Saveriano, Matteo; Abu-Dakka, Fares J.; Kyrki, Ville

Learning stable robotic skills on Riemannian manifolds

Published in:
Robotics and Autonomous Systems

DOI:
10.1016/j.robot.2023.104510

Published: 01/11/2023

Document Version
Publisher's PDF, also known as Version of record

Robotics and Autonomous Systems 169 (2023) 104510

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Learning stable robotic skills on Riemannian manifolds

Matteo Saveriano [a,*], Fares J. Abu-Dakka [b], Ville Kyrki [c]

[a] Automatic Control Lab, Department of Industrial Engineering, University of Trento, Trento, Italy
[b] Munich Institute of Robotics and Machine Intelligence (MIRMI), Technical University of Munich, Germany
[c] Intelligent Robotics Group, Department of Electrical Engineering and Automation, Aalto University, Finland

ARTICLE INFO

Article history:
Received 31 August 2022
Received in revised form 11 July 2023
Accepted 14 August 2023
Available online 19 August 2023

Keywords:
Learning from Demonstration
Learning stable dynamical systems
Riemannian manifold learning

ABSTRACT

In this paper, we propose an approach to learn stable dynamical systems that evolve on Riemannian manifolds. Our approach leverages a data-efficient procedure to learn a diffeomorphic transformation, enabling the mapping of simple stable dynamical systems onto complex robotic skills. By harnessing mathematical techniques derived from differential geometry, our method guarantees that the learned skills fulfill the geometric constraints imposed by the underlying manifolds, such as unit quaternions (UQ) for orientation and symmetric positive definite (SPD) matrices for impedance. Additionally, the method preserves convergence towards a given target. Initially, the proposed methodology is evaluated through simulation on a widely recognized benchmark, which involves projecting Cartesian data onto UQ and SPD manifolds. The performance of our proposed approach is then compared with existing methodologies. Apart from that, a series of experiments were performed to evaluate the proposed approach in real-world scenarios. These experiments involved a physical robot tasked with bottle stacking under various conditions and a drilling task performed in collaboration with a human operator. The evaluation results demonstrate encouraging outcomes in terms of learning accuracy and the ability to adapt to different situations.

1. Introduction

Robots that successfully operate in smart manufacturing have to be capable of precisely controlling their behavior in terms of movements and physical interactions [1]. In this regard, modern industrial and service robots need flexible representations of such intended behaviors in terms of motion, impedance, and force skills (see Fig. 1). The development of such representations is a key aspect in speeding up the integration of robotic solutions in social and industrial environments.

Learning approaches have the possibility to unlock the full potential of smart robotic solutions. Among the others, the Learning from Demonstration (LfD) paradigm [2] aims at developing learning solutions that allow the robot to enrich its skills via human guidance. Among the several existing approaches [3,4], the idea of encoding robotic skills into stable Dynamical Systems (DSs) has gained interest in the LfD community [5–12]. In this context, a robotic skill is any robot motion, either with a given start and goal (point-to-point or discrete motion), or periodic. A linear point-to-point motion is an example of *simple* robotic skill that can be generated, for instance, with a linear DS. Point-to-point motions with an arbitrarily complex path connecting start and goal points are examples of *complex* robotic skills that a non-linear DS can effectively generate. To learn complex robotic skills, DS-based approaches for LfD assume that the demonstrations of a robotic skill are observations of the time evolution of a DS. DSs are flexible motion generators that allow, for example, to encode periodic and discrete motions [13], to reactively avoid possible collisions [14–16], or to update the underlying dynamics in an incremental fashion [17,18].

Although DS-based representations have several interesting properties, most works assume that the data are observations of Euclidean space. However, this is not always the case in robotics where data can belong to a non-Euclidean space. Typical examples of non-Euclidean data are orientation trajectories represented as rotation matrices and Unit Quaternions (UQs), and Symmetric Positive Definite (SPD) matrices for quantities such as inertia, impedance gains, and manipulability, which all belong to Riemannian manifolds. Applying standard arithmetic tools from Euclidean geometry on Riemannian manifolds leads to inaccuracies and incorrect representations, which can be avoided if the proper mathematical tools developed for Riemannian manifolds are used [19].

In this paper, we propose the following.

- A novel geometry-aware approach to encode demonstrations evolving on a Riemannian manifold into a stable dynamical system (*SDS-RM*).

* Corresponding author.
E-mail addresses: matteo.saveriano@unitn.it (M. Saveriano), fares.abu-dakka@tum.de (F.J. Abu-Dakka), ville.kyrki@aalto.fi (V. Kyrki).

https://doi.org/10.1016/j.robot.2023.104510

Table 1
Comparison among state-of-the-art approaches for LfD on Riemannian manifolds and SDS-RM.

	End-Point	Multiple demos	Multimodal	Time-independent	Data-efficiency	Accuracy[+]	Training time
DMP [20,21]	✓	–	–	–	high	medium	low
R-GMM [22]	✓	✓	–	–	high	medium	low
FDM[a] [9]	✓	–	–	✓	high	medium	low
E-FLOW [23]	✓	✓	✓	✓	low	medium[b]	high
I-FLOW [24–26]	✓	✓	✓	✓	low	medium[b]	high
SDS-RM (ours)	✓	✓	✓	✓	high	high	low

[a]FDM is not designed to work on Riemannian manifolds. However, our formulation allows us to use them to learn a diffeomorphism in TS.
[b]E-FLOW and I-FLOW need a hyperparameter search to reach high accuracy. However, performing a hyperparameters search requires a GPU cluster and it is beyond the scope of this paper. With trial and error, we found a hyperparameter configuration that gives a medium accuracy.

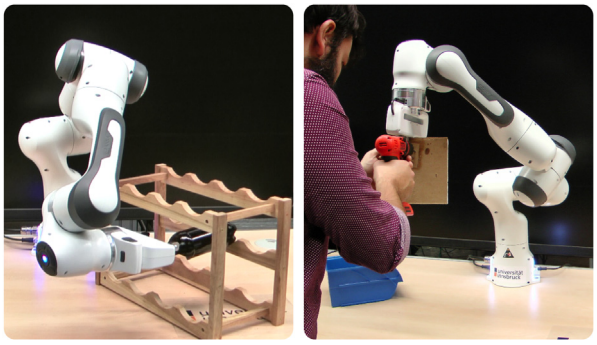


Fig. 1. Experimental setups involving a robot (Franka Emika Panda) and Riemannian data. (Left) The robot stacks a bottle on a rack. This task requires adaptation of position and orientation (UQ). (Right) The robot performs collaborative drilling with a human. This task exploits stiffness (SPD matrix) modulation.

- Mathematical foundations for SDS-RM to work in any Riemannian manifold. In the paper, we provide two manifolds as case studies: (*i*) UQ, and (*ii*) SPD manifolds.
- A data-efficient approach, based on Gaussian Mixture Models (GMMs), to learn diffeomorphisms on Riemannian manifolds.
- An extension of the LASA handwriting dataset [6], a popular benchmark in DS-based LfD, to generate UQ and SPD trajectories. The resulting *Riemannian LASA* dataset, publicly available at gitlab.com/geometry-aware/riemannian-lasa-dataset, will serve as a benchmark for a quantitative comparison of newly developed approaches.

To this end, SDS-RM leverages the concept of diffeomorphism (a bijective, continuous, and continuously differentiable mapping with a continuous and continuously differentiable inverse), to transform simple and stable (base) dynamics into complex robotic skills. Building on tools from Riemannian geometry, we first present rigorous stability proofs for the base and the diffeomorphic DSs. We then present a data-efficient approach that leverages a GMM [27] to learn the diffeomorphic mapping from linear motion to an arbitrarily complex demonstration, that is, a demonstration of a point-to-point motion on a Riemannian manifold with a possibly nonlinear path connecting start and goal points. As a result, we obtain a DS that accurately represents data evolving on Riemannian manifolds and that preserves the convergence to a given target as well as the geometric structure of the data.

The rest of the paper is organized as follows. Section 2 presents the related literature. Basic concepts of Riemannian geometry are given in Section 3. Section 4 provides the theoretical foundations of SDS-RM. In Section 5, we present an approach to learn stable skills via diffeomorphic maps. SDS-RM is evaluated on a public benchmark and compared against a state-of-the-art approach in Section 6. Experiments on a real robot (Franka Emika Panda) are presented in Section 7. Section 8 states the conclusion and proposes further research directions.

2. Related works

Classification on Riemannian manifolds is a well-studied problem in computer vision. In this regard, [28] proposes a distance measure for SPD matrices and exploits it in a similarity-based algorithm used for image classification. In order to recognize faces in videos, Huang et al. [29] propose a metric learning approach that learns a metric between Euclidean and Riemannian input spaces. Chakraborty et al. [30] extend the convolution operation to Riemannian manifolds using the weighted Frechét mean. In general, approaches that attempt to adapt deep neural networks to non-Euclidean fall under the umbrella of geometric deep learning [31].

However, LfD is a regression problem that received less in the literature on manifold learning. Therefore, in the rest of this section, we discuss LfD both in general and with a specific focus on DS-based and geometry-aware learning approaches. For a clear comparison, the main features of SDS-RM and of existing approaches are summarized in Table 1.

Learning from Demonstration (LfD) provides a user-friendly framework that allows non-roboticists to teach robots and enables robots to autonomously perform new tasks based on that human demonstration [2]. Over the last couple of decades, several LfD approaches have been developed [3,4]. LfD approaches can be categorized into two main groups depending on the underlying learning strategy: (*i*) deterministic approaches that try to reproduce the demonstrations with a function approximator, e.g., a neural network [32,33], and (*ii*) probabilistic approaches that learn a probability distribution from the demonstrations. Examples of this category include GMM [34], Task-Parameterized GMM (TP-GMM) [35], Probabilistic Movement Primitives (ProMP) [36], and Kernelized Movement Primitive (KMP) [37].

In both groups, it is possible to learn and retrieve a *static* or a *dynamic* mapping between input and output. In a static mapping the current input, e.g., the time, is mapped into the desired output, e.g., the robot's joint angles, while a DS maps the input, e.g., the robot joint angles, into its time derivative(s), i.e., the joint velocities. In this paper, we focus on learning stable DSs from data belonging to Riemannian manifolds. However, in order to provide a more comprehensive review of the existing literature, we also highlight some recent Riemannian-based approaches that learn static mappings. Zeestraten et al. [38] exploited Riemannian metrics to learn orientation trajectories with TP-GMM. Huang et al. [37] proposed to train KMP in the tangent space of unit quaternion trajectories. Kinaesthetic is used to estimate full stiffness matrices of an interaction task, which is subsequently used to learn force-based variable impedance profiles using Riemannian metric-based GMM/Gaussian Mixture Regression (GMR) [39]. Later the authors proposed to train KMP on the tangent space of SPD matrices [40]. Jaquier et al. [41] formulated a tensor-based GMM/GMR on SPD manifold, which

later has been exploited in manipulability transfer and tracking problem [42]. Calinon [22] extended the GMM formulation to a variety of manifolds including UQ and SPD. We name this approach Riemannian GMM (R-GMM). R-GMM is a prominent approach to performing learning from multiple demonstrations on Riemannian manifolds and we compare its performance against SDS-RM in Section 6.

Encoding manipulation skills into stable DSs is achieved by leveraging time-dependent or time-independent DSs. Dynamic Movement Primitives (DMPs) [5] are a prominent approach to encoding robotic skills into time-dependent representations. The classical DMP formulation has been extended in different ways [43]. Among the others, extensions to Riemannian manifolds are relevant for this work. Abu-Dakka et al. extend classical DMPs to encode discrete [44] and periodic [45] unit quaternion trajectories, while the work in [20] also considers rotation matrices. The stability of orientation DMPs is shown in [46]. In [21], DMPs are reformulated to generate discrete SPD profiles. DMPs exploit a time-driven forcing term, learned from a single demonstration, to accurately reproduce the desired skill. The advantage of having a time-driven forcing term is that the learning scale well to high-dimensional spaces—as the learned term only depends on a scalar input. The disadvantage is that the generated motion is forced to follow the stereotypical (i.e., demonstrated) one and generalizes poorly outside the demonstration area [8]. Alternative approaches [18,47] learn a state-depended forcing term, but they still use a vanishing time signal to suppress the forcing term and retrieve asymptotic stability. The tuning of this vanishing signal affects the reproduction accuracy. Moreover, time-dependent DSs generate time indexed trajectory that may become unfeasible if an external perturbation distracts the robot from the trajectory. On the contrary, time-independent DSs generate the next point based on the current value of the input and are robust to external perturbations.[1]

The Stable Estimator of Dynamical Systems (SEDS) [6] is one of the first approaches that learn stable and autonomous DSs. It exploits Lyapunov theory to derive stability constraints for a GMR-DS. The main limitations of SEDS are the relatively long training time and the reduced accuracy on complex motions. The loss of accuracy is caused by the stability constraints, which has been called the *accuracy vs stability dilemma* [8]. To alleviate this issue, the approach in [11] derives weak stability constraints for a neural network-based DS. Contraction theory [48] is used in [10] to derive stability conditions for a GMR-DS in Euclidean space, and in [49] to encode stable UQ orientations. The extension of SEDS in [7] significantly reduces inaccuracies and training time by separately learning a (possibly) unstable DS and a stabilizing control input.

