,  $U_M = S^1_{\neq x_S}$ . Equally, the set in the latent manifold  $U_N$ , also consider  $U_N = S^1_{\neq x_S}$ . The other set  $\mathcal{M} \ominus U_M = \{x_S\}$  is composed of the point not belonging to  $U_M$ . We can observe that  $U_M$  is diffeomorphic to the open line segment  $\hat{U}_M = (-\pi, \pi)$ . We refer to this set as first-cover of the tangent space  $\hat{U}_M = \hat{U}_N = (-\pi, \pi)$ . We can map any point from  $U_M$  to  $\hat{U}_M$  by the LogMap function. Inversely, we can map the points from the open line segment to the set  $U_M$  by the ExpMap function. We remark that points in  $U_M$  are



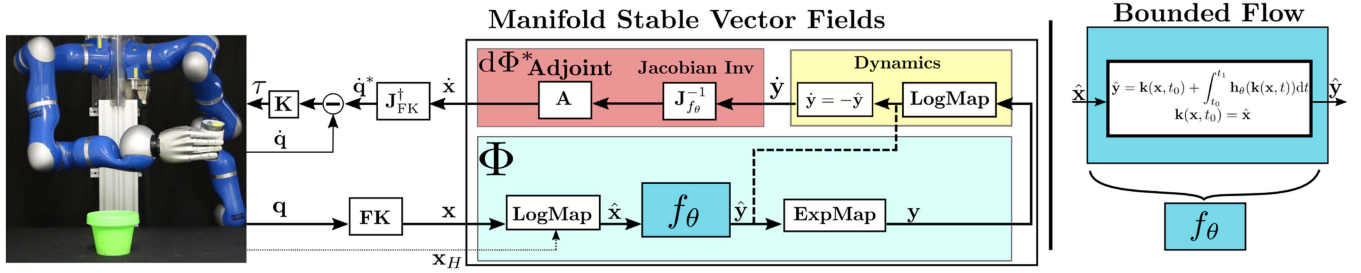


Fig. 4. Left: Manifold stable vector fields block diagram. Right: Proposed architecture for our diffeomorphic function  $f_\theta$ . As shown in (1), our manifold SVF is composed of three elements: a diffeomorphism  $\Phi$  (light blue box) (for simplicity, we only visualize the part related with the set  $U_M$ ), the latent dynamics  $g$  (yellow box) and, the pullback operator  $d\Phi^*$  (red box). The diffeomorphism  $\Phi$  is composed of three elements: the LogMap, a bounded diffeomorphism between first covers  $f_\theta$  (blue) and, the ExpMap. The pullback operator  $d\Phi^*$  has two elements: the Jacobian inverse, computed for the diffeomorphism  $f_\theta$ , and the Adjoint operator. Additionally, to control a robot, we first map the current joint configuration  $q$  to  $x \in SE(3)$  by Forward Kinematics. And once  $\hat{x} \in se(3)$  is computed, we map it back to the configuration space by  $J_{FK}^\dagger$ . Then, we apply a velocity controller in the configuration space. The dashed line from the output of  $f_\theta$  and the dynamics input represents a shortcut we consider in practice as long as the latent ExpMap and LogMap are computed in the same origin frame.

two-dimensional while points in  $\hat{U}_M$  are one-dimensional. Once the points are in the  $\hat{U}_M$ , we model a bounded diffeomorphic function  $f_\theta$  that maps the points in  $\hat{U}_M$  to  $\hat{U}_N$ . We present in Section IV how we model this bounded diffeomorphism  $f_\theta$ . This map can be thought of as a deformation of the line segments  $\hat{U}_M$  and  $\hat{U}_N$  is easy, representing it between the groups  $U_M$  and  $U_N$  is hard, given that  $U_M$  and  $U_N$  are not Euclidean spaces.

As shown before, to guarantee that  $\Phi$  is diffeomorphic for the whole manifold  $S^1$ , we need to guarantee that  $f_\theta$  becomes the identity map close to the boundaries of  $\hat{U}_M$ . For the case of  $S^1$ , the function  $f_\theta$  should approximate the identity map the closer the points are to  $-\pi$  and  $\pi$ . Intuitively, the function  $f_\theta$  represents a space deformation in  $(-\pi, \pi)$  that becomes the identity close to the boundaries  $-\pi$  or  $\pi$ . We illustrate this diffeomorphic map in Fig. 3.

