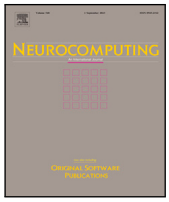
Neurocomputing 598 (2024) 128056

Contents lists available at ScienceDirect

Neurocomputing

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



A unified formulation of geometry-aware discrete dynamic movement primitives

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

[a] Electronic and Computer Science Department, Faculty of Engineering, Mondragon Unibertsitatea, 20500 Arrasate, Spain
[b] Automatic Control Lab, Department of Industrial Engineering, University of Trento, Trento, Italy
[c] Intelligent Robotics Group, Department of Electrical Engineering and Automation at Aalto University, Aalto, Finland

ARTICLE INFO

Communicated by Z. Wang

Keywords:
Motor control of artificial systems
Movement primitives theory
Dynamic movement primitives
Learning from demonstration
Riemannian manifolds

ABSTRACT

Learning from demonstration (LfD) is considered as an efficient way to transfer skills from humans to robots. Traditionally, LfD has been used to transfer Cartesian and joint positions and forces from human demonstrations. The traditional approach works well for some robotic tasks, but for many tasks of interest, it is necessary to learn skills such as orientation, impedance, and/or manipulability that have specific geometric characteristics. An effective encoding of such skills can be only achieved if the underlying geometric structure of the skill manifold is considered and the constrains arising from this structure are fulfilled during both learning and execution. However, typical learned skill models such as dynamic movement primitives (DMPs) are limited to Euclidean data and fail in correctly embedding quantities with geometric constraints. In this paper, we propose a novel and mathematically principled framework that uses concepts from Riemannian geometry to allow DMPs to properly embed geometric constrains. The resulting DMP formulation can deal with data sampled from any Riemannian manifold including, but not limited to, unit quaternions and symmetric and positive definite matrices. The proposed approach has been extensively evaluated both on simulated data and real robot experiments. The performed evaluation demonstrates that beneficial properties of DMPs, such as convergence to a given goal and the possibility to change the goal during operation, apply also to the proposed formulation.

1. Introduction

Reliable execution of robotic tasks in highly unstructured and dynamic scenarios is fundamental to bringing robots into human-inhabited environments. In such environments, robots need to accurately control their motion in free space as well as during physical interactions, which requires the capability to generate and adapt online reference behaviors in the form of motion, impedance, and/or force trajectories. Therefore, an effective encoding of diverse trajectory data is the key to spreading robotic solutions in everyday environments.

The Learning from Demonstration (LfD) paradigm [1] aims to develop learning solutions that allow the robot to enrich its skills via human guidance. Among the existing approaches [2,3], the idea of encoding robotic skills into stable dynamical systems has gained interest in the LfD community [4–6]. Dynamic Movement Primitives (DMPs) [7] are one of the first and most popular dynamical system-based approaches for LfD. DMPs are capable of encoding both discrete and periodic robotic skills into time-dependent systems. Discrete skills, also referred to as point-to-point motions, consist of motion trajectories with a fixed start and end point (goal) and are well-suited to represent many human daily tasks such as picking and placing objects.

The original DMP formulation considers one Degree of Freedom (DoF) trajectories. Multi-DoF trajectories are learned separately for each DoF and synchronized by a common phase variable. This strategy is effective for encoding independent skills like joint or Cartesian position trajectories, but it fails if the different DoFs are mutually dependent. This situation is common in robotics, where variables of interest may be interrelated by geometric constraints. Examples of such variables include: (*i*) orientation representations, like rotation matrices [8] or unit quaternions [8–10], and (*ii*) inertia [11], manipulability [12,13], stiffness, and damping [14,15] that are encapsulated in Symmetric Positive Definite (SPD) matrices. For variables interrelated by geometric constraints, the embedding strategy has to be modified to fulfill the constraints during both training and execution.

Several robotic skills consist of a combination of variables belonging to different manifolds. A simple example is a pose trajectory where the position lies in Cartesian space and the orientation is represented *e.g.,* as

* Corresponding author.
E-mail addresses: fabudakka@mondragon.edu (F.J. Abu-Dakka), matteo.saveriano@unitn.it (M. Saveriano), ville.kyrki@aalto.fi (V. Kyrki).

https://doi.org/10.1016/j.neucom.2024.128056
Received 7 August 2023; Received in revised form 6 June 2024; Accepted 10 June 2024
Available online 14 June 2024

Table 1
Comparison among the state-of-the-art of DMP-based approaches and our $\mathcal{G}$-DMP across different Riemannian manifolds: Euclidean space of dimension m $\mathcal{R}^m$, unit quaternion space $\mathcal{S}^3$, m-unit sphere manifold $\mathcal{S}^m$, 3D-rotation matrices space $\mathcal{SO}(3)$, special orthogonal group in m dimensions $\mathcal{SO}(m)$, and the space of $m \times m$ SPD matrices $\mathcal{S}^m_{++}$.

	$\mathcal{R}^m$	$\mathcal{S}^3$	$\mathcal{S}^m$	$\mathcal{SO}(3)$	$\mathcal{SO}(m)$	$\mathcal{S}^m_{++}$	Composite spaces *e.g.*, $\mathcal{S}^3 \times \mathcal{R}^3$
Ijspeert et al. [4,7]	✓	–	–	–	–	–	–
Ude et al. [8]	–	✓	–	✓	–	–	–
Koutras et al. [9], Abu-Dakka et al. [16], Saveriano et al. [18]	–	✓	–	–	–	–	–
Abu-Dakka et al. [17]	–	–	–	–	–	✓	–
Our $\mathcal{G}$-DMP	✓	✓	✓	✓	✓	✓	✓

unit quaternions. To avoid accuracy loss, Riemannian metrics should be embedded in the DMP formulation, allowing the consideration of all the constraints arising from various geometric structures in a unified and consistent manner. This is not possible with existing DMP formulations [4,8,9,16,17], which are space-dependent.

In this paper, we propose Geometry-aware DMP ($\mathcal{G}$-DMP), a new formulation that uses differential geometry to extend classical DMP for Euclidean data to other Riemannian manifolds. This extension allows discrete DMPs to effectively represent data evolving on different Riemannian manifolds, which subsequently allows the generation of smooth trajectories for data that do not belong to the Euclidean space. The formulation allows to encode various forms of point-to-point manipulation skills with specific geometric constraints in a unified and manifold independent manner. The general formulation provided in this paper can be applied to any trajectory of data by considering the corresponding Riemannian manifold. The effectiveness of the proposed approach is demonstrated both on synthetic data and physical experimental setups.

Preliminary results of this work have been published in [17], where we formulated DMP equations to learn SPD data profiles. This paper adds several significant novel contributions with respect to our published work:

1. A unified and mathematically principled framework, $\mathcal{G}$-DMP, that uses differential geometry to extend classical DMPs to any Riemannian manifold.
2. Exploitation of manifold composites to encode and learn composite manifolds in one single DMP formulation.
3. Proof of the stability of the proposed $\mathcal{G}$-DMP.
4. Formulation of $\mathcal{G}$-DMP goal switching without the need to use parallel transport.
5. An extensive evaluation and comparison with existing approaches.
6. Instructive and unified source codes accompany the paper with all necessary datasets at https://gitlab.com/geometry-aware/ga-dmp.

This paper is organized as follows: Next section presents the state-of-the-art. A background about standard DMPs and Riemannian geometry are given in Section 3. Afterwards, we provide the theoretical foundation of $\mathcal{G}$-DMPs in Section 4. Subsequently, we evaluate our approach in several experiments (Section 5). The work is concluded in Section 6.