Alternative approaches leverage a diffeomorphic mapping between Euclidean spaces to accurately fit the demonstrations while preserving stability. Neumann et al. [8] learn a diffeomorphism to map the demonstrations into a space where quadratic stability constraints introduce negligible deformations. The approach is effective but needs a long training time, as experimentally shown in [47]. In a similar direction, Euclideanizing Flows (E-FLOW) [23] fits a diffeomorphism that linearizes the demonstrations as if they were generated by a linear DS. The opposite idea, i.e., transform straight lines into complex motions, is exploited Fast Diffeomorphic Matching (FDM) [9] and by Imitation Flow (I-FLOW) [24,25]. FDM uses a composition of locally weighted (with an exponential kernel) translations to rapidly fit a diffeomorphism from a single demonstration. SDS-RM builds on similar ideas, but it works on Riemannian data and with multiple demonstrations. FDM is representative of a class of LfD approaches (including DMP) that learn from a single demonstration and it is interesting to compare its performance against SDS-RM. This experimental comparison is conducted in Section 6.

[1] The reader is referred to [5,43] for a thorough discussion on advantages and disadvantages of time-dependent and time-independent DS representations.

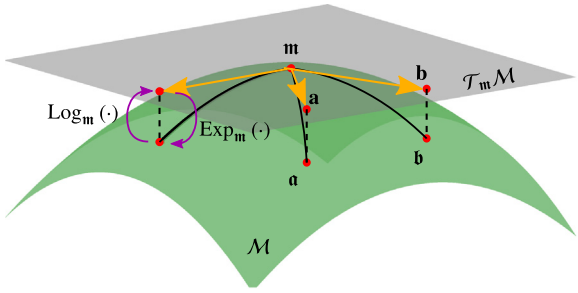


Fig. 2. A Riemannian manifold $\mathcal{M}$ (green surface) and its tangent space $\mathcal{T}_{\mathfrak{m}}\mathcal{M}$ (gray plane) centered at $\mathfrak{m}$. The logarithmic $\mathrm{Log}_{\mathfrak{m}}(\cdot)$ and exponential $\mathrm{Exp}_{\mathfrak{m}}(\cdot)$ maps move points from the manifold ($\mathfrak{a}$ and $\mathfrak{b}$) to the tangent space ($\mathbf{a}$ and $\mathbf{b}$) and vice-versa. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

E-FLOW and I-FLOW exploit deep invertible neural networks to learn a diffeomorphic mapping. Invertible neural networks are slightly more accurate than classical methods, as experimentally shown in [24]. However, as shown in [26] and in the experimental comparison carried out in Section 6, approaches based on deep networks require long training time and intense hyperparameters search, which represents a limitation in typical LfD settings. Therefore, SDS-RM leverages a mature and data-efficient approach in LfD (GMM/GMR) to fit a diffeomorphism on a manifold. In our previous work [26], we extend I-FLOW to Riemannian manifolds by projecting training data in a single tangent space. However, using a single tangent space is known to introduce approximation errors [22]. Moreover, a formal stability proof is missing in [26]. In this respect, SDS-RM builds on rigorous stability analysis and uses a moving tangent space to project the data, which minimizes inaccuracies as experimentally shown in Section 5.4.

Finally, the Riemannian Motion Policy (RMP) framework [50,51] consists of closed-loop controllers embedded in a second-order dynamics that exploits the Riemannian metric to optimally track a reference motion. In this respect, RMP and SDS-RM are complementary: SDS-RM can be used to generate the desired motion and RMP to optimally track it.

3. Background

A Riemannian manifold $\mathcal{M}$ is a smooth differentiable topological space, for each point of which $\mathfrak{m} \in \mathcal{M}$, it is possible to compute a Tangent Space (TS) $\mathcal{T}_{\mathfrak{m}}\mathcal{M}$ (see Fig. 2). The TS is equipped with a positive definite inner product $\langle \mathbf{a}, \mathbf{b} \rangle_{\mathfrak{m}} \in \mathbb{R}$, where $\mathbf{a}, \mathbf{b} \in \mathcal{T}_{\mathfrak{m}}\mathcal{M}$ and $\mathfrak{m} \in \mathcal{M}$ is the point where the TS is computed. The inner product allows the definition of the notion of distance on the manifold. Depending on the Riemannian manifold, points on the TS can be vectors, matrices, or more complex mathematical objects. For example, a TS of UQs consists of 3D vectors, while a TS of SPD matrices consists of symmetric matrices. In this work, we indicate a point on the TS using bold capital letters, i.e., $\mathbf{a} \in \mathcal{T}_{\mathfrak{m}}\mathcal{M}$.

The operator that transforms points from a Riemannian manifold to its tangent space is the *logarithmic map* $\mathrm{Log}_{\mathfrak{m}}(\cdot) : \mathcal{M} \rightarrow \mathcal{T}_{\mathfrak{m}}\mathcal{M}$. Its inverse operator is called the *exponential map* $\mathrm{Exp}_{\mathfrak{m}}(\cdot) : \mathcal{T}_{\mathfrak{m}}\mathcal{M} \rightarrow \mathcal{M}$. The *parallel transport* $\mathcal{T}_{\mathfrak{m}_1 \rightarrow \mathfrak{m}_2} : \mathcal{T}_{\mathfrak{m}_1}\mathcal{M} \rightarrow \mathcal{T}_{\mathfrak{m}_2}\mathcal{M}$ moves elements between tangent spaces while preserving their angle.

Table 2
Re-interpretation of basic standard operations in a Riemannian manifold [53].

	Euclidean space	Riemannian manifold
Subtraction	$\overrightarrow{\mathbf{ma}} = \mathbf{a} - \mathbf{m}$	$\overrightarrow{\mathfrak{m}\mathfrak{a}} = \text{Log}_{\mathfrak{m}}(\mathfrak{a})$
Addition	$\mathbf{p} = \mathbf{m} + \overrightarrow{\mathbf{ma}}$	$\mathfrak{a} = \text{Exp}_{\mathfrak{m}}(\overrightarrow{\mathfrak{m}\mathfrak{a}})$
Distance	$\text{dist}(\mathbf{m}, \mathbf{a}) = \|\mathbf{a} - \mathbf{m}\|$	$\text{dist}(\mathfrak{m}, \mathfrak{a}) = \|\overrightarrow{\mathfrak{m}\mathfrak{p}}\|_{\mathfrak{m}}$
Interpolation	$\mathbf{m}(t) = \mathbf{m}_1 + t\overrightarrow{\mathbf{m}_1\mathbf{m}_2}$	$\mathfrak{m}(t) = \text{Exp}_{\mathfrak{m}_1}(t\overrightarrow{\mathfrak{m}_1\mathfrak{m}_2})$

In case of UQ manifold, a point on the manifold is $\mathfrak{a} = \nu + \mathbf{u}$, the distance between two UQs $\mathfrak{a}_1$ and $\mathfrak{a}_2$ is $d(\mathfrak{a}_1, \mathfrak{a}_2) = \arccos(\mathfrak{a}_2{}^{\top}\mathfrak{a}_1)$, $\bar{\mathfrak{a}} = \nu - \mathbf{u}$ is the conjugate of $\mathfrak{a}$, and $\mathfrak{a}_1 * \mathfrak{a}_2$ indicates the quaternion product. Given this, we can define [20]

$$\text{Exp}_{\mathfrak{m}}(\mathbf{a}) = \begin{cases} \left[\cos(\|\mathbf{a}\|) + \sin(\|\mathbf{a}\|)\frac{\mathbf{a}}{\|\mathbf{a}\|}\right] * \mathfrak{m}, & \|\mathbf{a}\| \neq 0 \\ \left[1 + [0\ 0\ 0]^{\top}\right] * \mathfrak{m}, & \text{otherwise,} \end{cases} \tag{1}$$

$$\text{Log}_{\mathfrak{m}}(\mathfrak{a}) = \text{Log}(\mathfrak{a} * \bar{\mathfrak{m}}) = \begin{cases} \arccos(\nu)\frac{\mathbf{u}}{\|\mathbf{u}\|}, & \|\mathbf{u}\| \neq 0 \\ [0\ 0\ 0]^{\top}, & \text{otherwise.} \end{cases} \tag{2}$$

Note that the mappings in (1)–(2) are derived from Lie theory [52], which is widely used in robotics. In this formulation, the mappings are defined in the Lie algebra (the TS at 1 + [0 0 0]), and the quaternion product is used to parallel transport the vectors from the Lie algebra to the TS placed at a different point (e.g., $\mathfrak{m}$).

In case of SPD manifold, the operators are defined as in [53,54]

$$\text{Exp}_{\mathfrak{m}}(\mathbf{a}) = \mathfrak{m}^{\frac{1}{2}}\text{expm}\left(\mathfrak{m}^{-\frac{1}{2}}\mathbf{a}\mathfrak{m}^{-\frac{1}{2}}\right)\mathfrak{m}^{\frac{1}{2}}, \tag{3}$$

$$\text{Log}_{\mathfrak{m}}(\mathfrak{a}) = \mathfrak{m}^{\frac{1}{2}}\text{logm}\left(\mathfrak{m}^{-\frac{1}{2}}\mathfrak{a}\mathfrak{m}^{-\frac{1}{2}}\right)\mathfrak{m}^{\frac{1}{2}}, \tag{4}$$

$$\mathcal{T}_{\mathfrak{m}_1 \to \mathfrak{m}_2}(\mathbf{a}) = \mathfrak{m}_2^{\frac{1}{2}}\mathfrak{m}_1^{\frac{1}{2}}\,\mathbf{a}\,\mathfrak{m}_1^{\frac{1}{2}}\mathfrak{m}_2^{\frac{1}{2}}, \tag{5}$$

where expm(·) and logm(·) are the matrix exponential and logarithm functions respectively. Here, the affine-invariant distance [53] is used to compute $d(\mathfrak{a}, \mathfrak{m})$

$$d(\mathfrak{a}, \mathfrak{m}) = \left\|\text{logm}\left(\mathfrak{m}^{-\frac{1}{2}}\mathfrak{a}\mathfrak{m}^{-\frac{1}{2}}\right)\right\|_F, \tag{6}$$

where $\|\cdot\|_F$ is the Frobenius norm.

Other basic operations on manifolds can be computed as shown in Table 2.

4. Diffeomorphic DS on manifolds

Inspired by [23], we apply a diffeomorphism to a stable (base) DS evolving on the Riemannian manifold (see Fig. 3). The diffeomorphism is learned from a set of demonstrations and deforms the trajectories of the base DS to accurately match the demonstrations. In order to effectively apply this idea, we first need to design a stable DS evolving on the Riemannian manifold. This DS provides the basic motion that connects the initial point to the desired goal. Second, we need to show that a diffeomorphic transformation preserves the stability of the base DS. This result, known for the Euclidean space, extends to Riemannian manifolds as shown in this section.

4.1. Base DS on Riemannian manifolds

In Euclidean space, a linear DS in the form

$$\dot{\mathbf{x}} = -k_{\mathbf{x}}(\mathbf{x} - \mathbf{g}) \tag{7}$$

is often used as base[2] DS. Indeed, if the gain $k_{\mathbf{x}} > 0$, the linear DS in (7) is globally exponentially stable [55]. This implies that the linear DS generates trajectories connecting any initial point $\mathbf{x}(0) \in \mathbb{R}^n$ with any goal $\mathbf{g} \in \mathbb{R}^n$.

[2] Note that some authors assume, without loss of generality, that $\mathbf{g} = \mathbf{0}$.

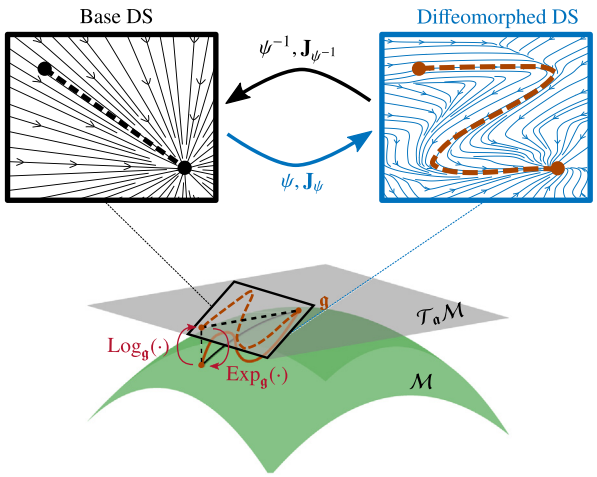


Fig. 3. The idea of SDS-RM is to learn a diffeomorphic mapping between TSs to transform the trajectories of a base DS into complex motions. Motion on the TS is projected back and forth to the underlying Riemannian manifold using exponential and logarithmic maps respectively.

We seek an "equivalent" of the stable linear dynamics for Riemannian manifolds. In this work, the base DS is the non-linear dynamics

$$\dot{\mathfrak{a}} = k_{\mathfrak{a}}\mathbf{a} = k_{\mathfrak{a}}\text{Log}_{\mathfrak{a}}(\mathfrak{g}) \quad \in \mathcal{T}_{\mathfrak{a}}\mathcal{M}, \tag{8}$$

where the logarithmic map is introduced in Section 3 and its expression depends on the considered Riemannian manifold. The non-linear DS in (8), called a *geodesic* DS, shares similarities with the linear DS in (7) and, for this reason, it is used in this work as base DS. Indeed, similarly to the term $(\mathbf{x} - \mathbf{g})$ in (7), the logarithmic map in (8) represents the displacement between the current point $\mathfrak{a} \in \mathcal{M}$ and the goal $\mathfrak{g} \in \mathcal{M}$. It is worth mentioning that in (8) we consider the TS at the current state $\mathfrak{a}$ instead of a fixed TS at the goal $\mathfrak{g}$. As discussed in [22], working in a single TS is inaccurate and introduces severe deformations in the learned trajectory. On the contrary, in our formulation we always place the TS at the current point, therefore minimizing the distortion.