### B. Latent Stable Dynamics $g$

For a given manifold  $\mathcal{N}$ , the vectors are represented in the tangent space of the manifold,  $T\mathcal{N}$ . Thus, a dynamic system in a manifold is a function that for any point in the manifold outputs a vector in the tangent space,  $g : \mathcal{N} \rightarrow T\mathcal{N}$ . Similarly to the transformation map  $\Phi$ , we propose to model the dynamics by parts

$$\dot{y} = g(y) = \begin{cases} -\text{LogMap}_{y_H}(y) & \text{if } y \in U_N \\ 0 & \text{if } y \in \mathcal{N} \ominus U_N \end{cases} \quad (4)$$

For any element in  $U_N$ , we first map the point to the tangent space centered at  $y_H$  and then, compute the velocity vector as  $\hat{y} = g(y) = -\hat{y}$ . These dynamics will induce a stable dynamic system in the manifold  $U_N$ , with a sink in  $y_H$ . For any point out of the set  $U_N$ , we set the velocity to zero. This will set an unstable equilibrium point for any point in  $\mathcal{N} \ominus U_N$ . In practice, given the LogMap in our dynamics (4) is the inverse of the ExpMap in  $\Phi$ , we can directly compute the dynamics using as input the output of  $f_\theta$  without moving to  $\mathcal{N}$  (dashed line in Fig. 4).

### C. Pullback Operator $d\Phi^*$

The pullback operator unrolls all the steps to the latent space,  $\mathcal{N}$ , done by the diffeomorphism,  $\Phi$ , back to the observation manifold,  $\mathcal{M}$ . Additionally, given the velocity vector is defined on the tangent space, the unrolling steps are done on the tangent space. The pullback operator for the mapping,  $f_\theta$ , is the Jacobian  $J_f$ . The inverse of the Jacobian, maps the velocity vector from the latent tangent space to the observation tangent space, centered in the origin,  $J_f^{-1} : T_{y_H}\mathcal{N} \rightarrow T_{x_H}\mathcal{M}$ . Additionally, we apply a second pullback operator to map the vector from the tangent space in the origin  $x_H$  to the tangent space in the current pose  $x$ ,  $A : T_{x_H}\mathcal{M} \rightarrow T_x\mathcal{M}$ . This linear map is known as the adjoint map and it can be understood as a change of reference frame for the velocity vectors. We direct the reader to [32] to find more information on how to model it. The whole pullback operator is then,  $d\Phi^* = A \circ J_f^{-1}$ .

## IV. BOUNDED FLOWS AS TRANSFORMATION $f_\theta$

In Section III-A, we propose to model the diffeomorphism between two subsets of the manifolds ( $U_M$  and  $U_N$ ) through the tangent space. To properly model the diffeomorphism, we have introduced a function  $f_\theta$  and defined its required properties. The function  $f_\theta$  should be a diffeomorphism and should become identity when approximating the boundaries of the tangent space sets  $\hat{U}_M$  and  $\hat{U}_N$ . To represent our function  $f_\theta$ , we build on top of the research on INN for Normalizing Flows [29], [33].

We propose to model the function  $f_\theta$  by adapting Neural ODEs [29] to our problem. Neural ODEs propose to model the diffeomorphism between two spaces by the flow of a parameterized vector field  $h_\theta$ . The flow  $k(x, t) : \mathbb{R}^{n+1} \rightarrow \mathbb{R}^n$ , represents the motion of a point for the time  $t$ , given the ODE,  $dx/dt = h_\theta(x) \equiv d(k(x, t))/dt = h_\theta(k(x, t))$

$$x_{t_1} = k(x, t_1) = x + \int_0^{t_1} h_\theta(k(x, t)) dt. \quad (5)$$

with  $t_1$  being a certain time instant and  $x$  the position of the particle in the instant  $t = 0$ . The flow function represents the position of a particle  $x$  follows given the vector field  $h_\theta$  at the instant  $t_1$ . In Neural ODEs, the function  $f_\theta$  is represented

by the output of the flow at time 1

$$\mathbf{y} = \mathbf{f}_\theta(\mathbf{x}) = \mathbf{k}(\mathbf{x}, t = 1) = \mathbf{x} + \int_0^1 \mathbf{h}_\theta(\mathbf{k}(\mathbf{x}, t)) dt. \quad (6)$$

As presented in [26], [29], the function is a diffeomorphism, as long as  $\mathbf{h}_\theta$  is a uniformly Lipschitz continuous vector field (Picard–Lindelöf theorem).