2. Related works

LfD is a valuable framework to teach the robot new skills without explicitly coding them. LfD framework is effective in extracting relevant patterns from a few task demonstrations and in generalizing these patterns to different scenarios. LfD has been deeply investigated and several approaches have been developed in the literature. These include, among others, DMP [4,19], Probabilistic Movement Primitives (ProMP) [20], Gaussian Mixture Models (GMMs) [21], and Kernelized Movement Primitives (KMP) [10].

In many previous works, training data are simply treated as time series of Euclidean vectors. Other approaches, like [22], learn and adapt quaternion trajectories without enforcing the unit norm constraint, which leads to non-unit quaternions and hence requires an additional re-normalization step. Nevertheless, several works in the literature have investigated, to some extent, the problem of learning manipulation skills with specific geometric constraints. Examples of such skills include orientations, impedance, and manipulability matrices that are encapsulated in SPD matrices. The following paragraphs examine the state-of-the-art approaches.

DMP-based approaches: For instance, Abu-Dakka et al. extended the classical DMPs to encode discrete [16] and periodic [23] unit quaternion trajectories, while the work in [8] also considers different formulation to cope with rotation matrices. The quaternion-based DMPs were also extended to include the real-time goal switching mechanism [8]. The stability of the orientation DMPs is shown in [18]. In [9], authors proposed a modified formulation of unit quaternion DMPs to prevent oscillations that may arise in some cases. Abu-Dakka and Kyrki [17] reformulated DMPs to generate discrete SPD profiles, which is also able to adapt to a new goal-SPD-point. There is an important conceptual difference, about how we fit a curve to data points of a demonstration on a manifold, between $\mathcal{G}$-DMP and our previous work [17]. In [17], to fit a curve to data points $\{\mathbf{P}_t\}_{t=0}^{T}$ on a Riemannian manifold $\mathcal{M}$, we sought a curve $\gamma : [t_0, t_T] \rightarrow \mathcal{M}$ that passed exactly through each point of the demonstration trajectory. That assumption does not guarantee proximity between each pair of consecutive points, and, as detailed in Section 4.1, this led to the need to use parallel transport to accurately compute the *covariance derivative*. However, in this paper, inspired by [24], we look for γ to be sufficiently straight while passing sufficiently close to the data points at the given intervals. This lets us remove the parallel transport operation, i.e., to approximate the covariant derivative with the total derivative, resulting in a more compact formulation and a more efficient implementation of $\mathcal{G}$-DMP.

Finally, unlike our unified formulation, the formulations of all these previously mentioned approaches are space-specific and do not consider the possibility of treating data from different manifolds in a unified and consistent manner. Table 1 compares our proposed $\mathcal{G}$-DMP and the state-of-the-art of the DMP-based approaches.

Alternative approaches: Point-to-point motions are of particular interest in robotics as they form the basis of many everyday manipulation tasks. Therefore, researchers have developed approaches alternative to DMPs to represent point-to-point motions. Focusing on variable orientation profiles, [25] extended GMMs to represent the distribution of the quaternion displacements. Starting from this extended GMM, the work in [26] exploits the Riemannian structure of the unit sphere to encode variable orientations into a geometry-aware Task-Parameterized GMM (TP-GMM) [21]. KMP are extended to unit quaternions in [10] by projecting orientation data onto the tangent space of the unit sphere (which is locally Euclidean). Learning is performed in the tangent space and generated data are projected back to the manifold.

Table 2
Key notations. Indices, super/subscripts, constants, and variables have the same meaning over the entire text.

mathcal symbols *e.g.,* $\mathcal{M}$	$\triangleq$	denote manifolds.	bold mathcal symbols *e.g.,* $\boldsymbol{\mathcal{P}}$	$\triangleq$	denote trajectories.
capital letter variables *e.g.,* $\mathbf{P}$	$\triangleq$	denote points in a manifold.	small letter variables *e.g.,* $\mathbf{p}$	$\triangleq$	denote points in a tangent space.
$\mathcal{T}_{\mathbf{P}}\mathcal{M}$	$\triangleq$	The tangent space of a manifold $\mathcal{M}$ around a point $\mathbf{P}$	++	$\triangleq$	++
$\mathcal{R}^m$	$\triangleq$	Euclidean space of dimension m.	$\mathcal{S}^m$	$\triangleq$	Sphere manifold of dimension m.
$\mathcal{SO}(m)$	$\triangleq$	Special orthogonal group of dimension m.	$\mathcal{SE}(m)$	$\triangleq$	Special Euclidean group of dimension m.
$\mathcal{S}^m_{++}$	$\triangleq$	Space of $m \times m$ SPD.	$\mathcal{SYM}^m$	$\triangleq$	Space of $m \times m$ symmetric matrices.
N	$\triangleq$	# of nonlinear basis functions	i	$\triangleq$	index : $i = 1, 2, \dots, N$
l	$\triangleq$	index : $l = 1, 2, \dots, T$	T	$\triangleq$	Number of samples
$y, \dot{y}$	$\triangleq$	trajectory data and its 1st derivative in classical DMP	$z, \dot{z}$	$\triangleq$	scaled velocity and acceleration in $\mathcal{G}$-DMP
$\mathcal{Y}, \dot{\mathcal{Y}}$	$\triangleq$	trajectory data and its 1st derivative in $\mathcal{G}$-DMP	$\mathcal{Z}, \dot{\mathcal{Z}}$	$\triangleq$	scaled velocity and acceleration in $\mathcal{G}$-DMP
$\alpha_z, \beta_z, \alpha_x, \alpha_g$	$\triangleq$	Positive constant gains.	x	$\triangleq$	DMP phase variable.
$f(x)$, $\boldsymbol{\mathcal{F}}(x)$	$\triangleq$	forcing term for different spaces	w_i	$\triangleq$	adjustable weights
Ψ_i	$\triangleq$	Gaussian basis functions	c_i and h_i	$\triangleq$	centers and widths of Ψ_i
$g \in \mathbb{R}$ and $\mathbf{G} \in \mathcal{M}$	$\triangleq$	attractor point (goal) in different spaces	$\hat{\mathcal{Y}} \in \mathcal{M}$	$\triangleq$	new manifold trajectory generated by $\mathcal{G}$-DMP

SPD matrices are used to encapsulate data in many applications, including brain–computer interfaces [27], transfer learning [28], diffusion tensor imaging [29], as well as various robotic skills [30]. Alternative to DMP, the method in [31] used a tensor-based formulation of GMM and Gaussian Mixture Regression (GMR) on the SPD that enabled learning and reproducing skills involving SPD without additional data re-parametrization. Recently, [13] proposed a kernelized treatment to learn and adapt SPD profiles in the tangent space of the SPD manifold.

$\mathcal{G}$-DMP **vs. state-of-the-art**: The aforementioned geometry-aware formulations are space-specific and do not consider the possibility of treating data from different manifolds in a unified and consistent manner. On the contrary, our $\mathcal{G}$-DMP formulation is general and can be applied to any trajectory of data even when different DoFs belong to different spaces. Moreover, DMPs are one of the most popular LfD approaches and many robotics applications rely on them. In this respect, $\mathcal{G}$-DMP provides a useful framework to let users already familiar with DMPs to develop new applications.

3. Preliminaries

In this section, we briefly introduce the classical formulation of discrete DMPs (Section 3.1) and define fundamental operations on Riemannian manifolds (Section 3.2). Table 2 summaries the key notations used in this paper.