As stated in Theorem 1, The base DS in (8) is also stable if the gain $k_{\mathfrak{a}} > 0$. This implies that the base DS in (8) generates trajectories connecting any initial point $\mathfrak{a}(0)$ with any goal $\mathfrak{g}$ (see Fig. 4). Stability results are formally stated in Theorem 1 (asymptotic stability) and Remark 3 (exponential stability).

Theorem 1. *Let the gain $k_{\mathfrak{a}} > 0$. Assume also that $\text{Log}_{\mathfrak{g}}(\mathfrak{a}) = -\mathcal{T}_{\mathfrak{g}\to\mathfrak{a}}\text{Log}_{\mathfrak{a}}(\mathfrak{g})$, where $\mathcal{T}_{\mathfrak{g}\to\mathfrak{a}}$ is the parallel transport from $\mathfrak{g}$ to $\mathfrak{a}$. Under these assumptions, the DS in (8) has a globally (in its domain of definition) asymptotically stable equilibrium at $\mathfrak{g}$.*

Proof. Let us express the velocity field (8) in $\mathfrak{g}$ using parallel transport. By assumption, it holds that

$$\mathcal{T}_{\mathfrak{g}\to\mathfrak{a}}\dot{\mathfrak{a}} = -k_{\mathfrak{a}}\text{Log}_{\mathfrak{g}}(\mathfrak{a}) \quad \in \mathcal{T}_{\mathfrak{g}}\mathcal{M}. \tag{9}$$

To prove the stability of (9), one can define the Lyapunov candidate function [56]

$$V(\mathfrak{a}) = \langle \mathbf{a}, \mathbf{a} \rangle_{\mathfrak{g}} = \langle \text{Log}_{\mathfrak{g}}(\mathfrak{a}), \text{Log}_{\mathfrak{g}}(\mathfrak{a}) \rangle_{\mathfrak{g}} \tag{10}$$

and follow the same arguments outlined in [42, Section 4.2.2] for SPD matrices. □

Remark 1. The assumption made in Theorem 1 that $\log_{\mathfrak{g}}(\mathfrak{a}) = -\mathcal{T}_{\mathfrak{g}\to\mathfrak{a}}\text{Log}_{\mathfrak{a}}(\mathfrak{g})$ holds for any Riemannian manifold [57, Theorem 6].

Remark 2. The results of Theorem 1 hold where the logarithmic map is uniquely defined, i.e., in a region that does not contain points conjugate to $\mathfrak{g}$ [56]. For SPD matrices, this holds everywhere [53]. Hence, Theorem 1 is globally valid on the manifold of SPD matrices. For unit m-sphere (including UQ), instead, the logarithmic map $\text{Log}_{\mathfrak{a}}(\cdot)$ is defined everywhere apart from the antipodal point $-\mathfrak{a}$ [58].

Even if it is not strictly required in our approach, it is interesting to show that, like the linear DS in (7), the DS in (8) is exponentially stable.

Remark 3. Under the assumptions of Theorem 1, the DS in (8) has a globally (in its domain of definition) exponentially stable equilibrium at $\mathfrak{g}$ [42, Section 4.2.2].

4.2. Diffeomorphic DS

A diffeomorphic map or a diffeomorphism ψ is a bijective, continuous, and with continuous inverse ψ^{-1} change of coordinates. In this work, we assume that $\psi : \mathcal{T}_{\mathfrak{a}}\mathcal{M} \rightarrow \mathcal{T}_{\mathfrak{a}}\mathcal{M}$, i.e., the diffeomorphism transforms a global coordinate $\mathbf{a} \in \mathcal{T}_{\mathfrak{a}}\mathcal{M}$ into another global coordinate $\mathbf{b} = \psi(\mathbf{a}) \in \mathcal{T}_{\mathfrak{a}}\mathcal{M}$. Further, the diffeomorphism ψ is assumed to be bounded, i.e., it maps bounded vectors into bounded vectors.

In order to match a set of demonstrations, we apply ψ to the base DS in (8). More in detail, let us assume that $\mathbf{a} = \text{Log}_{\mathfrak{a}}(\mathfrak{g})$, and that the dynamics of the global coordinates $\dot{\mathbf{a}}$ is described by (8). By taking the time derivative of $\mathfrak{b}$, we obtain the DS [23]

$$\dot{\mathfrak{b}} = \frac{\partial \psi}{\partial \mathbf{a}} \dot{\mathbf{a}} = k_{\mathfrak{a}} \mathbf{J}_{\psi}(\psi^{-1}(\mathbf{b}))\psi^{-1}(\mathbf{b}), \tag{11}$$

where $\mathbf{J}_{\psi}(\cdot)$ is the *Jacobian matrix* of ψ evaluated at a particular point and the inverse mapping $\mathbf{a} = \psi^{-1}(\mathbf{b})$ is used to remove the dependency on $\mathbf{a}$. Having assumed that ψ is a bounded diffeomorphism, the right side of (11) satisfies the Lipschitz condition and, therefore, the DS in (11) admits a unique solution. The stability of the *diffeomorphic dynamics* in (11) is stated by the following theorem.

Theorem 2. *The diffeomorphic DS in (11) inherits the stability properties of the base DS in (8). That is if the base DS is globally (in its domain of definition) asymptotically stable so is the diffeomorphic DS.*

Proof. From the proof of Theorem 1, it holds that $V(\mathfrak{a})$ defined in (10) is a Lyapunov function for the base DS in (8). As shown in [23, Section 3.2], the function $V_{\psi}(\mathfrak{b}) = \langle \psi^{-1}(\mathbf{b}), \psi^{-1}(\mathbf{b}) \rangle_{\mathfrak{g}'}$, where $\mathfrak{g}'$ is the point where $\psi^{-1}(\text{Log}_{\mathfrak{a}}(\mathfrak{g})) = \mathbf{0}$, is a valid Lyapunov function for the diffeomorphic DS in (11). □

Remark 4. Theorem 2 states the convergence of the DS (11) to the equilibrium $\mathfrak{g}'$. This point may differ from the equilibrium $\mathfrak{g}$ of the base DS (8). However, in LfD, we are interested in converging to a given goal—let's say $\mathfrak{g}$ for simplicity. Assuming that the inverse mapping $\psi^{-1}(\cdot)$ is identity at the goal, i.e., $\psi^{-1}(Log_{\mathfrak{a}}(\mathfrak{g})) = Log_{\mathfrak{a}}(\mathfrak{g}) = \mathbf{0}$, it is straightforward to show from Theorem 2 that the DS (11) also convergences to $\mathfrak{g}$.

Given the global coordinate $\mathbf{b}$, we compute the corresponding manifold point $\mathfrak{b}$ through the exponential map as

$$\mathfrak{b} = \text{Exp}_{\mathfrak{a}}(\mathbf{b}). \tag{12}$$

Recalling that the exponential map is a local diffeomorphism, the composite mapping $\text{Exp}_{\mathfrak{a}}(\mathbf{b}) = \text{Exp}_{\mathfrak{a}} \circ \psi(\mathbf{a})$ can be considered a diffeomorphism between manifolds.

5. Learning stable skills via diffeomorphisms

The stability theorems provided in Section 4 give solid theoretical foundations to our learning approach. In this section, we describe how to generate training data suitable to learn a diffeomorphic map on manifolds. The approach, as well as the derivations in Section 4, are quite general and allow the use of different approaches to find the sought diffeomorphism. We then describe how GMM/GMR, a consolidated and data-efficient approach for LfD, can be extended to learn such a diffeomorphism. Finally, we discuss how to apply our approach to two popular Riemannian manifolds, namely the unit quaternions and the SPD matrices.

5.1. Data pre-processing

We aim at learning a diffeomorphism $\psi(\cdot)$ that maps the trajectory $\mathfrak{a}(t)$, solution of the base DS in (8), into an arbitrarily complex demonstration. To this end, let us assume the user provides a set of $D \geq 1$ demonstrations each containing L points on a Riemannian manifold. Demonstrations are organized in the set $\mathfrak{B} = \left\{\mathfrak{b}_l^d\right\}_{l=1,d=1}^{L,D}$, where each $\mathfrak{b}_l^d \in \mathcal{M}$. We also assume that the demonstrations converge to the same goal ($\mathfrak{b}_L^1 = \cdots = \mathfrak{b}_L^D = \mathfrak{g}$) and that a sampling time δt is known. When collecting demonstrations using kinesthetic teaching, it is possible to observe some variations in the final point. In this case, we re-scale the trajectories to converge to the same goal, which is defined by the user (e.g., as the average of the end points). It is worth mentioning that, when orientation trajectories are collected from demonstrations with a real robot, it is needed to extract UQs from rotation matrices. This is because the robot's forward kinematics is typically expressed as a homogeneous transformation matrix [59]. While numerically extracting UQs from a sequence of rotation matrices, it can happen that the approach returns a quaternion at time t and its antipodal at $t+1$. This is because antipodal UQs represents the same rotation. To prevent this discontinuity, one can check that the dot product $\mathfrak{q}_t \cdot \mathfrak{q}_{t+1} > 0$, otherwise, replace $\mathfrak{q}_{t+1}$ with $-\mathfrak{q}_{t+1}$.

Given the set of demonstrations $\mathfrak{B}$, we generate a set of D base trajectories by projecting (8) on the manifold. More in detail, we set the initial condition $\mathfrak{a}_1^d = \mathfrak{b}_1^d$ and project the tangent space velocity on the manifold using the exponential map as

$$\mathfrak{a}_{l+1}^d = \text{Exp}_{\mathfrak{a}_l^d}\left(\delta t\, \dot{\mathfrak{a}}_l^d\right) \ \forall l, d \tag{13}$$

The time derivative $\dot{\mathfrak{a}}_l^d$ is defined as in (8), and the exponential/logarithmic maps for UQ and SPD manifolds are defined as in Section 3.

The D base trajectories are organized in a set $\mathfrak{A} = \left\{\mathfrak{a}_l^d\right\}_{l=1,d=1}^{L,D}$. In order to transform the datasets $\mathfrak{A}$ and $\mathfrak{B}$ into suitable training data we proceed as follows. We use the logarithmic map $\mathbf{a}_l^d = \text{Log}_{\mathfrak{a}_l^d}(\mathfrak{g})$, $\forall l, d$ to project the goal $\mathfrak{g}$ in each TS placed at $\mathbf{a}_l^d$. We use the logarithmic map $\mathbf{b}_l^d = \text{Log}_{\mathfrak{a}_l^d}(\mathfrak{b}_l^d)$, $\forall l, d$ to project each point in $\mathfrak{B}$ in the TS placed at $\mathbf{a}_l^d$. As a result, we obtain the sets $\mathcal{A} = \left\{\mathbf{a}_l^d\right\}_{l=1,d=1}^{L,D}$ and $\mathcal{B} = \left\{\mathbf{b}_l^d\right\}_{l=1,d=1}^{L,D}$. In other words, we have in $\mathcal{A}$ the points from the base the DS (8) that exponentially converge towards $\mathfrak{g}$ and in $\mathcal{B}$ their demonstrated values. Note that each $\mathbf{a}_l^d$ and $\mathbf{b}_l^d$ is expressed in the same TS to make them comparable.

After this procedure, the learning problem becomes how to fit a mapping between $\mathcal{A}$ and $\mathcal{B}$ while preserving the stability. Exploiting the theoretical results in Theorem 2 and Remark 4, this learning problem is solved by fitting a diffeomorphism between $\mathcal{A}$ and $\mathcal{B}$. The resulting approach is presented in the rest of this section.