Additionally, to compute the pullback operation, we are required to compute the Jacobian matrix of  $\mathbf{f}_\theta$ ,  $\mathbf{J}_f = \nabla_{\mathbf{x}} \mathbf{k}(\mathbf{x}, t_1)$ . Given the vector field  $\mathbf{h}_\theta$ , there exists an ODE representing the time evolution of the Jacobian

$$\begin{aligned} \dot{\mathbf{J}}_f(\mathbf{x}, t) &= \nabla_{\mathbf{k}} \mathbf{h}_\theta(\mathbf{k}(\mathbf{x}, t)) \mathbf{J}_f(\mathbf{x}, t) \\ \mathbf{J}(\mathbf{x}, t_0) &= \mathbf{I}. \end{aligned} \quad (7)$$

In practice, we can use an arbitrary ODE solver and find the values for  $\mathbf{J}(\mathbf{x}, t_1)$  and  $\mathbf{k}(\mathbf{x}, t_1)$  solving (6) and (7). In our case, to guarantee a high control frequency rate, we apply the forward Euler method to solve the ODE and then compute the Jacobian by backward differentiation. It is important to remark that these dynamics are used to represent the diffeomorphism  $\mathbf{f}_\theta$  between two spaces and not to represent the desired vector fields.

Relevant consideration for our problem is that the function  $\mathbf{f}_\theta$  should define a diffeomorphism between two bounded sets  $\hat{U}_M$  and  $\hat{U}_N$  and the transformation should become identity close to the boundaries of these sets. Nevertheless, without any additional considerations on  $\mathbf{h}_\theta$ , the flow could move a point in  $\hat{U}_M$  to any point in  $\mathbb{R}^n$ , with  $n$  the dimension of the Euclidean space in which the set  $\hat{U}_N$  is. To bound the flow between the sets, we impose structurally that the vector field  $\mathbf{h}_\theta$  vanishes when approaching the boundaries. If the flow dynamics are zero, then, the input and the output are the same and we don't apply space deformation at that point. Given a distance function  $\alpha(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$  that measures how close we are to the boundaries, we define the vector field as

$$\mathbf{h}_\theta(\mathbf{x}) = \alpha(\mathbf{x}) \psi_\theta(\mathbf{x}), \quad (8)$$

with  $\psi$  an arbitrarily chosen uniformly Lipschitz continuous parameterized vector field and  $\alpha$  the scaling function of the dynamics to satisfy the desired constraints, preventing to move out of the set.  $\alpha$  becomes zero close to the boundaries. Then, close to the boundaries,

$$\mathbf{y} = \mathbf{f}_\theta(\mathbf{x}) = \mathbf{k}(\mathbf{x}, t_1) \approx \mathbf{k}(\mathbf{x}, t_0) = \mathbf{x}. \quad (9)$$

Thus, the function  $\mathbf{f}_\theta$  is guaranteed to approximate the identity in the boundaries.

Given the set  $\hat{U}_M$  varies between the manifolds, we consider different distance functions  $\alpha$  for each possible manifold. For the case of  $SO(2)$ , the first covers are  $\hat{U}_M = \hat{U}_N = (-\pi, \pi)$ . To impose identity map in the boundaries, the dynamics are weighted with  $\alpha(x) = (\pi - |x|)/\pi$ .  $\alpha$  is a function that moves from 1 to 0 when we approach the  $\pm\pi$  boundaries.

For the case of  $S^2$ , the sets are  $\hat{U}_M = \hat{U}_N = \{\mathbf{x} \in \mathbb{R}^2; \|\mathbf{x}\| < \pi\}$ . This set is diffeomorphic to the set in  $U_M = U_N \subset S^2$ , which considers all the points in the manifold except the antipodal point. To impose the dynamics to become zero close