3.1. Dynamic movement primitives

DMP is composed of a system of nonlinear differential equations capable of encoding movements while guaranteeing convergence to a designated goal point (attractor) [19]. The foundational work on DMPs for discrete, point-to-point, motions was first introduced by Ijspeert et al. [7]. However, in order to generate movements adaptable to new situations without inducing excessive accelerations or amplification, Pastor et al. introduced some modifications [22]. In this paper, we adopt the formulation proposed by Pastor et al.. For a single DoF trajectory y, the DMP system of equations proposed in [22] is described as follows:

$$\tau \dot{z} = \alpha_z(\beta_z(g - y - (g - y_0)x + f(x)) - z), \tag{1}$$

$$\tau \dot{y} = z, \tag{2}$$

$$\tau \dot{x} = -\alpha_x x, \tag{3}$$

where τ is a positive scalar that represents the temporal scaling factor and determines the overall duration of the movement. $\dot{y}$ represents velocity and z denotes scaled velocity. x is a phase variable, governing the dynamical system's evolution towards the attractor point. It is used to avoid explicit time dependency in the formulation. The canonical system, given by (3), is initialized as $x(0) = 1$ and vanishes exponentially[1] as $t \to \infty$ if the gain $\alpha_x > 0$. β_z and α_z are positive gains that define the dynamical system's behavior. In order to ensure a critically damped system, we choose $\alpha_z = 4\beta_z$. The attractor (goal) point of the movement is denoted by g. This system of equations prevents high accelerations at the beginning of the motion or when the goal is close to the initial state, allowing for the reproduction of motions with the same initial and target states while preventing over-amplifications and trajectory mirroring effects when changing the goal.

The nonlinear forcing term $f(x)$ is classically parameterized as a linear combination of N nonlinear radial basis functions scaled by the phase variable x. $f(x)$ allows the dynamical system to preserve the shape of any smooth trajectory, and subsequently, generate this trajectory from an initial position y_0 to the attractor g. Thus, $f(x)$ is defined as:

$$f(x) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x, \tag{4}$$

$$\Psi_i(x) = \exp\left(-h_i \left(x - c_i\right)^2\right), \tag{5}$$

where w_i are the weights adjusted based on measured data to achieve the desired behavior. $\Psi_i(x)$ are Gaussian basis functions with centers c_i and widths h_i. For a given number of basis functions N, centers c_i and widths h_i are defined as follows:

$$c_i = \exp\left(-\alpha_x \frac{i-1}{N-1}\right), \quad h_i = \frac{1}{(c_{i+1} - c_i)^2}, \quad h_N = h_{N-1}$$

where $i = 1, \dots, N$. For each DoF.

In order to control multiple DoFs systems, such as trajectories of joint angles of D DoF manipulator, we consider a separate transformation system (1)–(2) for each of the D DoFs to control. Additionally, we utilize a single canonical system (3) shared across the D transformation systems, which synchronizes their time evolution.

3.2. Riemannian manifolds

An m-dimensional manifold is a topological space where each point locally resembles Euclidean space $\mathcal{R}^m$. A differentiable manifold extends this notion to ensure that at each point, there exists a tangent

[1] The minimum phase to execute a motion within T_f seconds can be computed through $x(T_f) = \exp(-\frac{\alpha_x}{\tau} T_f)$.

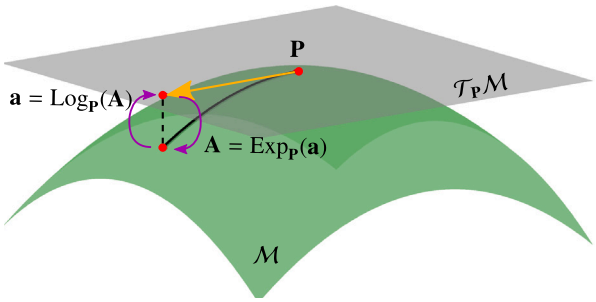


Fig. 1. A Riemannian manifold $\mathcal{M}$ and its tangent space $\mathcal{T}_{\mathbf{P}}\mathcal{M}$ defined at point $\boldsymbol{P}$.

space. A Riemannian manifold $\mathcal{M}$ is a smooth and differentiable manifold where each tangent space is equipped with a Riemannian metric tensor. This tensor, denoted as $\langle\cdot,\cdot\rangle_{\mathbf{P}}$, is a positive definite inner product defined on the tangent space $\mathcal{T}_{\mathbf{P}}\mathcal{M}$ for every point $\mathbf{P}\in\mathcal{M}$. The Riemannian metric introduces the concept of length on the manifold. By utilizing this metric, we can generalize the notion of a "straight line" between two points by defining a geodesic as the shortest curve that connects two points on a manifold. This geodesic allows for the transportation of vectors between tangent spaces [32,33]. A geodesic $\gamma(t)$ is defined as a continuously differentiable curve that connects points $\boldsymbol{A},\boldsymbol{B}$ on the manifold $\mathcal{M}$. It locally minimizes the distance between these points, and its length is given by the functional:

$$\mathcal{L}_{\boldsymbol{A}}^{\boldsymbol{B}}(\gamma)=\int_0^1\langle\dot{\gamma}(t),\dot{\gamma}(t)\rangle\,\mathrm{d}t. \tag{6}$$

The distance between points $\boldsymbol{A}$ and $\boldsymbol{B}$ is then defined by minimizing (6), *i.e.,*

$$\mathrm{dist}(\boldsymbol{A},\boldsymbol{B})=\min\mathcal{L}_{\boldsymbol{A}}^{\boldsymbol{B}}(\gamma) \tag{7}$$

3.2.1. Mapping operators

The tangent spaces and their bases provide the ability to perform linear algebra. In order to perform computations on the manifold while preserving distances, a mapping system is needed to switch between the tangent space $\mathcal{T}_{\mathbf{P}}\mathcal{M}$ and the manifold $\mathcal{M}$, see Fig. 1. These mapping operators are:

- **The logarithmic map** $\left(\mathrm{Log}_{\mathbf{P}}(\cdot)\right)$ is a function that maps a point $\boldsymbol{A}\in\mathcal{M}$ into a point $\boldsymbol{a}\in\mathcal{T}_{\mathbf{P}}\mathcal{M}$ (see Fig. 1). It is defined as:

$$\mathrm{Log}_{\mathbf{P}}(\cdot):\mathcal{M}\mapsto\mathcal{T}_{\mathbf{P}}\mathcal{M}, \tag{8}$$

- **The exponential map** $\left(\mathrm{Exp}_{\mathbf{P}}(\cdot)\right)$ is the inverse of the logarithmic map. It maps a point $\boldsymbol{a}\in\mathcal{T}_{\mathbf{P}}\mathcal{M}$ in the tangent space of $\boldsymbol{P}$ to a point $\boldsymbol{A}\in\mathcal{M}$ such that $\boldsymbol{A}$ lies on the geodesic starting from $\mathbf{P}$ in the direction of $\boldsymbol{a}$ with distance of $\|\boldsymbol{a}\|=\langle\boldsymbol{a},\boldsymbol{a}\rangle_{\mathbf{P}}$ (see Fig. 1). It is defined as:

$$\mathrm{Exp}_{\mathbf{P}}(\cdot):\mathcal{T}_{\mathbf{P}}\mathcal{M}\mapsto\mathcal{M}, \tag{9}$$

3.2.2. Cartesian products in Riemannian geometry

In Riemannian geometry, the Cartesian product of two Riemannian manifolds $\mathcal{M}$ and $\mathcal{N}$ is also a manifold denoted as $\mathcal{M}\times\mathcal{N}$. This construction allows us to combine the geometric structures of both $\mathcal{M}$ and $\mathcal{N}$ into a single manifold.