5.2. GMM /GMR-based diffeomorphism

A GMM [27] models the joint probability distribution $p(\cdot)$ between training data as a weighted sum of K Gaussian components $\mathcal{N}(\cdot)$, i.e.,

$$p(\mathbf{a}, \mathbf{b}|\Theta_k) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{a}, \mathbf{b}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{14}$$

where each $\Theta_k = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ contains learning parameters. The K mixing weights π_k satisfy $\sum_{k=1}^{K} \pi_k = 1$, while the means and covariance matrices are defined as

$$\boldsymbol{\mu}_k = \begin{bmatrix} \boldsymbol{\mu}_k^a \\ \boldsymbol{\mu}_k^b \end{bmatrix}, \quad \boldsymbol{\Sigma}_k = \begin{bmatrix} \boldsymbol{\Sigma}_k^{aa} & \boldsymbol{\Sigma}_k^{ab} \\ \boldsymbol{\Sigma}_k^{ba} & \boldsymbol{\Sigma}_k^{bb} \end{bmatrix}. \tag{15}$$

As shown in [13] for periodic DSs in Euclidean space, we can use conditioning and expectation on the joint distribution in (14) to compute the mapping $\psi(\mathbf{a})$ and its inverse $\psi^{-1}(\mathbf{b})$. The sought mappings $\psi(\mathbf{a}) = \mathbb{E}[p(\mathbf{b}|\mathbf{a})]$ and $\psi^{-1}(\mathbf{b}) = \mathbb{E}[p(\mathbf{a}|\mathbf{b})]$ are computed in closed-form using GMR [27,60] as:

$$\begin{aligned} \psi(\mathbf{a}) &= \sum_{k=1}^{K} h_k(\mathbf{a}) \left(\boldsymbol{\mu}_k^b \right. \\ &\quad \left. + \boldsymbol{\Sigma}_k^{ba} (\boldsymbol{\Sigma}_k^{aa})^{-1} (\mathbf{a} - \boldsymbol{\mu}_k^a) \right), \\ h_k(\mathbf{a}) &= \frac{\pi_k \mathcal{N}(\mathbf{a}|\boldsymbol{\mu}_k^a, \boldsymbol{\Sigma}_k^{aa})}{\sum_{i=1}^{K} \pi_i \mathcal{N}(\mathbf{a}|\boldsymbol{\mu}_i^a, \boldsymbol{\Sigma}_i^{aa})}, \end{aligned} \tag{16}$$

and

$$\begin{aligned} \psi^{-1}(\mathbf{b}) &= \sum_{k=1}^{K} h_k(\mathbf{b}) \left(\boldsymbol{\mu}_k^a \right. \\ &\quad \left. + \boldsymbol{\Sigma}_k^{ab} (\boldsymbol{\Sigma}_k^{bb})^{-1} (\mathbf{b} - \boldsymbol{\mu}_k^b) \right), \\ h_k(\mathbf{b}) &= \frac{\pi_k \mathcal{N}(\mathbf{b}|\boldsymbol{\mu}_k^b, \boldsymbol{\Sigma}_k^{bb})}{\sum_{i=1}^{K} \pi_i \mathcal{N}(\mathbf{b}|\boldsymbol{\mu}_i^b, \boldsymbol{\Sigma}_i^{bb})}. \end{aligned} \tag{17}$$

It is worth noticing that since both $\psi(\mathbf{a})$ and its inverse $\psi^{-1}(\mathbf{b})$ exist and are differentiable, $\psi(\mathbf{a})$ is a diffeomorphism.

In order to build the DS in (11), we need to compute the Jacobian matrix $\mathbf{J}_\psi(\mathbf{a})$ which has the closed-form expression given in (18). For completeness, we provide the full derivation of $\mathbf{J}_\psi(\mathbf{a})$ in Appendix A. Note that the term $\hat{\psi}(\mathbf{a})$ in (18) is already computed in (16) and can be reused to speed up the computation of the Jacobian.

$$\begin{aligned} \mathbf{J}_\psi(\mathbf{a}) &= \sum_{k=1}^{K} h_k(\mathbf{a}) \left[\boldsymbol{\Sigma}_k^{ba} (\boldsymbol{\Sigma}_k^{aa})^{-1} + \right. \\ &\quad \left(\sum_{i=1}^{K} h_i(\mathbf{a}) \left(\boldsymbol{\Sigma}_i^{aa} \right)^{-1} (\mathbf{a} - \boldsymbol{\mu}_i^a) - \right. \\ &\quad \left. \left. h_k(\mathbf{a}) \left(\boldsymbol{\Sigma}_k^{aa} \right)^{-1} (\mathbf{a} - \boldsymbol{\mu}_k^a) \right) \hat{\psi}(\mathbf{a})^\top \right], \\ \hat{\psi}(\mathbf{a}) &= \boldsymbol{\mu}_k^b + \boldsymbol{\Sigma}_k^{ba} (\boldsymbol{\Sigma}_k^{aa})^{-1} (\mathbf{a} - \boldsymbol{\mu}_k^a). \end{aligned} \tag{18}$$

5.3. Point-to-point motion on Riemannian manifolds

The GMM/GMR-based diffeomorphism presented in the previous section does not explicitly consider that we aim at reproducing discrete motions, i.e., motions with a specific initial and final point. In particular, there is no guarantee that the learned diffeomorphism is an identity at the goal, i.e., that $\psi(\text{Log}_{\mathfrak{g}}(\mathfrak{g})) = \psi(\mathbf{0}) = \psi^{-1}(\mathbf{0}) = \mathbf{0}$, which is sufficient to guarantee that base and diffeomorphic DSs have the same goal (Remark 4). This property is of importance in DS-based LfD, as we are generally interested in converging to a given target that is independent of the learning process. Moreover, since the base and diffeomorphic DSs have the same initial condition ($\mathfrak{a}_0 = \mathfrak{b}_0$), it is also beneficial that the learned diffeomorphism is an identity at the initial point, i.e., that $\psi(\text{Log}_{\mathfrak{a}_0}(\mathfrak{g})) = \mathbf{a}_0 = \psi^{-1}(\text{Log}_{\mathfrak{a}_0}(\mathfrak{b}_0)) = \mathbf{b}_0$, to prevent discontinuities in the initial velocity.

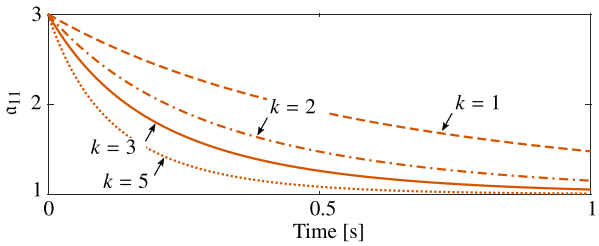


Fig. 4. The convergence rate of the base DS in (8) depends on the gain $k_{\mathfrak{a}} = \frac{k}{L\delta t}$. The figure shows the trajectory of the first entry $\mathfrak{a}_{11}$ of a 2×2 SPD matrix for $\mathfrak{a}_{11}(0) = 3$, $\mathfrak{g}_{11} = 1$, $L = 100$, $\delta t = 0.01$ s, and $k \in [1, 2, 3, 5]$.

In order to force the diffeomorphism to be an identity at the goal, we augment the learned GMM with a "small" component placed at $\text{Log}_{\mathfrak{g}}(\mathfrak{g}) = \mathbf{0}$. More in details, we augment the K learned components of the GMM (14) with π_{K+1} and $\mathcal{N}(\mathbf{a}, \mathbf{b}|\boldsymbol{\mu}_{K+1}, \boldsymbol{\Sigma}_{K+1})$ and set $\boldsymbol{\mu}_{K+1} = \mathbf{0}$ and $\boldsymbol{\Sigma}_{K+1} = k_{\mathcal{N}}\mathbf{I}$. We re-scale the priors from 1 to K as $\pi_k = \pi_k - \pi_{K+1}/K$ to ensure that the $K+1$ priors sum up to one. Conditioning with this new component makes points be mapped arbitrarily close to the goal. The distance to the goal depends on the gain $k_{\mathcal{N}}$. In this work, we set $k_{\mathcal{N}} = 1 \times 10^{-5}$ and $\pi_{K+1} = 0.01$. We use a similar strategy to enforce a smooth start. Given the initial point on the manifold $\mathfrak{a}_0 = \mathfrak{b}_0 = \bar{\mathfrak{a}}$, we project it into the TS and place a small Gaussian around it, i.e., $\pi_0 = 0.01$ and $\mathcal{N}(\mathbf{a}, \mathbf{b}|\boldsymbol{\mu}_0 = [\text{Log}_{\bar{\mathfrak{a}}}(\mathfrak{g})^\top, \mathbf{0}^\top]^\top, \boldsymbol{\Sigma}_0 = 1 \times 10^{-5}\mathbf{I})$. Conditioning with this new component ensures that $\psi(\text{Log}_{\bar{\mathfrak{a}}}(\mathfrak{g})) \approx \psi^{-1}(\text{Log}_{\bar{\mathfrak{a}}}(\mathfrak{b}_0)) \approx \text{Log}_{\bar{\mathfrak{a}}}(\mathfrak{g})$.

The possibility to change the goal, even on-the-fly (goal switching), is one of the appealing properties of DS-based skill representations. Changing the goal in our SDS-RM also is possible. However, as already known for other DS-based approaches [21], switching the goal may cause jumps in the generated trajectories and consequently high acceleration of the robot. In order to avoid this problem, we exploit a geodesic dynamics that smoothly varies the goal from $\mathfrak{g}$ to $\mathfrak{g}_{new}$. In formulas

$$\dot{\mathfrak{g}}_{new} = k_{\mathfrak{g}} \text{Log}_{\mathfrak{g}}(\mathfrak{g}_{new}), \tag{19}$$

where $k_{\mathfrak{g}} > 0$ ensures convergence to $\mathfrak{g}_{new}$ as stated in Theorem 1. An example of this procedure applied to UQ is shown in Section 7.1.

The tunable gain $k_{\mathfrak{b}}$ controls the convergence rate of the base DS in (8) (see Fig. 4). Given the demonstrations $\mathfrak{B} = \left\{ \mathfrak{b}_l^d \right\}_{l=1,d=1}^{L,D}$ and the sampling time δt, we want that $\mathfrak{a}_L^d$ – obtained using (13) – reaches $\mathfrak{b}_L^d$ within a certain time. As shown in Fig. 4, too small values of $k_{\mathfrak{b}}$ may fail to ensure that $\mathfrak{a}_L^d \approx \mathfrak{b}_L^d$. On the contrary, too large values of $k_{\mathfrak{b}}$ cause a large part of the trajectory to be close to $\mathfrak{b}_L^d$. This makes the learning problem harder as similar points need to be mapped into potentially different ones, i.e., the data distribution tends to be multi-modal. Inspired by results from linear systems analysis, we can rewrite the control gain as $k_{\mathfrak{a}} = \frac{k}{L\delta t}$. Dividing by $L\delta t$ makes the control gain independent from the number of samples and training time. Therefore, the newly introduced parameter k produces the same results at different temporal scales. Given the initial point on the manifold $\mathfrak{a}(0)$, one can choose how well the base trajectory has to reach the goal and define k accordingly.

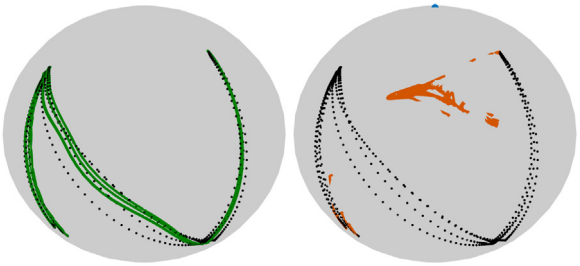
The proposed approach to learn stable DS on Riemannian manifolds is summarized in Algorithm 1.

Algorithm 1: SDS-RM

1: Pre-process data
   - Collect demonstrations $\mathfrak{B} = \left\{\mathfrak{b}_l^d\right\}_{l=1,d=1}^{L,D}$
   - Define the sampling time δt
   - Compute base trajectories $\mathfrak{A} = \left\{\mathfrak{a}_l^d\right\}_{l=1,d=1}^{L,D}$ using (13)
   - Project to TS using $\text{Log}_{\mathfrak{a}}(\mathfrak{g})$ ((2) or (4)) to obtain $\mathcal{A} = \left\{\mathbf{a}_l^d\right\}_{l=1,d=1}^{L,D}$, $\mathcal{B} = \left\{\mathbf{b}_l^d\right\}_{l=1,d=1}^{L,D}$. (For SPD profiles, vectorize data using Mandel's notation.)

2: Learn a diffeomorphism represented as a GMM
   - Place a small Gaussian component at the origin of the TS ($\pi_{K+1} = \overline{\pi}$, $\mathcal{N}(\mathbf{a}, \mathbf{b}|\mathbf{0}, k_{\mathcal{N}}\boldsymbol{I})$)
   - Place a small Gaussian component at the initial point ($\pi_0 = \overline{\pi}$, $\mathcal{N}(\mathbf{a}, \mathbf{b}|\overline{\mathbf{a}}, k_{\mathcal{N}}\boldsymbol{I})$)

3: Generate Riemannian motion via GMR
   - Compute ψ^{-1} from (17), $\boldsymbol{J}_{\psi}$ from (18), and the velocity from (11).
   - Project on the manifold using (13).



(a) Current TS (b) Lie algebra

Fig. 5. Results obtained with SDS-RM on the "N" shape.

5.4. Learning in current and fixed TS

In this example, we show the benefits of learning manifold motions in the tangent space placed at the current point, called the *current TS* in this work, in contrast to projecting the entire trajectory in a unique (fixed) TS. We use the "N" shape on $\mathcal{S}^2$ provided in [22] and shown in Fig. 5 (black dots). The trajectory is designed to span both the north and south hemispheres, where the Lie algebra is known to introduce a non-negligible approximation [22].

We follow the steps in Algorithm 1 using in one case the current TS and the Lie algebra (i.e., the TS at the north pole) in the other. Qualitative results in Fig. 5(a) confirm that using the current TS, SDS-RM can effectively encode the trajectory. The same result can be obtained using a TS at the goal and parallel transporting the data at each step (see the assumption in Theorem 1). However, this choice would increase the computational complexity due to the need for parallel transport. As expected, using the Lie algebra results in severe distortions (Fig. 5(b)).