---

### Algorithm 1: Behavioural Cloning for Manifold-SVF

---

**Given:** mSVF: Manifold SVF function ;  
 $\theta_0$ : initial parameters of the function mSVF;  
 $I$ : Optimization steps;  
 $\mathcal{D} : \{\{\mathbf{x}_{i,t}, \dot{\mathbf{x}}_{i,t}\}_{t=1}^{T_i}\}_{i=1}^N$ :  $N$  trajectories, of  $T_i$  length, with the position in  $\mathbf{x} \in \mathcal{M}$  and the velocity vector in  $\dot{\mathbf{x}} \in T_{\mathbf{x}}\mathcal{M}$

```

1 for  $i \leftarrow 0$  to  $I - 1$  do
2    $\mathbf{x}_b, \dot{\mathbf{x}}_b \sim \mathcal{D}$ ; // Sample a batch from dataset
3    $\mathcal{L}(\theta_i) = \frac{1}{B} \sum_{k=0}^B \|\dot{\mathbf{x}}_k - \text{mSVF}(\mathbf{x}_k; \theta_i)\|_2^2$ ;
4    $\theta_{i+1} \leftarrow \theta_i + \alpha \nabla_{\theta} \mathcal{L}(\theta_i)$ ;
5 return  $\theta^*$ ;
```

---

to the boundaries of the set, the distance function is  $\alpha(\mathbf{x}) = (\|\mathbf{x}\| - \pi)/\pi$ .

For the case of  $SO(3)$ , the sets are  $\hat{U}_M = \hat{U}_N = \{\mathbf{x} \in \mathbb{R}^3; \|\mathbf{x}\| < \pi\}$ . The sets are diffeomorphic to the  $SO(3)$  sets  $U_M = U_N = SO(3)_{\neq \mathbf{R}_\pi} \subset SO(3)$ , that consider all possible rotation matrices except the ones that have a  $\pi$  rotation from the origin. The dynamics are weighted by the function  $\alpha(\mathbf{x}) = (\|\mathbf{x}\| - \pi)/\pi$ .

For the case of the special Euclidean groups  $SE(2)$  and  $SE(3)$ , the orientation-related dimensions maintain the same first covers of the special orthogonal groups. For the position-related dimensions, we bound the first cover to the desired workspace. Given  $(\mathbf{p}, \theta) \in \mathfrak{se}(3)$ , with  $\mathbf{p}$  the position related variables and  $\theta$ , orientation related variables. We consider two scaling functions, one for orientations and one for positions. The orientation scaling function  $\alpha_{\text{ori}}(\theta)$  is computed given the scaling functions above. The scaling function for the positions  $\alpha_{\text{pos}}(\mathbf{p})$  can be used to enforce workspace limits and varies depending on the chosen workspace boundaries. We compute the distance function by  $\alpha(\mathbf{p}, \theta) = \alpha_{\text{pos}}(\mathbf{p}) \alpha_{\text{ori}}(\theta)$ .

## V. EXPERIMENTAL RESULTS

We present three experiments to evaluate the performance of our approach. In the first experiment, we illustrate, in a  $S^2$  manifold, the performance of our proposed  $\mathbf{f}_\theta$  w.r.t. functions that do not take into consideration the manifold and treat it as Euclidean. Even if  $S^2$  is not a Lie Group, we can apply the proposed approach also on it and serves as a useful manifold for illustration.

In the second and third experiments, we evaluate the performance of our model in the Lie Groups  $SE(2)$  and  $SE(3)$ , for a 2D peg-in-a-hole task and a pouring task respectively.

### A. Network Evaluation in $S^2$ Manifold

We study the problem of learning stable vector fields in 2-sphere,  $S^2$  by behavioral cloning (Algorithm 1). The objective of this experiment is to evaluate the influence of choosing different INN as mapping  $\mathbf{f}_\theta$ .

For evaluation, we consider three models. The three models use our proposed architecture in Fig. 4 and vary in the used diffeomorphism  $\mathbf{f}_\theta$ . We consider two models using the INN from previous works [18], [20] that considers a diffeomorphism in the whole Euclidean space  $\mathbf{f}_\theta : \mathbb{R}^n \rightarrow \mathbb{R}^n$  and our proposed INN that learns a diffeomorphism in bounded