For any points $\mathbf{P}_1\in\mathcal{M}$ and $\boldsymbol{U}_1\in\mathcal{N}$, and their corresponding tangent vectors $\boldsymbol{p}_1\in\mathcal{T}_{\mathbf{P}_1}\mathcal{M}$ and $\boldsymbol{u}_1\in\mathcal{T}_{\boldsymbol{U}_1}\mathcal{N}$, the tangent space of $\mathcal{M}\times\mathcal{N}$ at the point $(\mathbf{P}_1,\boldsymbol{U}_1)$ is isomorphic to the direct sum of the tangent spaces of $\mathcal{M}$ and $\mathcal{N}$:

$$\mathcal{T}_{(\mathbf{P}_1,\boldsymbol{U}_1)}(\mathcal{M}\times\mathcal{N})\cong\mathcal{T}_{\mathbf{P}_1}\mathcal{M}\oplus\mathcal{T}_{\boldsymbol{U}_1}\mathcal{N}, \tag{10}$$

This means that any tangent vector at $(\mathbf{P}_1,\boldsymbol{U}_1)$ can be uniquely decomposed into a pair of tangent vectors, one in $\mathcal{T}_{\mathbf{P}_1}\mathcal{M}$ and the other in $\mathcal{T}_{\boldsymbol{U}_1}\mathcal{N}$.

Table 3
Re-interpretation of basic standard operations in a Riemannian manifold [34].

	Euclidean space	Riemannian manifold
Subtraction	$\overrightarrow{\mathbf{p}_1\mathbf{p}_2}=\mathbf{p}_2-\mathbf{p}_1$	$\overrightarrow{\mathbf{P}_1\mathbf{P}_2}=\mathrm{Log}_{\mathbf{P}_1}(\mathbf{P}_2)$
Addition	$\mathbf{p}_2=\mathbf{p}_1+\overrightarrow{\mathbf{p}_1\mathbf{p}_2}$	$\mathbf{P}_2=\mathrm{Exp}_{\mathbf{P}_1}\left(\overrightarrow{\mathbf{P}_1\mathbf{P}_2}\right)$
Distance	$\mathrm{dist}(\mathbf{p}_1,\mathbf{p}_2)=\|\mathbf{p}_2-\mathbf{p}_1\|$	$\mathrm{dist}(\mathbf{P}_1,\mathbf{P}_2)=\|\overrightarrow{\mathbf{P}_1\mathbf{P}_2}\|_{\mathbf{P}_1}$
Interpolation	$\mathbf{p}(t)=\mathbf{p}_1+t\overrightarrow{\mathbf{p}_1\mathbf{p}_2}$	$\mathbf{P}(t)=\mathrm{Exp}_{\mathbf{P}_1}\left(t\overrightarrow{\mathbf{P}_1\mathbf{P}_2}\right)$

To facilitate computations on the Cartesian product manifold $\mathcal{M}\times\mathcal{N}$, we require to redefine the mapping operators in (8) and (9) as follows:

$$\mathrm{Log}_{(\mathbf{P}_1,\boldsymbol{U}_1)}(\mathbf{P}_2,\boldsymbol{U}_2):\mathcal{M}\times\mathcal{N}\mapsto\mathcal{T}_{(\mathbf{P}_1,\boldsymbol{U}_1)}(\mathcal{M}\times\mathcal{N}), \tag{11}$$

$$\mathrm{Exp}_{(\mathbf{P}_1,\boldsymbol{U}_1)}(\boldsymbol{p},\boldsymbol{u}):\mathcal{T}_{(\mathbf{P}_1,\boldsymbol{U}_1)}(\mathcal{M}\times\mathcal{N})\mapsto\mathcal{M}\times\mathcal{N}. \tag{12}$$

This leads to

$$\mathrm{Log}_{(\mathbf{P}_1,\boldsymbol{U}_1)}(\mathbf{P}_2,\boldsymbol{U}_2)=\mathrm{Log}_{\begin{bmatrix}\mathbf{P}_1\\\boldsymbol{U}_1\end{bmatrix}}\left(\begin{bmatrix}\mathbf{P}_2\\\boldsymbol{U}_2\end{bmatrix}\right)=\begin{bmatrix}\mathrm{Log}_{\mathbf{P}_1}(\mathbf{P}_2)\\\mathrm{Log}_{\boldsymbol{U}_1}(\boldsymbol{U}_2)\end{bmatrix},$$

$$\mathrm{Exp}_{(\mathbf{P}_1,\boldsymbol{U}_1)}(\boldsymbol{p},\boldsymbol{u})=\mathrm{Exp}_{\begin{bmatrix}\mathbf{P}_1\\\boldsymbol{U}_1\end{bmatrix}}\left(\begin{bmatrix}\boldsymbol{p}\\\boldsymbol{u}\end{bmatrix}\right)=\begin{bmatrix}\mathrm{Exp}_{\mathbf{P}_1}(\mathbf{P}_2)\\\mathrm{Exp}_{\boldsymbol{U}_1}(\boldsymbol{U}_2)\end{bmatrix}.$$

where $(\boldsymbol{p},\boldsymbol{u})\in\mathcal{T}_{(\mathbf{P}_1,\boldsymbol{U}_1)}(\mathcal{M}\times\mathcal{N})$ and $(\mathbf{P}_2,\boldsymbol{U}_2)\in\mathcal{M}\times\mathcal{N}$.

3.2.3. Computing in Riemannian manifolds

Let $\mathbf{P}_1,\mathbf{P}_2\in\mathcal{M}$ and $\mathbf{p}_1,\mathbf{p}_2\in\mathcal{R}^m$, then the reinterpretation of basic standard operations (*e.g.,* addition and subtraction) in a Riemannian manifold are summarized in Table 3.

3.2.4. Riemannian geometric mean

Given a set of points $\{\mathbf{P}_i\}_{i=1}^n\in\mathcal{M}$ and a geodesic distance $\mathrm{dist}(\mathbf{P}_j,\mathbf{P}_i)$ between two points in $\mathcal{M}$, the Fréchet mean [35] is estimated by minimizing the sum of squared geodesic distances

$$\overline{\mathbf{P}}=\arg\min_{\mathbf{P}\in\mathcal{M}}\sum_{i=1}^{N}\mathrm{dist}^2(\mathbf{P},\mathbf{P}_i), \tag{13}$$

This estimation can be efficiently computed iteratively by following Alg. 1 [35].

Algorithm 1 Intrinsic mean

Initialization: $\overline{\mathbf{P}}=\mathbf{P}_1$
1: **while** $\|\boldsymbol{a}\|<\delta$ **do**
2: $\quad\boldsymbol{a}=\frac{1}{N}\sum_{i=1}^{N}\mathrm{Log}_{\overline{\mathbf{P}}}(\mathbf{P}_i)$
3: $\quad\overline{\mathbf{P}}=\mathrm{Exp}_{\overline{\mathbf{P}}}(\epsilon\boldsymbol{a})\,;\ \epsilon\le1$
4: **end while**

4. Proposed approach

In this section, we provide a generalized and unified formulation for DMPs based on Riemannian geometry in order to learn and adapt robot manipulation skills regardless its corresponding space, for example orientation trajectories $\left(\mathcal{SO}(3)\text{ or }\mathcal{S}^3\right)$, pose data $(\mathcal{SE}(3))$, and SPD profiles $(\mathcal{S}_{++}^m)$ such as stiffness, manipulability, inertia. We also show that our $\mathcal{G}$-DMP inherits desirable properties of the original formulation like convergence to a target and goal switching.