6. Validation

In this section, we validate the proposed approach on a public benchmark – properly modified to represent trajectories evolving on UQ and SPD manifolds – and compare the results against FDM [9], R-GMM [22], and E-FLOW [23]. It is worth mentioning that FDM has not been designed to work on Riemannian manifolds. However, the procedure described in Section 4 allows exploiting different approaches to fit a diffeomorphism between TSs.

6.1. The Riemannian LASA dataset

In the LfD literature, there is no available dataset to test DS-based algorithm on Riemannian manifolds. Therefore, we have created a new one by modifying the popular benchmark – the LASA handwriting data-set [6] – to generate manifold (namely UQ and SPD) motions. The LASA handwriting contains 30 classes of 2D Euclidean motions starting from different initial points and converging to the same goal $[0, 0]^{\top}$. Each motion is demonstrated 7 times. A demonstration has exactly 1000 samples and includes position, velocity, and acceleration profiles.

The key idea to generate Riemannian data from Euclidean points is to consider each demonstration as an observation of a motion in the TS of a given Riemannian manifold. This allows us to use the exponential map to project the motion onto the manifold. As discussed in Section 3, the TS is computed wrt a point on the manifold. For converging motions, as the one generated by SDS-RM, the TS can be conveniently placed at the goal. We defined the goal as $\mathbf{q}_g = 1 + [0, 0, 0]^{\top}$ for UQs and as $\mathbf{G} = \text{diag}([100, 100])$ for SPD matrices, but other choices are possible. It is worth noticing that the described procedure is rather general and can be applied to Euclidean benchmarks different from the LASA dataset.

Data in the original LASA dataset are 2D (xy-plane), but the TS of UQs and 2×2 SPD matrices are[3] 3D. To add the third dimension, we consider the 7 demonstrations of each motion class as a matrix $\mathbf{C}_i$ for $i = 1, \ldots, 30$ with 14 rows and 1000 columns. Out of each $\mathbf{C}_i$, we extract the 4 matrices $\mathbf{C}_{1,i} = \mathbf{C}_i[0:2, :]$, $\mathbf{C}_{2,i} = \mathbf{C}_i[4:6, :]$, $\mathbf{C}_{3,i} = \mathbf{C}_i[8:10, :]$, and $\mathbf{C}_{4,i} = \mathbf{C}_i[[12, 13, 0], :]$. As a result, we obtain 4 demonstrations for each motion class, with the third component sampled for the demonstration along the x-axis. In this way, the third component is similar across different demonstrations of the same motion – as in a typical LfD setting – and contains sufficient complexity. Finally, the 3D trajectories contained in the matrices $\mathbf{C}_{1,i}$ to $\mathbf{C}_{4,i}$ are then projected to the corresponding manifold using the exponential map. For UQ, we scale the data between $[-1, 1]$ before projecting them on the unit sphere.

6.2. Evaluation procedure

We use the Riemannian LASA dataset described in the previous section to compare SDS-RM against two baselines and three state-of-the-art approaches. The baselines are built considering as base DS the Euclidean dynamics in (7) and a GMM-based diffeomorphism in Euclidean space. The baseline for UQs, named DS+Normalize, performs an extra normalization step to fulfill manifold constraints. The baseline for SPD matrices, named DS+Cholesky, exploits Cholesky decomposition and Mandel's notation to vectorize the matrix for training and re-build an SPD matrix from the generated vector. The other approaches included in the comparison are FDM, R-GMM, and E-FLOW. The Riemannian LASA dataset contains 30 classes. In this experiment, we consider a subset of 26 individual motions that neglects the 4 multi-modal motions. The multi-modal motions, obtained by combining 2 or 3 individual motions, contain different patterns

[3] The TS of a SPD matrix is a symmetric matrix which can be vectorized. For 2×2 SPD matrices the TS has 3 independent components.

Fig. 6. Qualitative results obtained on the Riemannian LASA dataset. Reproduced trajectories (brown solid lines) are obtained by applying the diffeomorphism learned with SDS-RM on the TS demonstrations (black dashed lines). It is worth noticing that TS data are in 3D, but we choose a view angle that makes the plot similar to the original 2D data. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

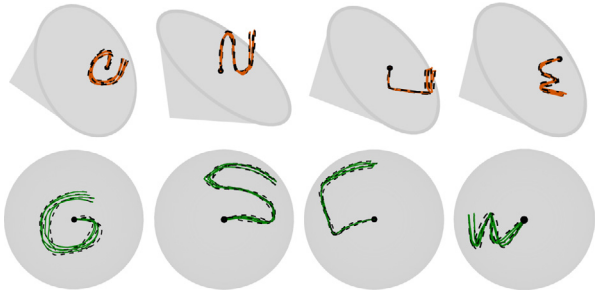


Fig. 7. Four motion classes of the Riemannian LASA dataset. (Top) Demonstrations (black dashed lines) and trajectories generated by SDS-RM (brown solid lines) evolving on the SPD cone. (Bottom) Demonstrations (black dashed lines) and trajectories generated by SDS-RM (green solid lines) evolving on the UQ sphere. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

in different areas of the state space. An approach that effectively learns from multiple demonstrations, like SDS-RM, R-GMM, and E-FLOW, can encode such a variability. This is qualitatively shown in the last four plots of Fig. 6. On the contrary, approaches that learn from a single demonstration, like FDM and DMP, are expected to perform poorly. In order to have a fair comparison with FDM, we neglect the 4 multi-modal motions in our comparison. For each of the 26 classes, we considered all the 4 available demonstrations in SDS-RM and only one (average) demonstration for FDM. We down-sampled the trajectories to contain exactly 100 points to significantly speed up the testing procedures and test the data efficiency of each approach.

The two baselines, as well as R-GMM and SDS-RM, have a single hyperparameter k that is the number of Gaussian components. For FDM, instead, the hyperparameter k is the number of kernel functions used to fit the diffeomorphism. On the contrary, E-FLOW has a few open hyperparameters including the network structure (number of layers, neurons per layer), the learning rate, and the number of epochs. Performing an exhaustive hyperparameters search requires a GPU cluster and it is beyond the scope of this work. Hence, we keep fixed the structure of the network and the learning rate (provided by the author's implementation) and vary the number of epochs k. Table 3 reports the value of k used in this experiment.

The performance of each approach is measured considering the accuracy in reproducing the demonstrations contained in the dataset and the Training Time (TT). The accuracy is measured as the Root Mean Square Error (RMSE) between each demonstration and the corresponding generated motion (first 100 steps), i.e., a trajectory generated starting from the same point on the manifold. Depending on the manifold (UQ or SPD), distances between points are computed considering the proper Riemannian distance (see Section 3). The TT is divided by the number of classes to represent the time needed to encode a single motion.

6.3. Results

The accuracy of SDS-RM in learning the motions in the Riemannian LASA dataset is qualitatively shown in Fig. 6. We show the demonstrated (black dashed lines) and reproduced (brown solid lines) motions in the TS. Recall that, in the Riemannian LASA dataset, UQ and SPD motions share the same TS up to a scaling factor. Therefore, we expect similar results in both manifolds. This is also shown in Fig. 7 where the learned trajectories for 4 motion classes in the dataset are projected on the SPD (top row) and UQ (bottom row) manifold. As expected, the generated motion on the manifold follows accurately the demonstration.

The quantitative results of this evaluation are shown in Table 3. For each approach, we use a metric to experimentally verify the violation of geometric constraints. For UQ, we compute the maximum deviation of the norm of each generated quaternion from 1 and found that the deviation is about 1×10^{-16}. For SPD matrices, we compute the minimum value of the eigenvalues of the generated matrices and found that it was always positive. As expected, all the considered approaches are able to fulfill the underlying geometric constraints and therefore to properly represent manifold data.

SDS-RM accurately represents manifold data and it outperforms the baselines as well as the state-of-the-art approaches FDM, R-GMM, and E-FLOW. The baselines are effective in fulfilling manifold constraints (the unit norm for UQ, symmetry and positive definiteness for SPD), but fail to accurately encode Riemannian data. SDS-RM is 45 % more accurate than FDM. This

Table 3
Results for different learning approaches applied to the Riemannian LASA dataset. We report mean and standard deviation for RMSE and TT.

$\mathcal{M}$	Approach	k	RMSE	TT [s]
$\mathcal{S}^3$	SDS-RM (ours)	10	0.029 ± 0.019	0.755 ± 0.241
	DS + Normalize	10	0.126 ± 0.113	1.621 ± 0.707
	FDM [9]	250	0.043 ± 0.030	0.201 ± 0.053
	R-GMM [22]	10	0.036 ± 0.016	0.387 ± 0.047
	E-FLOW [23]	1000	0.141 ± 0.128	112.25 ± 0.570
$\mathcal{S}^2_{++}$	SDS-RM (ours)	10	0.029 ± 0.019	1.514 ± 0.726
	DS + Cholesky	10	0.121 ± 0.043	1.899 ± 0.490
	FDM [9]	250	0.042 ± 0.029	0.221 ± 0.023
	R-GMM [22]	10	0.037 ± 0.017	14.514 ± 1.560
	E-FLOW [23]	1000	0.140 ± 0.126	127.55 ± 8.434

Authors would like to thank N. Perrin and P. Schlehuber-Caissier for providing the source code of the FDM approach in [9].

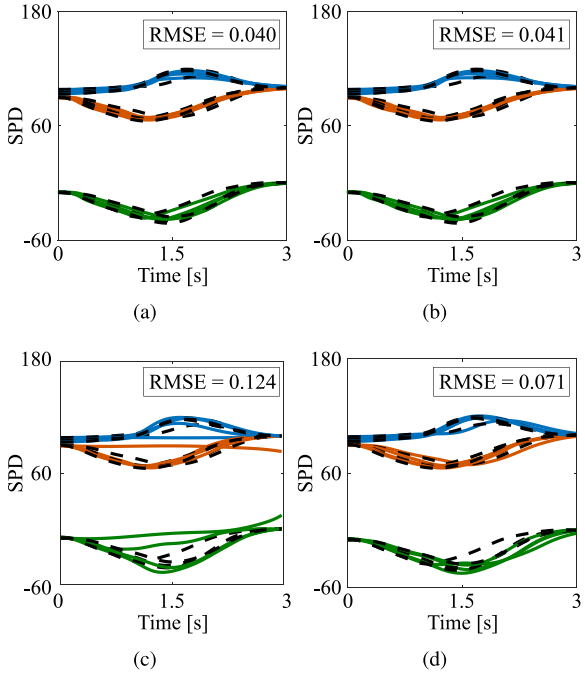


Fig. 8. Results for the data-efficiency test. SDS-RM is trained by sub-sampling each demonstration to 100 points. (a) SDS-RM reproduces accurately the sub-sampled demonstrations. (b) Adjusting the sampling time SDS-RM reproduces accurately the original demonstrations (1000 points) without re-training. (c) E-FLOW is inaccurate when learning from sub-sampled demonstrations. (d) E-FLOW achieves the SDS-RM accuracy when learning from the original demonstrations (1000 points).

is an expected result as FDM learns from a single demonstration obtained in this case by averaging the 4 demonstrations in each class. We expect a similar result by applying DMP-based approaches [20,21]. Regarding the training time, SDS-RM learns a UQ motion (4D) on average in 0.755 s, while FDM takes only 0.201 s. For SPD data (2×2 matrices), SDS-RM needs on average in 1.514 s the learn a motion, while FDM takes only 0.221 s. This is also expected as FDM uses only 1 demonstration for training, resulting in 4 times fewer data than SDS-RM. To summarize, FDM learns very quickly and it is recommended in applications where $<$ 0.5 s training time is needed. However, most applications do not have such a strict training time requirement but need accurate motions. In this case, SDS-RM offers high accuracy with a reasonable training time.

R-GMM learns from multiple demonstrations, but it outputs the same trajectory irrespective of the initial pose. This results in a loss of accuracy, with SDS-RM being about 25 % more accurate than R-GMM. Regarding the training time, R-GMM learns faster than SDS-RM a UQ motion (0.387 s on average), but it is almost 10 times slower on SPD. Overall, we conclude that SDS-RM offers a better compromise between accuracy and training time.

E-FLOW has the worst performance in terms of accuracy and training time. This has two main reasons. First, approaches based on invertible neural networks are sensitive to the hyperparameters choice, but performing an exhaustive hyperparameters search would have significantly increased the training time. Second, approaches based on invertible neural networks are data greedy. In other words, they require a relatively large amount of data to accurately learn. To demonstrate this, we performed a data-efficiency test by learning a motion in the Riemannian LASA dataset using 100 and 1000 points for each demonstration. Results comparing SDS-RM with E-FLOW are shown in Fig. 8. SDS-RM has the same accuracy (RMSE) in both cases, meaning data 100 points are already enough to accurately learn the motion. On the contrary, E-FLOW improves significantly using 1000 points (and the same hyperparameters setting). However, it needs about 900 s to fit a single motion with 1000 points. This may be acceptable in some applications, but it is a clear limitation in smart manufacturing where both precision and usability play a key role.