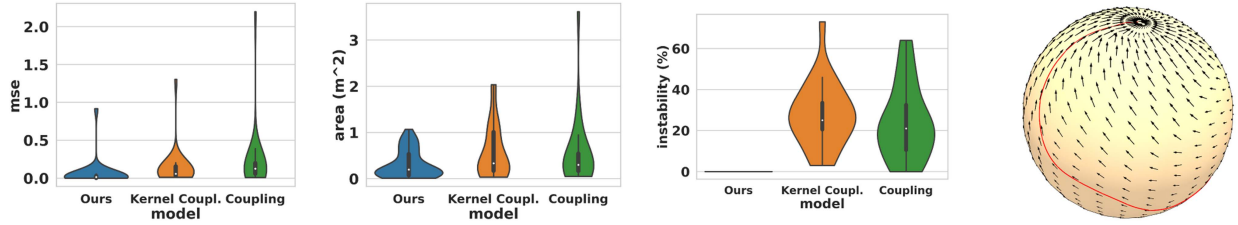


Fig. 5. Left: Kernel Coupling, Coupling, and Ours(Smooth Piecewise Linear) Layers compared in terms of Mean Squared Error (MSE), Area and Instability %. Kernel Coupling and Coupling Layer apply a diffeomorphism between  $\mathbb{R}^n$  and Ours between the first covers. Right: Example of LASA trajectory and learned vector field.

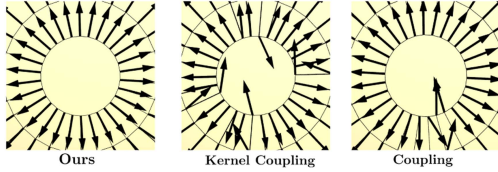


Fig. 6. Vector fields in the antipodal point of the Sphere. Our proposed diffeomorphism guarantees a source in the antipodal, while the unbounded INN does not.

domains,  $f_\theta : \hat{U}_M \rightarrow \hat{U}_N$ . We modified the LASA dataset [5] to  $S^2$  manifolds. We consider 22 different shape trajectories and evaluate the models given three metrics: MSE, Area, and Instability percentage. For measuring the instability percentage, we initialized a set of points in random positions on  $S^2$  and generated a trajectory with the learned vector fields. Then, we measured how many trajectories reach the target position after a certain period.

From Fig. 5, we can observe that the three architectures performed similarly in both MSE and Area measures and were able to mimic the performance of the demonstrations properly. This indicates that the proposed algorithm can learn vector fields on smooth manifolds. Nevertheless, as shown in the Instability % metric, the performance of the Kernel Coupling Layer [20] and the Coupling Layer [18] decay when initializing the trajectories in a random position. Given the Kernel Coupling Layer and the Coupling Layer define a diffeomorphism in the whole Euclidean space, they lack any guarantee of being bijective between  $\hat{U}_M$  and  $\hat{U}_N$ . Thus, these approaches lack guarantees about the stability of the vector field in  $\hat{U}_M$ . We can observe the instability of the vector fields by observing the antipodal point of the sphere, where the boundaries of the first cover  $\hat{U}_M$  are defined. As shown in Fig. 6, while our INN can guarantee all the vectors pointing out of the antipodal (a source in the antipodal point), the kernel coupling layer and coupling layer are not able to guarantee stability close to the boundaries generating oscillatory behaviors around the antipodal point.

### B. Evaluation of $SE(2)$ Stable Vector Fields in a 2D Peg-in-A-Hole Task

We consider the environment presented in Fig. 7. The robot is a 5-DOF robot moving in a 2d plane. The goal of the task is to move the end-effector of the robot into the hole while avoiding collisions against the walls. We generated a 1 K trajectory demonstration to train our models by applying

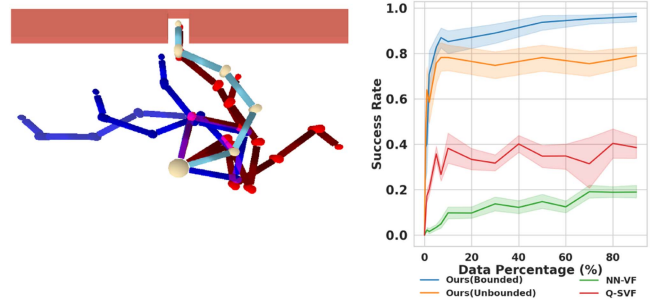


Fig. 7. Left: Peg-in-a-hole environment. We show in different colors, generated trajectories from different initial configurations. Right: Success rate Vs. Data percentage. We evaluate the performance of a set of models when trained with different amounts of data.