4.1. Geometry-aware DMPs formulation

In this section, we introduce the mathematical foundations of $\mathcal{G}$-DMP technique. The $\mathcal{G}$-DMP formulation offers a comprehensive and cohesive approach to encode and execute a discrete trajectory $\mathcal{Y}=\{t_l,\boldsymbol{Y}_l\}_{l=0}^T$, commonly known as a point-to-point trajectory, which evolves within the confines of a Riemannian manifold $\mathcal{M}$, where

each $\boldsymbol{Y}_l \in \mathcal{M}$. Its attractor dynamics on the manifold guarantee the convergence of $\mathcal{Y}$ toward a goal $\boldsymbol{G} \in \mathcal{M}$ regardless of the initial starting point $\boldsymbol{Y}_0$. To achieve this, it is necessary to transform the classical DMP system described by (1)–(2) into a unified geometry-aware formulation utilizing principles from Riemannian geometry. In pursuit of this objective, we initiate the process by considering the expression of a general second-order system evolving on a manifold, as outlined by Fiori et al. [36]

$$\tau \nabla_{\mathcal{Z}} \mathcal{Z} = \boldsymbol{h}(\mathcal{Z}, \mathcal{Y}, x), \tag{14}$$

$$\tau \dot{\mathcal{Y}} = \mathcal{Z}, \tag{15}$$

where $\mathcal{Z}$ and $\dot{\mathcal{Z}}$ represent the scaled first and second derivatives of $\mathcal{Y}$. The phase variable x is similar to the one defined in (1) and (3). The *covariant derivative* $\nabla_{\mathcal{Z}} \mathcal{Z}$ can be defined from the total derivative $\dot{\mathcal{Z}}$ using parallel transport [17,36]. However, computing the parallel transport is, in general, time-consuming. Assuming that consecutive points on the manifold are sufficiently close, and the geodesic between them approximates a straight line, the covariant derivative can be well approximated by manifold-valued finite differences [24,37]. This approximation significantly simplifies the computation process while introducing negligible errors. Thus, in this work, we consider the approximation $\nabla_{\mathcal{Z}} \mathcal{Z} \approx \dot{\mathcal{Z}}$. The function $\boldsymbol{h}(\cdot)$ may encompass multiple additive contributions. In this study, we assume that

$$\boldsymbol{h}(\mathcal{Z}, \mathcal{Y}, x) = \alpha_z \left(\beta_z \left(\mathrm{Log}_{\mathcal{Y}}(\boldsymbol{G}) - \mathrm{Log}_{\boldsymbol{Y}_0}(\boldsymbol{G})\, x + \boldsymbol{\mathcal{F}}(x) \right) - \mathcal{Z} \right), \tag{16}$$

where $\boldsymbol{G} \in \mathcal{M}$ is the goal point. The function $\mathrm{Log}_{\mathcal{Y}}(\cdot)$ is defined in (8). Additionally, positive gains α_z and β_z are introduced. The term $-\alpha_z \mathcal{Z}$ represents a *dissipative force* that plays a similar role to damping in a mechanical system. The term $\alpha_z(\beta_z \mathrm{Log}_{\mathcal{Y}}(\boldsymbol{G}))$ corresponds to *conservative force* and can be interpreted as the negative gradient of a potential. This can be demonstrated by considering that $-\frac{1}{2}\nabla_{\mathcal{Y}} \mathrm{dist}^2(\mathcal{Y}, \boldsymbol{G}) = \mathrm{Log}_{\mathcal{Y}}(\boldsymbol{G})$ [36], where $\mathrm{dist}(\cdot,\cdot)$ denotes the Riemannian distance. Finally, the term $\boldsymbol{\mathcal{F}}(x)$ represents a phase-dependent forcing term which is learned from the demonstration and will be further discussed in this section.

Consequently, we can redefine the dynamic system presented in (1)–(2) as follows

$$\tau \dot{\mathcal{Z}} = \alpha_z \left(\beta_z \left(\mathrm{Log}_{\mathcal{Y}}(\boldsymbol{G}) - \mathrm{Log}_{\boldsymbol{Y}_0}(\boldsymbol{G})\, x + \boldsymbol{\mathcal{F}}(x) \right) - \mathcal{Z} \right). \tag{17}$$

$$\tau \dot{\mathcal{Y}} = \mathcal{Z}, \tag{18}$$

The forcing term $\boldsymbol{\mathcal{F}}(x)$ is defined as follows

$$\boldsymbol{\mathcal{F}}(x) = \frac{\sum_{i=1}^{N} \boldsymbol{w}_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x, \tag{19}$$

where $\boldsymbol{w}_i \in \mathcal{R}^{m \times N}$ are the weights (free parameters) that can be estimated by encoding any sampled trajectory (*e.g.,* any robot manipulation skill profile). In order to estimate the parameters of a corresponding $\mathcal{G}$-DMPs, we need to estimate the 1st and 2nd time derivatives of the demonstrated trajectory. The 1st time derivative is computed as follows

$$\dot{\mathcal{Y}} = \left\{ \left(\mathrm{Log}_{\boldsymbol{Y}_{l-1}} (\boldsymbol{Y}_l) \right) / \delta t \right\}_{l=1}^{T} \in \mathcal{T}_{\boldsymbol{Y}_{l-1}} \mathcal{M}, \tag{20}$$

where $\delta t = t_l - t_{l-1}$. The 2nd-time-derivative $\ddot{\mathcal{Y}}$ can be computed straight forward from $\dot{\mathcal{Y}}$ using standard Euclidean tools, *i.e.,* $\ddot{\mathcal{Y}} = \{t_l, \ddot{\boldsymbol{y}}_l\}_{l=1}^{T}$ where $\ddot{\boldsymbol{y}}_l = (\dot{\boldsymbol{y}}_l - \dot{\boldsymbol{y}}_{l-1})/\delta t$.

Having all necessary data $\{t_l, \boldsymbol{Y}_l, \dot{\boldsymbol{y}}_l, \ddot{\boldsymbol{y}}_l\}$, and by inverting (17), the parameters $\boldsymbol{w}_i$ and the approximated desired shape of the demonstration are estimated as follows

$$\frac{\sum_{i=1}^{N} \boldsymbol{w}_i \Psi_i(x_l)}{\sum_{i=1}^{N} \Psi_i(x_l)} x_l = \frac{\tau^2 \ddot{\boldsymbol{y}}_l + \alpha_z \tau \dot{\boldsymbol{y}}_l}{\alpha_z \beta_z} - \mathrm{Log}_{\boldsymbol{Y}_l}(\boldsymbol{G}) + \mathrm{Log}_{\boldsymbol{Y}_0}(\boldsymbol{G})\, x \tag{21}$$

Using (21), the weights $\boldsymbol{w}_i$ can be estimated by encoding any sampled robot manipulation skill data.