7. Robot experiments

This section presents experiments[4] with a 7 Degree of Freedom (DoF) manipulator (Franka Emika Panda). The robot's behavior is governed by the Cartesian impedance controller

$$\begin{aligned} \mathbf{F}_p &= \mathbf{K}_p \left(\mathbf{p}_d - \mathbf{p}\right) + \mathbf{D}_p \left(\dot{\mathbf{p}}_d - \dot{\mathbf{p}}\right), \\ \mathbf{F}_o &= \mathbf{K}_o \operatorname{Log}^q_{\mathbf{q}_d}(\mathbf{q}) + \mathbf{D}_o \left(\boldsymbol{\omega}_d - \boldsymbol{\omega}\right) \end{aligned} \tag{20}$$

where the subscript p stands for position, o for orientation, and d for desired. $\mathbf{p}_d$ and $\mathbf{q}_d$ are desired position and orientation (expressed as UQ) of the robot end-effector. $\mathbf{p}$ and $\mathbf{q}$ indicate the measured position and orientation of the end-effector. Desired and measured linear (angular) velocities are indicated as $\dot{\mathbf{p}}_d$ and $\dot{\mathbf{p}}$ ($\boldsymbol{\omega}_d$ and $\boldsymbol{\omega}$). $\mathbf{K}_{p/o}$ and $\mathbf{D}_{p/o}$ are the robot stiffness and damping matrices expressed in the robot base frame. Given the stiffness matrices $\mathbf{K}_{p/o}$ – computed as detailed later in this section – the damping matrices $\mathbf{D}_{p/o}$ are obtained by the double diagonalization design [61]. Cartesian forces defined in (20) are mapped into joint torques using the transpose of the manipulator Jacobian ($\mathbf{J}^\top$),

$$\boldsymbol{\tau}_d = \mathbf{J}^\top \begin{bmatrix} \mathbf{F}_p \\ \mathbf{F}_o \end{bmatrix}, \tag{21}$$

[4] A video of the experiments is available as supplementary material.

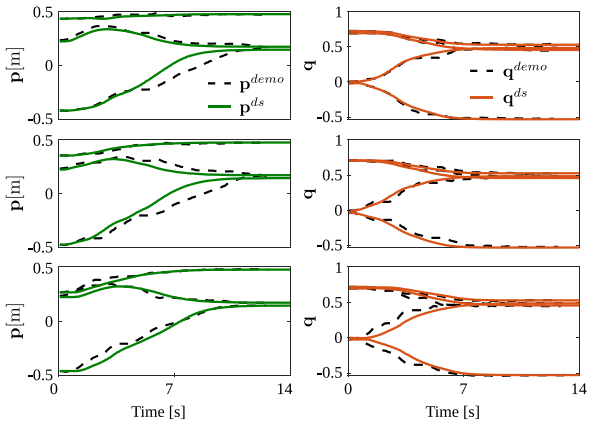


Fig. 9. Demonstrated (dashed lines) and reproduced (solid lines) pose trajectories for the bottle stacking experiment.

7.1. Bottle stacking

The task of stacking bottles in a rack requires continuous adjustment of position and orientation (see Fig. 10). Apart from reaching a desired (stacking) pose, the robot should follow accurately the demonstrated trajectory to prevent hitting the rack. We provide 3 kinesthetic demonstrations (dashed lines in Fig. 9) starting at different locations and converging to the stacking pose defined by $\mathbf{p}_g = [0.474, 0.185, 0.155]^\top$ m and $\mathbf{q}_g = -0.520 + [0.542, 0.442, 0.491]^\top$. The demonstrations are of the form $\{\{\mathbf{p}_{l,d}^{demo}, \mathbf{q}_{l,d}^{demo}\}_{l=1}^{L}\}_{d=1}^{D}$ where L is the total length of the demonstrations and $D = 3$ is the number of demonstrations. Demonstrated positions and orientations are encoded into two stable DSs using SDS-RM. We, empirically, use 10 Gaussian components for each system. It is worth mentioning that, in order to fit position trajectories, we, simply, replace logarithmic and exponential maps in Algorithm 1 with an identity map. Results of the learning process are shown in Fig. 9 (solid lines). The robot is controlled with the Cartesian impedance controller (20) where $\mathbf{p}_d$ and $\mathbf{q}_d$ are generated with SDS-RM. The stiffness matrices are kept to constant high values ($\mathbf{K}_p = \text{diag}([1000, 1000, 1000])$ N/m and $\mathbf{K}_o = \text{diag}([150, 150, 150])$ N m/rad) in this task. With the selected impedance gains, the robot is able to follow the desired pose trajectories, as shown in the top row of Fig. 10.

One of the interesting properties of DS-based trajectory generators is the possibility to converge to different goals. Changing the goal is possible also on Riemannian manifolds by following the approach we have presented in Section 5.3. To demonstrate the robustness of SDS-RM to goal switching we repeated the stacking task in different conditions. In each case, the shifted goal pose is obtained by manually placing the robot in a suitable release configuration (from which the bottle was released inside the rack) without correcting for small differences in the z-axis. Fig. 10 (middle) shows the robot successfully stacking the bottle at a different position ($\mathbf{p}_g = [0.385, 0.143, 0.172]^\top$ m). Fig. 10 (bottom) shows the robot successfully performing the stacking task with a rotated rack, which implies a significant change in the stacking orientation ($\mathbf{q}_g = -0.58 + [0.37, 0.63, 0.35]^\top$) and a less pronounced change in the goal position ($\mathbf{p}_g = [0.469, 0.200, 0.165]^\top$ m).

The results of this experiment show that SDS-RM accurately encodes full pose trajectories while ensuring convergence to a given target (even if not explicitly demonstrated) and fulfilling the underlying geometric constraints (the deviation of the norm from 1 is about 1×10^{-16}) in variable orientation data. For comparison, the baseline DS+Normalize fails to accurately reach the goal, and, as a result, the robot stacks the bottle in the wrong configuration (see the accompanying video). A similar lack of accuracy also affects E-FLOW, as experimentally shown in Section 6.3. R-GMM generates the same motion irrespective of the initial/goal point, which means it would fail in stacking the bottle in a different location.

7.2. Cooperative drilling

In this task, the robot picks a wooden plate from a container and moves it to a demonstrated pose where a human operator will drill it (see Fig. 1). Therefore, the robot has to be stiff at the end of the motion to keep the desired pose during the interaction (drilling). During the motion, low impedance gains can be used to make the robot compliant. More importantly, the robot needs to be compliant at the beginning of the motion in order to gently pick the wooden plate from the container. Otherwise, small inaccuracies in the generated motion trajectory may cause abrupt movements of the container. We provide 3 kinesthetic demonstrations (see Fig. 11(a)) from different starting poses and converging to the same goal. The demonstrations are of the form $\{\{\mathbf{p}_{l,d}^{demo}, \mathbf{q}_{l,d}^{demo}\}_{l=1}^{L}\}_{d=1}^{D}$ where L is the total length of the demonstrations and $D = 3$ is the number of demonstrations.

As in the previous experiment, demonstrated positions and orientations are encoded into two stable DSs using SDS-RM. We use 10 Gaussian components for each system. The desired variable stiffness profiles are generated using the variability in the demonstrations as suggested in several previous works [62–64]. More in detail, we first notice that the Cartesian impedance controller (20) assumes that position and orientation are decoupled. In other words, it assumes that positional errors only affect the force, while rotational errors only affect the torque. This allows us to learn independent linear and angular stiffness profiles. The idea is to compute the desired stiffness profile from the inverse of the covariance matrix.

For the linear stiffness matrix $\mathbf{K}_p$, we first compute the covariance matrix for each of the D demonstrated positions and for each time step $l = 1, \dots, L$ as

$$\boldsymbol{\Sigma}_{p,l} = \frac{1}{D}\sum_{i=1}^{D}\left(\mathbf{p}_{l,i}^{demo} - \boldsymbol{\mu}_{p,l}\right)\left(\mathbf{p}_{l,i}^{demo} - \boldsymbol{\mu}_{p,l}\right)^\top, \tag{22}$$

where the mean $\boldsymbol{\mu}_{p,l}$ is computed as

$$\boldsymbol{\mu}_{p,l} = \frac{1}{D}\sum_{i=1}^{D}\mathbf{p}_{l,i}^{demo}. \tag{23}$$

Then, we compute the eigenvalue decomposition of each $\boldsymbol{\Sigma}_{p,l} = \mathbf{V}_l\boldsymbol{\Lambda}_l\mathbf{V}_l^\top$, where $\mathbf{V}_l$ is an orthogonal matrix and $\boldsymbol{\Lambda}_l = \text{diag}([\lambda_{1,l}, \lambda_{2,l}, \lambda_{3,l}])$. Since all the demonstrations end up in the same position, we know that the eigenvalues of $\boldsymbol{\Sigma}_{p,L}$ vanishes, i.e., $\boldsymbol{\Lambda}_L = \mathbf{0}$ and $\mathbf{V} = \mathbf{I}$. Moreover, we want the stiffness to be maximum at L. Therefore, we compute the desired stiffness profile as

$$\mathbf{K}_{p,l}^{demo} = \mathbf{V}_l\begin{bmatrix}\frac{1}{\lambda_{1,l}+\bar{k}_p} & 0 & 0\\ 0 & \frac{1}{\lambda_{2,l}+\bar{k}_p} & 0\\ 0 & 0 & \frac{1}{\lambda_{3,l}+\bar{k}_p}\end{bmatrix}\mathbf{V}_l^\top, \tag{24}$$

where the maximum linear stiffness gain is set to $\bar{k}_p = 1000$ N/m. As shown in Fig. 11(b), the stiffness profile in (24) converges to $\mathbf{K}_{p,L} = \text{diag}([\bar{k}_p, \bar{k}_p, \bar{k}_p])$ N/m and varies according to the variability in the demonstrations. Note that existing approaches also impose a minimum value for the stiffness. This is straightforward to implement but it was not needed in the performed experiment as the minimum value of the stiffness computed by means of (24) was already enough to track the desired trajectory.

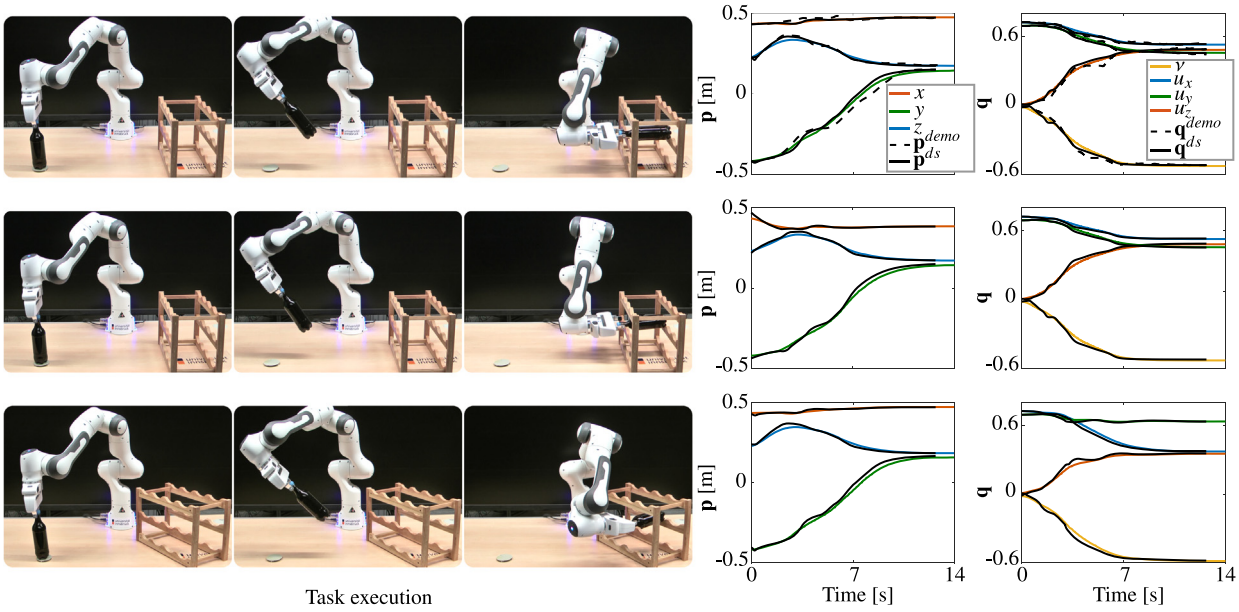
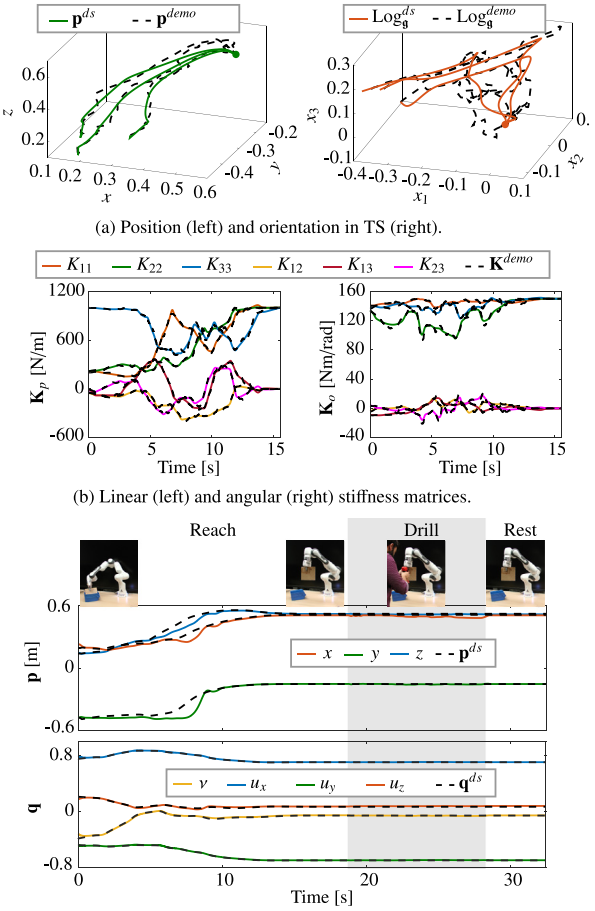


Fig. 10. Results of the bottle stacking experiment collected from three runs; each starts from the same starting pose towards different goals. Left panels show the snapshots of the robot execution; right panels show desired and executed motion trajectories. (Top) The robot reproduces the trajectory generated by SDS-RM from known (demonstrated) starting and goal poses. (Middle) The robot stacks the bottle at a different position ($\mathbf{p}_g = [0.385, 0.143, 0.172]^\top$ m). (Bottom) The rack is rotated and the robot stacks the bottle at a different pose ($\mathbf{p}_g = [0.469, 0.200, 0.165]^\top$ m and $\mathbf{q}_g = -0.58 + [0.37, 0.63, 0.35]^\top$).