RRT-Connect [34] on the environment. We compare the performance of our model w.r.t. three baselines. First, we consider a vector field modeled by a naive fully connected neural network in the tangent space of  $SE(2)$ . Second, we trained a stable vector field in the configuration space,  $Q$ . Third, similarly to the experiment in  $S^2$ , we model a vector field with the architecture in Fig. 4, but consider a vanilla INN as  $f_\theta$  instead of the proposed INN. To evaluate the performances, we initialize the robot in a random configuration and reactively evolve the dynamics. To control the robot, we apply operational space control [35]. Given the current end-effector pose,  $x \in SE(2)$ , we compute the desired velocity at the end effector  $\dot{x} \in \mathbb{R}^3$  and pullback to the configuration space by the Jacobian pseudoinverse.

We present the results in Fig. 7. We measure the success of the different methods to approach the goal without colliding under different amounts of training data. The vanilla neural network model performed the worst with any amount of trained data. A vanilla-NN is not limiting the family of possible vector fields, thus it may learn vector fields with multiple equilibrium points, limit cycles, or even unstable ones. This results in highly unstable vector fields with poor performance. The results also show the relevance of choosing a good task space representation. Learning in  $SE(2)$  outperforms the configuration space approach. The difference in performance might be related to the vector field dimensionality, 5 for the configuration space and 3 for  $SE(2)$  and also, with the task itself: as the peg-in-a-hole task is defined in the operational space the  $SE(2)$  vector fields fit better the problem. Finally, we observe the benefit of our proposed INN w.r.t. vanilla INN approach. Given that the vanilla INN lacks global stability guarantees, the robot gets stuck in limit cycles and the performance decays.



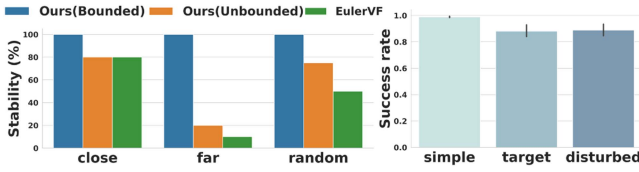


Fig. 8. Results for the pouring experiment. Right: simulated experiment results. We compare the stability property of the three models given three possible types of initial configurations (close to the target, far from the target, and random configuration). Left: real robot experiments results.

In conclusion, we have observed that (i) stability guarantees greatly improves the performance of the policy for behavioral cloning problems (ii) representing the vector field in a proper manifold can boost the performance, and (iii) a bounded INN guarantees stability, while the unbounded one does not, given  $\Phi$  is not diffeomorphic anymore.

### C. Learning a Pouring Task With $SE(3)$ Stable Vector Fields

In this experiment, we evaluate the performance of our method on a pouring task (Fig. 1). To properly pour, the robot requires to combine multiple positions and orientation changes. First, we compare in simulation our method with Euler angle-based vector fields. We consider two version of our model: One with bounded  $f_\theta$ , introduced in Section IV and one with a vanilla unbounded INN as  $f_\theta$  [36]. Then, we evaluate the performance of our model in a real robot under target modifications and human disturbances.

For this experiment, we use a 7 DoF Kuka LWR arm. The provided task demonstrations consist of 30 kinesthetic teaching trajectories with a wide variety of initial configurations. We considered different end-effector positions and orientations and trained the three models by behavioral cloning (Algorithm 1). To control the robot, we apply operational space control [35] for our proposed model (Fig. 4) and position control for the Euler angles vector field. Note that our proposed method adapts to any other type of robot (prismatic joints, parallel robot) by changing the forward kinematics function. We evaluate the three models in three scenarios, robot performance with an initial configuration close to the target, initial configuration far from the target, and random initial configuration. We consider 10 different initial configuration and measure the robot's performance. In the three cases, we measured the stability guarantees of the models (i.e. the guarantee of arriving at the target pose after a certain time). We present the experiment results in Fig. 8. From this figure, we can see that our model with the bounded function  $f_\theta$  outperformed the other models in the three cases. These results validate our claims on the requirements of defining a function  $f_\theta$  between the first covers, to guarantee stability in the whole Lie Group. Euler angle-based vector fields perform quite well for the case of close initial configuration. Euler Angles are an undesirable representation for feedback control due to their singularities and non-uniqueness. Nevertheless, we can assume these types of situations are rare close to the target and can perform relatively well. Nevertheless, their performance decay considering initial configurations far from the target. Given the non-uniqueness

of the Euler-angles, representing globally stable vector fields in Euler-angles is not possible. In the case of our model with vanilla INN, it shows unstable behavior far from the target, while it remains quite stable close to it. Diffeomorphism-based SVF lack stability guarantees if the function  $\Phi$  is not bijective. This lack of bijectiveness is more prone to happen close to the boundaries of the first cover and  $\Phi$  remains bijective close to the target, with the guarantee of being stable.