In the reproduction, Eq. (18) is integrated using the forward Euler–Riemann stepping method [36] as

$$\hat{\mathcal{Y}}(t + \delta t) = \mathrm{Exp}_{\boldsymbol{Y}_t} \left(\mathcal{Z}(t) \frac{\delta t}{\tau} \right), \tag{22}$$

where $\hat{\mathcal{Y}} \in \mathcal{M}$ represents the new robot manipulation skills data. Eq. (22) is manifold dependent. $\mathrm{Exp}_{\boldsymbol{Y}_t}(\cdot)$ is defined as in (9), and we refer to the appendix for the expression of $\mathrm{Exp}_{\boldsymbol{Y}_t}(\cdot)$ for the manifolds used in this work.

In case the manifold is a Lie group, the expression of a general second-order system on a Lie group becomes [36]

$$\tau \dot{\mathcal{Z}} = \boldsymbol{h}(\mathcal{Z}, \mathcal{Y}, x), \tag{23}$$

$$\tau \dot{\mathcal{Y}} = \boldsymbol{g}(\mathcal{Z}, \mathcal{Y}), \tag{24}$$

from which is straightforward to derive that

$$\tau \dot{\mathcal{Z}} = \alpha_z \left(\beta_z \left(\mathrm{Log}\left(\boldsymbol{Y}_g * \mathcal{Y}^{-1} \right) - \mathrm{Log}\left(\boldsymbol{Y}_g * \boldsymbol{Y}_0^{-1} \right) + \boldsymbol{\mathcal{F}}(x) \right) - \mathcal{Z} \right), \tag{25}$$

$$\tau \dot{\mathcal{Y}} = \boldsymbol{g}(\mathcal{Z}, \mathcal{Y}). \tag{26}$$

Eq. (25) is formally the same as (17), provided we use the logarithmic map $\mathrm{Log}_{\mathcal{Y}}(\cdot) = \mathrm{Log}\left(\boldsymbol{Y}_g * \mathcal{Y}^{-1} \right)$ defined using Lie group theory. The term $\boldsymbol{m}(\cdot)$ in (26) is the *inverse left translation*, which maps a tangent vector from the Lie algebra to the tangent space at $\boldsymbol{Y}_t$ and depends on the specific Lie group. The expressions of $\boldsymbol{g}(\cdot)$ and $\mathrm{Log}(\cdot)$ for unit quaternions and rotation matrices, two Lie groups commonly used in robotics, are given in Appendices A.3 and A.5.

As a remark, we used the Riemannian formulation (17)–(18) in the rest of the paper. However, for the sake of completeness, we also have provided a formulation for Lie groups in (25)–(26).

4.2. Goal switching

In many real scenarios, while the robot executes its trajectory, it may encounter situations where it needs to adapt its trajectory to a new goal, *e.g.,* new pick-up point, on the fly. This change of goal, referred to as goal switching, is a common requirement in dynamic environments. In order to achieve smooth transitions between goals and avoid unnecessary jumps, the authors of [4] suggested adding an extra first-order differential equation to gradually transition the current goal g to the new goal g_{new} over time. This differential equation can be written as

$$\tau \dot{g} = \alpha_g (g_{new} - g), \tag{27}$$

where $\alpha_g > 0$ is a positive constant gain. The gradual transition in (27) ensures that the robot's behavior remains continuous and responsive to changes in its task environment.

Analogously, Riemannian manifold-based $\mathcal{G}$-DMP can switch the goal using

$$\tau \dot{\boldsymbol{\mathcal{G}}} = \alpha_g \mathrm{Log}_{\boldsymbol{G}} \left(\boldsymbol{G}_{new} \right). \tag{28}$$

Eq. (28) allows to continuously update $\boldsymbol{G}$ until it smoothly reaches the new value $\boldsymbol{G}_{new} \in \mathcal{M}$.

4.3. Stability analysis

Theorem 1 states the stability conditions of the geometry-aware DMP formulation in Section 4.1.

Theorem 1. *Assume that $\boldsymbol{\mathcal{F}}(x) \to 0$ for $t \to +\infty$ and that the gains $\alpha_z, \beta_z > 0$. Under these assumptions, the geometry-aware DMP has a globally (in its domain of definition) asymptotically stable equilibrium at $(\boldsymbol{G}, \boldsymbol{0})$.*

Proof. Recall that, by assumption, we restrict the domain to the points where the logarithmic map $\text{Log}_{\mathcal{Y}}(\boldsymbol{G})$ is uniquely defined. Recall also that the forcing term $\boldsymbol{\mathcal{F}}(x)$ in (17) is a weighted sum of Gaussian basis functions. Therefore, the non-linear terms in (17) and (18) are smooth and uniquely defined functions. Consider also that the time dependency introduced by x vanishes for $t \to +\infty$. Hence, (17) and (18) are an *asymptotically autonomous differential system* and the stability can be proved by analyzing its asymptotic behavior [38]. This allows us to neglect the terms $\boldsymbol{\mathcal{F}}(x)$ and $\text{Log}_{\boldsymbol{Y}_0}(\boldsymbol{G})\,x$ in the stability analysis and to focus on the asymptotic dynamics

$$\dot{\mathcal{Z}} = \alpha_z \beta_z \text{Log}_{\mathcal{Y}}(\boldsymbol{G}) - \alpha_z \mathcal{Z}, \tag{29}$$

$$\dot{\mathcal{Y}} = \mathcal{Z}, \tag{30}$$

where we set $\tau = 1$ without loss of generality.

We first show that $(\boldsymbol{G}, \boldsymbol{0})$ is an equilibrium point of the system (29) and (30). The right side of (30) vanishes only for $\boldsymbol{Z} = \boldsymbol{0}$. With $\boldsymbol{Z} = \boldsymbol{0}$, the right side of (29) vanishes only for $\text{Log}_{\boldsymbol{Y}}(\boldsymbol{G}) = \boldsymbol{0} \Leftrightarrow \boldsymbol{Y} = \boldsymbol{G}$. This implies that the system (29) and (30) has a unique equilibrium point at $(\boldsymbol{G}, \boldsymbol{0})$.

We now show that the equilibrium $(\boldsymbol{G}, \boldsymbol{0})$ is a global attractor in the chart where the logarithmic map $\text{Log}_{\mathcal{Y}}(\boldsymbol{G})$ is uniquely defined. To this end, we define the candidate Lyapunov function

$$V(\mathcal{Y}, \mathcal{Z}) = \text{dist}^2(\mathcal{Y}, \boldsymbol{G}) + \frac{1}{\alpha_z \beta_z} \langle \mathcal{Z}, \mathcal{Z} \rangle_{\mathcal{Y}}, \tag{31}$$

where $\text{dist}(\cdot,\cdot)$ is the Riemannian distance defined as in (7) and $\langle \cdot, \cdot \rangle_{\mathcal{Y}}$ is the positive definite inner product (see Section 3.2). $V(\mathcal{Y}, \mathcal{Z})$ is positive definite everywhere if $\alpha_z \beta_z > 0$ and vanishes only at $\boldsymbol{Y} = \boldsymbol{G}$ ($\text{dist}^2(\boldsymbol{G}, \boldsymbol{G}) = 0$) and $\boldsymbol{Z} = \boldsymbol{0}$ ($\langle \boldsymbol{0}, \boldsymbol{0} \rangle_{\mathcal{Y}} = 0$). To show that $V(\mathcal{Y}, \mathcal{Z})$ is a valid Lyapunov function we need to show that its time derivative is negative definite and vanishes at $(\boldsymbol{G}, \boldsymbol{0})$. The time derivative of $V(\mathcal{Y}, \mathcal{Z})$ can be written as