(a) Position (left) and orientation in TS (right).

(b) Linear (left) and angular (right) stiffness matrices.

(c) Task execution.

Fig. 11. Results for the cooperative drilling experiment.

The angular stiffness matrix $\mathbf{K}_o$ is typically kept constant [62, 63] or diagonal [64] in related work. We propose instead to exploit the variance of the demonstrations in the tangent space of the UQ to derive a full stiffness profile. This is possible as the tangent space is locally Euclidean. The first step is to project the demonstrated orientations in the tangent space at the goal quaternion $\mathbf{q}_g$, obtaining $\{\{\mathbf{q}_{l,d}^{demo} = \mathrm{Log}_{\mathbf{q}_g}(\mathbf{q}_{l,d}^{demo})\}_{l=1}^{L}\}_{d=1}^{D}$. We compute the covariance matrix of the tangent space demonstrations $\mathbf{q}_{l,d}^{demo}$ as

$$\boldsymbol{\Sigma}_{q,l} = \frac{1}{D}\sum_{i=1}^{D}\left(\mathbf{q}_{l,i}^{demo} - \boldsymbol{\mu}_{q,l}\right)\left(\mathbf{q}_{l,i}^{demo} - \boldsymbol{\mu}_{q,l}\right)^\top, \tag{25}$$

where the mean $\boldsymbol{\mu}_{q,l} = \frac{1}{D}\sum_{i=1}^{D}\mathbf{q}_{l,i}^{demo}$. As for the linear stiffness, we compute the eigenvalue decomposition of $\boldsymbol{\Sigma}_{o,l} = \mathbf{U}_l\boldsymbol{\Gamma}_l\mathbf{U}_l$, where $\mathbf{U}_l$ is an orthogonal matrix and $\boldsymbol{\Gamma}_l = \mathrm{diag}([\gamma_{1,l}, \gamma_{2,l}, \gamma_{3,l}])$. Since all the tangent space data end up to zero – as the tangent space is placed at the goal – we know that the eigenvalues of $\boldsymbol{\Sigma}_{o,L}$ vanishes, i.e., $\boldsymbol{\Gamma}_L = \mathbf{0}$ and $\mathbf{U} = \mathbf{I}$. Moreover, we want the stiffness to be maximum at L. Therefore, we compute the desired stiffness profile as

$$\mathbf{K}_{o,l}^{demo} = \mathbf{U}_l\begin{bmatrix}\frac{1}{\gamma_{1,l}+\bar{k}_o} & 0 & 0\\ 0 & \frac{1}{\gamma_{2,l}+\bar{k}_o} & 0\\ 0 & 0 & \frac{1}{\gamma_{3,l}+\bar{k}_o}\end{bmatrix}\mathbf{U}_l^\top, \tag{26}$$

where the maximum angular stiffness gain is set to $\bar{k}_0 = 150$ N m/rad. As shown in Fig. 11(b), the stiffness profile computed in (24) converges to $\mathbf{K}_{o,L} = \mathrm{diag}([\bar{k}_o, \bar{k}_o, \bar{k}_o])$ N m/rad and varies according to the variability in the demonstrations.

The generated linear and angular stiffness profiles are encoded into two stable DSs using SDS-RM. We, empirically, use 15 Gaussian components for each system. The results of the learning procedure, shown in Fig. 11(b), confirm that SDS-RM accurately reproduces complex SPD profiles while ensuring convergence to a given goal.

After the learning, the pose trajectory and stiffness profiles are used to control the robot (see Fig. 11(c)). The robot picks

the wooden plate from a (blue) container and reaches the drill pose. During the motion, the robot is complaint which allows a safer response to possible external perturbations. For instance, a high initial stiffness combined with (small) inaccuracies in the generated trajectory could have generated a jerky motion of the container while the robot lifts the wooden plate. The goal pose, instead, is reached with maximum stiffness. As shown in Fig. 11(c), during the drilling task the maximum position deviation along the drilling direction (x-axis) is 3.1 cm (generating a reaction force of 31 N), while the maximum orientation deviation about the z-axis (where the robot perceives the highest momentum) is 1.8 deg (generating a reaction torque of 4.7 N/m). This shows that the robot is capable to keep the goal pose, letting the human co-worker drill the wooden plate.

8. Conclusions

In this paper, we presented Stable Dynamical System on Riemannian Manifolds (SDS-RM), an approach to learn stable DSs evolving on Riemannian manifolds. SDS-RM builds on theoretical stability results, derived for dynamics evolving on Riemannian manifolds, to learn stable and accurate DS representations of Riemannian data. Similar to its Euclidean counterparts, SDS-RM learns a diffeomorphic transformation between a simple stable system and a set of complex demonstrations. The key difference wrt Euclidean approaches is that SDS-RM uses tools from differential geometry to correctly represent complex manifold data, such as orientations and stiffness matrices, with their underlying geometric constraints, e.g., unit norm for unit quaternion orientation and symmetry and positive definiteness for stiffness matrices. The proposed approach is first evaluated in simulation and compared with an existing approach, modified to deal with Riemannian data. Due to the lack of publicly available Riemannian datasets, we developed a procedure to augment a popular – and potentially any other – Euclidean benchmark with UQ and SPD profiles. Finally, in order to perform a thorough evaluation, we also conducted a set of experiments with a real robot performing bottle stacking and cooperative (with a human operator) drilling. Overall, the conducted evaluation shows that SDS-RM represents a good compromise between accuracy and training time and that it can be effectively adopted to generate complex robotic skills on manifolds.

In our evaluation, we show the adaptation capabilities of SDS-RM by changing initial and/or goal points. However, in more general settings, it is needed to incorporate task-dependant parameters to adapt the execution to different domains. Augmenting SDS-RM with task-dependant parameters while maintaining its stability properties is an interesting research direction that we intend to explore in future work.

Moreover, SDS-RM has been evaluated on orientation (UQ) and stiffness (SPD) profiles, but it may be extended to other Riemannian manifolds. Therefore, our future research will also focus on investigating the possibility to learn stable DS on diverse manifolds like Grassmann or hyperbolic. Grassmann manifolds elegantly encode orthogonal projections, while hyperbolic manifolds represent a continuous embedding of discrete structures with possible application to task and motion planning. These manifolds are widely unexploited in robotics and can potentially unlock new applications.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The link to the data is shared in the manuscript. Code will be shared on request.

Acknowledgments

Part of the research presented in this work has been conducted when: M. Saveriano was at the Department of Computer Science, University of Innsbruck, Innsbruck, Austria, and F. Abu-Dakka was at the Intelligent Robotics Group, Department of Electrical Engineering and Automation, Aalto University, Finland.

This work has been partially supported by the Austrian Research Foundation (Euregio IPN 86-N30, OLIVER), by CHIST-ERA project IPALM (Academy of Finland decision 326304), by the European Union under NextGenerationEU project iNest (ECS 00000043), and by euROBIN project under grant agreement No. 101070596.

Appendix A. Jacobian of the mean of a GMR

Recall that

$$\begin{aligned}\psi(\mathbf{a}) &= \sum_{k=1}^{K} h_k(\mathbf{a})\hat{\psi}(\mathbf{a}),\\ \hat{\psi}(\mathbf{a}) &= \boldsymbol{\mu}_k^b + \boldsymbol{\Sigma}_k^{ba}(\boldsymbol{\Sigma}_k^{aa})^{-1}(\mathbf{a}-\boldsymbol{\mu}_k^a).\end{aligned} \tag{27}$$

Using the chain rule and (27), $J_\psi(\mathbf{a})$ writes as:

$$\begin{aligned}\mathbf{J}_\psi(\mathbf{a}) = \frac{\partial \psi(\mathbf{a})}{\partial \mathbf{a}} = \sum_{k=1}^{K} \frac{\partial h_k(\mathbf{a})}{\partial \mathbf{a}}\hat{\psi}(\mathbf{a})^\top \\ + h_k(\mathbf{a})\frac{\partial \hat{\psi}(\mathbf{a})}{\partial \mathbf{a}}.\end{aligned} \tag{28}$$

Let us compute the two partial derivatives at the right side of (28) separately. Considering the expression of $\hat{\psi}(\mathbf{a})$ in (27), and applying the chain rule, it is easy to verify that

$$\frac{\partial \hat{\psi}(\mathbf{a})}{\partial \mathbf{a}} = \boldsymbol{\Sigma}_k^{ab}(\boldsymbol{\Sigma}_k^{aa})^{-1}. \tag{29}$$

Using the quotient rule, and setting $\hat{\mathcal{N}}_k = \mathcal{N}(\mathbf{a}|\boldsymbol{\mu}_k^a, \boldsymbol{\Sigma}_k^{aa})$, the expression of $\frac{\partial h_k(\mathbf{a})}{\partial \mathbf{a}}$ writes as

$$\begin{aligned}\frac{\partial h_k(\mathbf{a})}{\partial \mathbf{a}} =& \frac{\pi_k \frac{\partial \hat{\mathcal{N}}_k}{\partial \mathbf{a}} \sum_{i=1}^{K} \pi_i \hat{\mathcal{N}}_i}{\left(\sum_{i=1}^{K} \pi_i \hat{\mathcal{N}}_i\right)^2} \\ &- \frac{\pi_k \hat{\mathcal{N}}_k \sum_{i=1}^{K} \pi_i \frac{\partial \hat{\mathcal{N}}_i}{\partial \mathbf{a}}}{\left(\sum_{i=1}^{K} \pi_i \hat{\mathcal{N}}_i\right)^2}.\end{aligned} \tag{30}$$

Recall that the derivative of a multivariate Gaussian distribution $\hat{\mathcal{N}}$ wrt the input is given by [65]

$$\frac{\partial \hat{\mathcal{N}}}{\partial \mathbf{a}} = -\hat{\mathcal{N}}\boldsymbol{\Sigma}^{-1}(\mathbf{a}-\boldsymbol{\mu}). \tag{31}$$

Using (31) to compute the derivatives in (30) we obtain:

$$\begin{aligned}\frac{\partial h_k(\mathbf{a})}{\partial \mathbf{a}} =& \frac{-\pi_k \hat{\mathcal{N}}_k (\boldsymbol{\Sigma}_k^{aa})^{-1}(\mathbf{a}-\boldsymbol{\mu}_k^a) \sum_{i=1}^{K} \pi_i \hat{\mathcal{N}}_i}{\left(\sum_{i=1}^{K} \pi_i \hat{\mathcal{N}}_i\right)^2} \\ &+ \frac{\pi_k \hat{\mathcal{N}}_k \sum_{i=1}^{K} \pi_i \hat{\mathcal{N}}_i (\boldsymbol{\Sigma}_i^{aa})^{-1}(\mathbf{a}-\boldsymbol{\mu}_i^a)}{\left(\sum_{i=1}^{K} \pi_i \hat{\mathcal{N}}_i\right)^2} \\ =& \frac{\pi_k \hat{\mathcal{N}}_k}{\sum_{i=1}^{K} \pi_i \hat{\mathcal{N}}_i} \left(-(\boldsymbol{\Sigma}_k^{aa})^{-1}(\mathbf{a}-\boldsymbol{\mu}_k^a) \frac{\sum_{i=1}^{K} \pi_i \hat{\mathcal{N}}_i}{\sum_{i=1}^{K} \pi_i \hat{\mathcal{N}}_i} \right.\end{aligned}$$

$$+ \frac{\sum_{i=1}^{K} \pi_i \hat{\mathcal{N}}_i (\boldsymbol{\Sigma}_i^{aa})^{-1} (\mathbf{a} - \boldsymbol{\mu}_i^a)}{\sum_{i=1}^{K} \pi_i \hat{\mathcal{N}}_i} \Bigg)$$
$$= h_k(\mathbf{a}) \left(-(\boldsymbol{\Sigma}_k^{aa})^{-1} (\mathbf{a} - \boldsymbol{\mu}_k^p) + \sum_{i=1}^{K} h_i(\mathbf{a}) (\boldsymbol{\Sigma}_i^{aa})^{-1} (\mathbf{a} - \boldsymbol{\mu}_i^a) \right). \quad (32)$$

By substituting (29) and (32) into (28), we obtain the sought expression of the Jacobian in (18).

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2023.104510.