We also evaluate the performance of our model on a real robot, measuring the model's performance under target modifications and human disturbances. To adapt to different target positions, we use the current one  $x_{\text{target}} \in SE(3)$  as the origin of the LogMap (Fig. 4). This allows us to represent the vector fields relative to the current target position. We track the target pot by Optitrack motion capture systems. The control signal is computed in a close-loop at a rate of 100 Hz.

For the system evaluation, we predefined 10 different initial configurations covering the whole workspace. The robot holds a glass with 4 balls and we measured the number of balls that enter the pot after executing the trajectory. We considered 3 scenarios: normal execution, physical disturbance, and target modification.

Looking at the results in Fig. 8, it is clear that the robot achieves a very robust performance. In the normal execution, it pours almost all the balls in the pot, given any initial configuration. This result shows the generalization properties of our model: the robot was initialized in a position that does not belong to the demonstration set, but was able to solve the task. We also tested the system under heavy physical disturbances, including pushing and holding the robot. In this scenario, the performance decays, but the robot was able to succeed most of the time. Finally, we observe the vector field was able to properly adapt to different pot positions. The robot succeeded to put almost all the balls in the pot except for some target positions that were beyond the workspace limits of the robot.

## VI. DISCUSSION & CONCLUSIONS

We have proposed a novel Motion Primitive model that can learn stable vector fields on Lie Groups from human demonstrations. Our work extends previous works on modeling stable vector fields to represent them on Lie Groups. The proposed model allows us to generate reactive and stable robot motions for the full pose (orientation and position). Through an extensive evaluation phase, we have validated the modeling decisions to guarantee stability and the importance of representing the vector fields on Lie Groups to properly solve robot tasks.

We have many directions to improve our model. First, the chosen diffeomorphic function  $\Phi$  has some limitations. Our proposed model cannot set the sink in the antipodal points, given the map in antipodal points is an identity map. In practice, we can set the attractor in an arbitrary pose by adding a linear transformation that moves the sink. Nevertheless, we consider that this limitation might influence the performance when modeling complex motion skills with significant changes in orientation. In the future, we aim to explore novel functions to represent the diffeomorphism  $\Phi$ . The experiments we have carried out focus on the performance evaluation of our proposed stable vector

fields. However, these models are of particular interest combined with additional motion skills, such as obstacle avoidance or joint limit avoidance vector fields, as done in RMP [19] or Composable Energy Policies (CEP) [37]. We will investigate how to combine vector fields in future works.

Another possibility is to use the proposed method as a cost function. Indeed, the architecture encodes in itself a Lyapunov-stable potential function. We can use this function as a terminal cost function (value function) or as a cost function in trajectory optimization problems, allowing the integration of additional cost functions. This approach could be beneficial in long-horizon planning problems [38].