$$\begin{aligned} \dot{V}(\mathcal{Y}, \mathcal{Z}) &= \frac{d}{dt} \text{dist}^2(\mathcal{Y}, \boldsymbol{G}) + \frac{1}{\alpha_z \beta_z} \frac{d}{dt} \langle \mathcal{Z}, \mathcal{Z} \rangle_{\mathcal{Y}} \\ &= -2 \langle \text{Log}_{\mathcal{Y}}(\boldsymbol{G}), \dot{\mathcal{Y}} \rangle_{\mathcal{Y}} + \frac{2}{\alpha_z \beta_z} \langle \dot{\mathcal{Z}}, \mathcal{Z} \rangle_{\mathcal{Y}} \end{aligned} \tag{32}$$

where we used the expression $\frac{d}{dt} \text{dist}^2(\mathcal{Y}, \boldsymbol{G}) = -2 \langle \text{Log}_{\mathcal{Y}}(\boldsymbol{G}), \dot{\mathcal{Y}} \rangle_{\mathcal{Y}}$ from [39] and the bi-linearity and the symmetry of the interior product to write $\frac{d}{dt} \langle \mathcal{Z}, \mathcal{Z} \rangle_{\mathcal{Y}} = 2 \langle \dot{\mathcal{Z}}, \mathcal{Z} \rangle_{\mathcal{Y}}$. By replacing $\dot{\mathcal{Z}}$ from (29) and $\dot{\mathcal{Y}}$ from (30) into (32), we obtain

$$\begin{aligned} \dot{V}(\mathcal{Y}, \mathcal{Z}) &= -2 \langle \text{Log}_{\mathcal{Y}}(\boldsymbol{G}), \mathcal{Z} \rangle_{\mathcal{Y}} + 2 \langle \text{Log}_{\mathcal{Y}}(\boldsymbol{G}), \mathcal{Z} \rangle_{\mathcal{Y}} \\ &\quad - \frac{2}{\beta_z} \langle \mathcal{Z}, \mathcal{Z} \rangle_{\mathcal{Y}} = -\frac{2}{\beta_z} \langle \mathcal{Z}, \mathcal{Z} \rangle_{\mathcal{Y}} \leq 0, \end{aligned}$$

where the last inequality holds if $\beta_z > 0$. Therefore, $\dot{V}(\mathcal{Y}, \mathcal{Z}) \leq 0$ everywhere in the chart and vanishes only at $\boldsymbol{Z} = \boldsymbol{0}$. The LaSalle's invariance theorem [40] allows to conclude the stability of (29)–(30). □

Remark 1. The results of Theorem 1 hold where the logarithmic map is uniquely defined, *e.g.,* $\mathcal{T}_{\boldsymbol{Y}_{l-1}} \mathcal{M}$ can be extended as much as it will not contain points conjugate to $\boldsymbol{Y}_{l-1}$ [41]. For manifolds with no cut-locus, this holds everywhere. Hence, Theorem 1 is globally valid on manifolds with no cut-locus (*e.g.,* the manifold of SPD matrices with positive definite eigenvalues [34]). However, for manifolds with cut-locus (*e.g.,* unit m-sphere manifolds [32]), the logarithmic map $\text{Log}_{\mathcal{Y}}(\boldsymbol{G})$ is defined in a region that does not contain points conjugate to $\boldsymbol{G}$. For the unit m-sphere, the logarithmic map $\text{Log}_{\mathcal{Y}}(\boldsymbol{G})$ is uniquely defined everywhere apart from the antipodal point $-\boldsymbol{G}$.

For illustration, we used the proposed $\mathcal{G}$-DMP to learn two trajectories; (*i*) the "N" shape on $\mathcal{S}^2$ provided in [30] (Fig. 2a), and (*ii*) a "C" curve with π diameter (Fig. 2b). The "N" trajectory covers both the north and south hemispheres and, as shown in [30], working on the Lie algebra will introduce large distortions. Moreover, the "N" shape is an antipodal free trajectory, such that $\mathcal{Y} = \{\boldsymbol{Y}_l\}_{l=1}^{T-1} \in \mathcal{S}^2 \mid |\boldsymbol{Y}_l \cdot \boldsymbol{G}| < 1$. However, the "C" curve includes the antipodal of $\boldsymbol{G}$. Fig. 2a shows

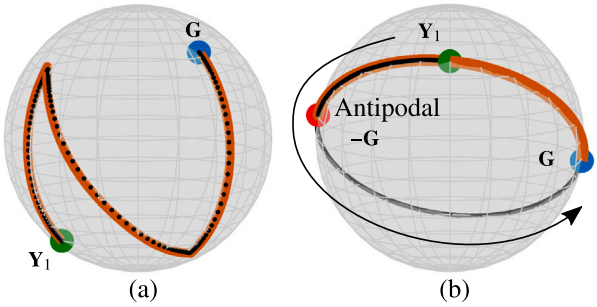


Fig. 2. Results of $\mathcal{G}$-DMP while learning and producing trajectories that cover both south and north hemispheres. Black dashed curves denote demonstrations, while brown curves represent reproduction. Green point $\boldsymbol{Y}_1$ denotes the starting point of the trajectory, while the blue one indicates the goal $\boldsymbol{G}$. The red point illustrates the antipodal point of the goal. The figure shows $\mathcal{G}$-DMP while executing a trajectory that (a) does not contain an antipodal of the goal $\boldsymbol{G}$, and (b) contains an antipodal of the goal.

$\mathcal{G}$-DMP successfully reproducing the shape and converges to the goal (blue point). However, in (b), it fails to follow the trajectory when it encounters the antipodal of the goal (point in red). $\mathcal{G}$-DMP is supposed to follow the trajectory in the direction of the black arrow starting from the green point. However, it follows the trajectory until the antipodal point, then returns back to reach the goal from the opposite direction. A possible way to solve this issue is to split the trajectory into segments. For the example in Fig. 2b, this can be done by splitting the trajectory into 2 segments, namely $\boldsymbol{Y}_1$ to $\boldsymbol{Y}_2$, and $\boldsymbol{Y}_2$ to $\boldsymbol{G}$, where $\boldsymbol{Y}_2$ is any point in the demonstration between $-\boldsymbol{G}$ and $-\boldsymbol{Y}_1$. One can then fit 2 separate $\mathcal{G}$-DMP and smoothly merge them [18].

4.4. $\mathcal{G}$-DMP on Riemannian manifold products

Let us define $\mathcal{Y} \in \mathcal{M}$ and $\mathcal{U} \in \mathcal{N}$ as two arbitrary trajectories from two Riemannian manifolds $\mathcal{M}$ and $\mathcal{N}$, respectively. Let us call $\boldsymbol{\mathcal{H}} = \{t_l, (\boldsymbol{Y}_l, \boldsymbol{U}_l)\}_{l=1}^{T}$ the set of data points in one demonstration. We can now define the *composite* $\mathcal{G}$-DMP as

$$\tau \dot{\mathcal{V}} = \alpha_z (\beta_z \text{Log}_{(\mathcal{Y},\mathcal{U})} (\boldsymbol{G}_{\boldsymbol{Y}}, \boldsymbol{G}_{\boldsymbol{U}}) - \mathcal{V}) + \boldsymbol{\mathcal{F}}(x), \tag{33}$$

$$\tau \dot{\boldsymbol{\mathcal{H}}} = \mathcal{V}, \tag{34}$$

where $\mathcal{V} \in \mathcal{T}_{(\boldsymbol{Y}_l, \boldsymbol{U}_l)}(\mathcal{M} \times \mathcal{N})$ and $\text{Log}_{(\mathcal{Y},\mathcal{U})} (\boldsymbol{G}_{\boldsymbol{Y}}, \boldsymbol{G}_{\boldsymbol{U}})$ is the logarithmic map that maps the attractors $\boldsymbol{G}_{\boldsymbol{Y}} \in \mathcal{M}$ and $\boldsymbol{G}_{\boldsymbol{U}} \in \mathcal{N}$ from the manifold composite $\mathcal{M} \times \mathcal{N}$ to the tangent space $\mathcal{T}_{(\boldsymbol{Y}_l, \boldsymbol{U}_l)}(\mathcal{M} \times \mathcal{N})$ at each time-step.