References

[1] J. Kuffner, J. Xiao, Motion for manipulation tasks, in: Springer Handbook of Robotics, Springer, 2016, pp. 897–930.
[2] S. Schaal, Is imitation learning the route to humanoid robots? Trends Cogn. Sci. 3 (6) (1999) 233–242.
[3] A. Billard, S. Calinon, R. Dillmann, Learning from humans, in: Springer Handbook of Robotics, Second Ed., 2016.
[4] H. Ravichandar, A.S. Polydoros, S. Chernova, A. Billard, Recent advances in robot learning from demonstration, Ann. Rev. Control Robot Autonomous Syst. 3 (1) (2020) 297–330.
[5] A.J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, Dynamical movement primitives: Learning attractor models for motor behaviors, Neural Comput. 25 (2) (2013) 328–373.
[6] S.M. Khansari-Zadeh, A. Billard, Learning stable non-linear dynamical systems with Gaussian mixture models, IEEE Trans. Robot. 27 (5) (2011) 943–957.
[7] S.M. Khansari-Zadeh, A. Billard, Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions, Robot. Auton. Syst. 62 (6) (2014) 752–765.
[8] K. Neumann, J.J. Steil, Learning robot motions with stable dynamical systems under diffeomorphic transformations, Robot. Auton. Syst. 70 (2015) 1–15.
[9] N. Perrin, P. Schlehuber-Caissier, Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems, Systems Control Lett. 96 (2016) 51–59.
[10] C. Blocher, M. Saveriano, D. Lee, Learning stable dynamical systems using contraction theory, in: Nternational Conference on Ubiquitous Robots and Ambient Intelligence, 2017, pp. 124–129.
[11] J. Duan, Y. Ou, J. Hu, Z. Wang, S. Jin, C. Xu, Fast and stable learning of dynamical systems based on extreme learning machine, Trans. Syst. Man Cybern.: Syst. 49 (6) (2017) 1175–1185.
[12] A. Lemme, F. Reinhart, K. Neumann, J.J. Steil, Neural learning of vector fields for encoding stable dynamical systems, Neurocomputing 141 (2014) 3–14.
[13] F. Khadivar, I. Lauzana, A. Billard, Learning dynamical systems with bifurcations, Robot. Auton. Syst. 136 (2021) 103700.
[14] M. Ginesi, D. Meli, A. Roberti, N. Sansonetto, P. Fiorini, Dynamic movement primitives: Volumetric obstacle avoidance using dynamic potential functions, J. Intell. Robot. Syst. 101 (4) (2021) 1–20.
[15] M. Saveriano, F. Hirt, D. Lee, Human-aware motion reshaping using dynamical systems, Pattern Recognit. Lett. 99 (2017) 96–104.
[16] M. Saveriano, D. Lee, Learning barrier functions for constrained motion planning with dynamical systems, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2019, pp. 112–119.
[17] K. Kronander, S.M. Khansari-Zadeh, A. Billard, Incremental motion learning with locally modulated dynamical systems, Robot. Auton. Syst. 70 (2015) 52–62.
[18] M. Saveriano, D. Lee, Incremental skill learning of stable dynamical systems, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2018, pp. 6574–6581.
[19] M.P. Do Carmo, J. Flaherty Francis, Riemannian Geometry, Vol. 6, Springer, 1992.
[20] A. Ude, B. Nemec, T. Petric, J. Morimoto, Orientation in Cartesian space dynamic movement primitives, in: ICRA, 2014, pp. 2997–3004.
[21] F.J. Abu-Dakka, V. Kyrki, Geometry-aware dynamic movement primitives, in: ICRA, 2020, pp. 4421–4426.
[22] S. Calinon, Gaussians on Riemannian manifolds: Applications for robot learning and adaptive control, IEEE Robot. Autom. Mag. (RAM) 27 (2) (2020) 33–45.
[23] M.A. Rana, A. Li, D. Fox, B. Boots, F. Ramos, N. Ratliff, Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems, in: A.M. Bayen, A. Jadbabaie, G. Pappas, P.A. Parrilo, B. Recht, C. Tomlin, M. Zeilinger (Eds.), Conference on Learning for Dynamics and Control, in: Proceedings of Machine Learning Research, vol.120, 2020, pp. 630–639.
[24] J. Urain, M. Ginesi, D. Tateo, J. Peters, Imitationflow: Learning deep stable stochastic dynamic systems by normalizing flows, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2020, pp. 5231–5237.
[25] J. Urain, D. Tateo, J. Peters, Learning stable vector fields on Lie groups, 2021, arXiv preprint arXiv:2110.11774.
[26] W. Wang, M. Saveriano, F.J. Abu-Dakka, Learning deep robotic skills on Riemannian manifolds, IEEE Access 10 (2022) 114143–114152.
[27] D.A. Cohn, Z. Ghahramani, M.I. Jordan, Active learning with statistical models, J. Artif. Intell. Res. 4 (1996) 129–145.
[28] Z. Gao, Y. Wu, M. Harandi, Y. Jia, A robust distance measure for similarity-based classification on the SPD manifold, IEEE Trans. Neural Netw. Learn. Syst. 31 (9) (2020) 3230–3244.
[29] Z. Huang, R. Wang, S. Shan, L. Van Gool, X. Chen, Cross Euclidean-to-Riemannian metric learning with application to face recognition from video, IEEE Trans. Pattern Anal. Mach. Intell. 40 (12) (2017) 2827–2840.
[30] R. Chakraborty, J. Bouza, J.H. Manton, B.C. Vemuri, Manifoldnet: A deep neural network for manifold-valued data with applications, IEEE Trans. Pattern Anal. Mach. Intell. 44 (2) (2022) 799–810.
[31] M.M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, Geometric deep learning: Going beyond Euclidean data, IEEE Signal Process. Mag. 34 (4) (2017) 18–42.
[32] M.Y. Seker, M. Imre, J.H. Piater, E. Ugur, Conditional neural movement primitives, in: Robotics: Science and Systems, Vol. 10, 2019.
[33] S. Bahl, M. Mukadam, A. Gupta, D. Pathak, Neural dynamic policies for end-to-end sensorimotor learning, in: NeurIPS, 2020.
[34] D.A. Reynolds, Gaussian mixture models, Encycl. Biom. 741 (2009) 659–663.
[35] S. Calinon, D. Bruno, D.G. Caldwell, A task-parameterized probabilistic model with minimal intervention control, in: ICRA, 2014, pp. 3339–3344.
[36] A. Paraschos, C. Daniel, J. Peters, G. Neumann, Probabilistic movement primitives, in: NeurIPS, 2013, pp. 2616–2624.
[37] Y. Huang, F.J. Abu-Dakka, J. Silvério, D.G. Caldwell, Toward orientation learning and adaptation in Cartesian space, IEEE Trans. Robot. 37 (1) (2021) 82–98.
[38] M.J. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, D.G. Caldwell, An approach for imitation learning on Riemannian manifolds, IEEE Robot. Autom. Lett. 2 (3) (2017) 1240–1247.
[39] F.J. Abu-Dakka, L. Rozo, D.G. Caldwell, Force-based variable impedance learning for robotic manipulation, Robot. Auton. Syst. 109 (2018) 156–167.
[40] F. Abu-Dakka, Y. Huang, J. Silvério, V. Kyrki, A probabilistic framework for learning geometry-based robot manipulation skills, Robot. Auton. Syst. 141 (2021) 103761.
[41] N. Jaquier, S. Calinon, Gaussian mixture regression on symmetric positive definite matrices manifolds: Application to wrist motion estimation with sEMG, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2017, pp. 59–64.
[42] N. Jaquier, L. Rozo, D.G. Caldwell, S. Calinon, Geometry-aware manipulability learning, tracking and transfer, Int. J. Robot. Res. (IJRR) (2020).
[43] M. Saveriano, F.J. Abu-Dakka, A. Kramberger, L. Peternel, Dynamic movement primitives in robotics: A tutorial survey, 2021, arXiv preprint arXiv:2102.03861.
[44] F.J. Abu-Dakka, B. Nemec, J.A. Jørgensen, T.R. Savarimuthu, N. Krüger, A. Ude, Adaptation of manipulation skills in physical contact with the environment to reference force profiles, Auton. Robots 39 (2) (2015) 199–217.
[45] F.J. Abu-Dakka, M. Saveriano, L. Peternel, Periodic DMP formulation for quaternion trajectories, in: IEEE International Conference of Advanced Robotics, 2021, pp. 658–663.
[46] M. Saveriano, F. Franzel, D. Lee, Merging position and orientation motion primitives, in: ICRA, 2019, pp. 7041–7047.
[47] M. Saveriano, An energy-based approach to ensure the stability of learned dynamical systems, in: IEEE International Conference on Robotics and Automation, 2020, pp. 4407–4413.
[48] W. Lohmiller, J. Slotine, On contraction analysis for nonlinear systems, Automatica 34 (6) (1998) 683–696.
[49] H. Ravichandar, A. Dani, Learning position and orientation dynamics from demonstrations via contraction analysis, Auton. Robots 43 (4) (2019) 897–912.
[50] N.D. Ratliff, J. Issac, D. Kappler, S. Birchfield, D. Fox, Riemannian motion policies, 2018, arXiv preprint arXiv:1801.02854.

[51] M. Mukadam, C.-A. Cheng, D. Fox, B. Boots, N. Ratliff, Riemannian motion policy fusion through learnable Lyapunov function reshaping, in: Conference on Robot Learning, PMLR, 2020, pp. 204–219.
[52] J. Sola, J. Deray, D. Atchuthan, A micro Lie theory for state estimation in robotics, 2018, arXiv preprint arXiv:1812.01537.
[53] X. Pennec, P. Fillard, N. Ayache, A Riemannian framework for tensor computing, Int. J. Comput. Vis. 66 (1) (2006) 41–66.
[54] S. Sra, R. Hosseini, Conic geometric optimization on the manifold of positive definite matrices, SIAM J. Optim. 25 (1) (2015) 713–739.
[55] J. Slotine, W. Li, Applied Nonlinear Control, Prentice-Hall Englewood Cliffs, 1991.
[56] F. Pait, D. Colón, Some properties of the Riemannian distance function and the position vector X, with applications to the construction of Lyapunov functions, in: IEEE Conference on Decision and Control, 2010, pp. 6277–6280.
[57] S. Fiori, Manifold calculus in system theory and control–fundamentals and first-order systems, Symmetry 13 (11) (2021).
[58] X. Pennec, Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements, J. Math. Imaging Vision 25 (1) (2006) 127–154.
[59] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics: Modelling, Planning and Control, Springer, 2009.
[60] S. Calinon, Robot Programming by Demonstration: A Probabilistic Approach, EPFL/CRC Press, 2009.
[61] A. Albu-Schaffer, C. Ott, U. Frese, G. Hirzinger, Cartesian impedance control of redundant robots: Recent results with the DLR-light-weight-arms, in: 2003 IEEE International Conference on Robotics and Automation, 2003, pp. 3704–3709.
[62] S. Calinon, I. Sardellitti, D.G. Caldwell, Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 249–254.
[63] J. Silvério, Y. Huang, F.J. Abu-Dakka, L. Rozo, D.G. Caldwell, Uncertainty-aware imitation learning using kernelized movement primitives, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2019, pp. 90–97.
[64] K. Kronander, A. Billard, Learning compliant manipulation through kinesthetic and tactile human-robot interaction, IEEE Trans. Haptics 7 (3) (2013) 367–380.
[65] K.B. Petersen, M.S. Pedersen, The Matrix Cookbook, Technical University of Denmark, 2012.

Matteo Saveriano received his B.Sc. and M.Sc. degree in automatic control engineering from University of Naples, Italy, in 2008 and 2011, respectively. He received is Ph.D. from the Technical University of Munich in 2017. Currently, he is an assistant professor at the Department of Industrial Engineering (DII), University of Trento, Italy. Previously, he was an assistant professor at the University of Innsbruck and a post-doctoral researcher at the German Aerospace Center (DLR). He is an Associate Editor for RA-L. His research activities include robot learning, human–robot interaction, understanding and interpreting human activities.

Fares J. Abu-Dakka received his B.Sc. degree in Mechanical Engineering from Birzeit University, Palestine in 2003 and his DEA and Ph.D. degrees in robotics motion planning from the Polytechnic University of Valencia, Spain in 2006 and 2011, respectively. Between 2013 and 2016, he held a visiting professor position at ISA of the Carlos III University of Madrid, Spain. In the period between 2016 and 2019, he was a Postdoc at Istituto Italiano di Tecnologia. During 2019–2022, he was a Research Fellow at Aalto University. Currently, since 2022, he is a Senior Scientist and leading the Robot Learning Group at MIRMI, Technical University of Munich, Germany. His research lies in the intersection of control theory, differential geometry, and machine learning in order to enhance robot manipulation performance and safety. He is an Associate Editor for IEEE-ICRA, IEEE-IROS, and IEEE-RA-L.

Ville Kyrki is Associate Professor in automaton technology at the Department of Electrical Engineering and Automaton, Aalto University, Finland. During 2009–2012 he was a professor in computer science with specialization in intelligent robotics at the Lappeenranta University of Technology, Finland. He received the M.Sc. and Ph.D. degrees in computer science from Lappeenranta University of Technology in 1999 and 2002, respectively. He conducts research mainly in intelligent robotic systems, with emphasis on methods and systems that cope with imperfect knowledge and uncertain senses.