## REFERENCES

- [1] S. Schaal, "Learning from demonstration," in *Proc. Adv. Neural Inf. Process. Syst.*, M. C. Mozer, M. Jordan, and T. Petsche, Eds., MIT Press, 1997, vol. 9, pp. 1040–1046.
- [2] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, pp. 1–8.
- [3] S. Schaal, "Dynamic movement primitives—a framework for motor control in humans and humanoid robotics," in *Adaptive Motion of Animals and Machines*. Tokyo, Japan: Springer, 2006, pp. 261–280. [Online]. Available: <http://www.roboticsproceedings.org/rss16/p056.html>
- [4] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Netw.*, vol. 21, no. 4, pp. 642–653, 2008.
- [5] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, Oct. 2011.
- [6] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 2616–2624.
- [7] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intell. Serv. Robot.*, vol. 9, no. 1, pp. 1–29, 2016.
- [8] T. Kulak, J. Silvério, and S. Calinon, "Fourier movement primitives: An approach for learning rhythmic robot skills from demonstrations," in *Proc. Robot.: Sci. Syst.*, 2020. [Online]. Available: <http://www.roboticsproceedings.org/rss16/p056.html>
- [9] J. Yuan, "Closed-loop manipulator control using quaternion feedback," *IEEE J. Robot. Autom.*, vol. 4, no. 4, pp. 434–440, Aug. 1988.
- [10] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 365–371.
- [11] L. Koutas and Z. Doulgeri, "A correct formulation for the orientation dynamic movement primitives for robot control in the cartesian space," in *Proc. Conf. Robot Learn.*, 2020, pp. 293–302.
- [12] A. Ude, B. Nemec, T. Petric, and J. Morimoto, "Orientation in cartesian space dynamic movement primitives," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 2997–3004.
- [13] Y. Huang, L. Roza, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *Int. J. Robot. Res.*, vol. 38, no. 7, pp. 833–852, 2019.
- [14] Y. Huang, F. J. Abu-Dakka, J. Silvério, and D. G. Caldwell, "Toward orientation learning and adaptation in cartesian space," *IEEE Trans. Robot.*, vol. 37, no. 1, pp. 82–98, Feb. 2021.
- [15] M. J. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, and D. G. Caldwell, "An approach for imitation learning on riemannian manifolds," *IEEE Robot. Automat. Lett.*, vol. 2, no. 3, pp. 1240–1247, Jul. 2017.
- [16] L. Roza and V. Dave, "Orientation probabilistic movement primitives on riemannian manifolds," in *Proc. Conf. Robot Learn.*, 2022, pp. 373–383.
- [17] K. Neumann and J. J. Steil, "Learning robot motions with stable dynamical systems under diffeomorphic transformations," *Robot. Auton. Syst.*, vol. 70, pp. 1–15, 2015.
- [18] J. Urain, M. Ginesi, D. Tateo, and J. Peters, "Imitationflows: Learning deep stable stochastic dynamic systems by normalizing flows," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5231–5237.
- [19] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian motion policies," 2018, *arXiv:1801.02854*.
- [20] M. A. Rana, A. Li, D. Fox, B. Boots, F. Ramos, and N. Ratliff, "Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems," in *Proc. Learn. Dyn. Control*, 2020, pp. 630–639.
- [21] J. M. Lee, *Introduction to Smooth Manifolds*. New York, NY, USA: Springer-Verlag, 2006.
- [22] K. Neumann, A. Lemme, and J. J. Steil, "Neural learning of stable dynamical systems based on data-driven Lyapunov candidates," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 1216–1222.
- [23] N. Perrin and P. Schlehuber-Caissier, "Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems," *Syst. Control Lett.*, vol. 96, pp. 51–59, 2016.
- [24] J. Umlauf and S. Hirche, "Learning stable stochastic nonlinear dynamical systems," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3502–3510.
- [25] D. J. Rezende et al., "Normalizing flows on tori and spheres," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 8083–8092.
- [26] E. Mathieu and M. Nickel, "Riemannian continuous normalizing flows," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 2503–2515, 2020.
- [27] M. C. Gemici, D. Jimenez Rezende, and S. Mohamed, "Normalizing flows on Riemannian manifolds," in *Proc. NeurIPS Workshop Bayesian Deep Learn.*, 2016.
- [28] A. Lou et al., "Neural manifold ordinary differential equations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 17548–17558.
- [29] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 6572–6583.
- [30] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, and D. Duvenaud, "FFJORD: Free-form continuous dynamics for scalable reversible generative models," in *Proc. Int. Conf. Learn. Representations*, 2019.
- [31] L. Falorsi, P. de Haan, T. Davidson, and P. Forré, "Reparameterizing distributions on lie groups," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2019, pp. 3244–3253.
- [32] J. Sola, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," 2018, *arXiv:1812.01537*.
- [33] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proc. Int. Conf. Mach. Learn.*, 2015, vol. 37, pp. 1530–1538.
- [34] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE Int. Conf. Robot. Automat.*, 2000, pp. 995–1001.
- [35] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE J. Robot. Autom.*, vol. 3, no. 1, pp. 43–53, Feb. 1987.
- [36] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," in *Proc. Int. Conf. Learn. Representations*, 2017.
- [37] J. Urain, A. Li, P. Liu, C. D'Eramo, and J. Peters, "Composable energy policies for reactive motion generation and reinforcement learning," in *Robot.: Sci. Syst.*, 2021.
- [38] A. Lambert, A. T. Le, J. Urain, G. Chalkatzaki, B. Boots, and J. Peters, "Learning implicit priors for motion optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022.