As an illustrative example, consider the pose of the end-effector of a robot, which can be represented as the Cartesian product of the hypersphere $\mathcal{S}^3$ and 3D-Euclidean space $\mathcal{R}^3$, *i.e.,* $\mathcal{H} = \mathcal{S}^3 \times \mathcal{R}^3$. It is worth mentioning that the pose of the end-effector of a robot can be alternatively represented as a homogeneous transformation matrix $\boldsymbol{H} \in \mathcal{SE}(3)$ using the Lie group theory formulation [42]; however, in this work, we exploit the Cartesian product property of Riemannian manifolds.

Remark 2. The stability of manifold composites $\mathcal{G}$-DMP formulation in (33) and (34) can be straightforwardly proven by applying Theorem 1 separately to $\mathcal{M}$ and $\mathcal{N}$.

5. Validation

We validated the proposed $\mathcal{G}$-DMP in simulation as well as in real setups. More in detail, we performed the following evaluations:

- In simulation:
  - We augmented two public datasets; 2D-LASA handwriting dataset [5] and 2D-Letters handwriting dataset [30] with

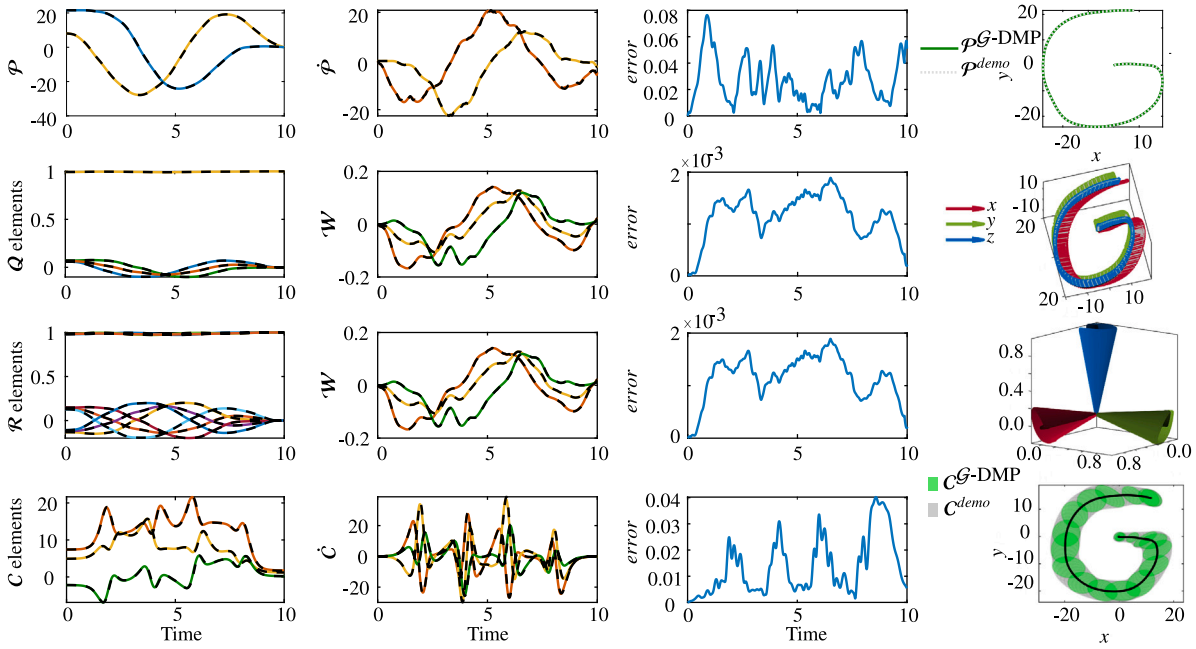


Fig. 3. Illustrates the performance of $\mathcal{G}$-DMP when executing Riemannian LASA dataset. 1st *row*: Euclidean 2D trajectory. 2nd *row*: Unit quaternion trajectory. *3rd row:* Rotation matrix trajectory. 4th *row*: SPD trajectory. 1st *column*: Trajectories from different manifolds. 2nd *column*: first-derivative in different manifolds. *3rd column:* The distance in each manifold between the demonstration and the $\mathcal{G}$-DMP reproduction. 4th *column*: The Cartesian representation of the $\mathcal{G}$-DMP reproduction. In 1st and 2nd *columns*, dashed lines represent demonstration data while colored solid lines represent the $\mathcal{G}$-DMP results.

data samples from three Riemannian manifolds (unit quaternion, rotation matrix, and symmetric and positive definite matrix).
- We compared $\mathcal{G}$-DMP with the baseline approaches [9,17].
- Learning manipulability ellipsoids and position by learning $\mathcal{R}^2 \times \mathcal{S}^2_{++}$ with $\mathcal{G}$-DMP.
- Goal switching simulation.

• In real experiment:

- Refilling a watering can by learning $\mathcal{R}^3 \times \mathcal{S}^3 \times \mathcal{S}^3_{++}$ with $\mathcal{G}$-DMP.
- Picking from different boxes task by learning $\mathcal{R}^3 \times \mathcal{S}^3_{++}$ with $\mathcal{G}$-DMP.

We have created one by modifying the 2D-LASA and the 2D-Letters datasets. Mainly, we extended both datasets to include $\mathcal{S}^3, \mathcal{SO}(3)$, and $\mathcal{S}^2_{++}$ along with the original $\mathcal{R}^2$. The 2D-LASA handwriting dataset 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. On the other hand, the 2D-Letters handwriting dataset contains 26 letters of 2D Euclidean motions starting from different initial points and ending to different goals. Each motion is demonstrated 10 times. A demonstration has exactly 200 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 tangent space of a given Riemannian manifold. This allows us to use the exponential map to project the motion onto the manifold. In both datasets, demonstrations are in 2D (xy-plane), however, in order to create the 3D tangent space for both $\mathcal{S}^3$ and $\mathcal{SO}(3)$, we added a z-axis to each demonstration as an average of x- and y-axes. As a result, we obtain $\mathcal{S}^3$ and $\mathcal{SO}(3)$ demonstrations for each demonstration from both datasets.

In order to create SPD training data profiles, we followed different strategies and used the 2D-LASA dataset to generate covariance matrix profiles and the 2D-Letters dataset to generate manipulability profiles. More in detail, we first fit a GMM for each class of the 2D-LASA dataset. We then used GMR to retrieve a 2×2 covariance matrix profile. This covariance matrix profile served as SPD training data for $\mathcal{G}$-DMP. Instead, for the 2D-Letters dataset, we placed the base of a 3-DoF 2D-manipulator at $[0,0]^\top$, and determined the manipulability profile of the manipulator while it tracks the Cartesian trajectory of each demonstration. This manipulability profile served as SPD training data for $\mathcal{G}$-DMP.

5.1. Validation using Riemannian LASA dataset

In order to validate the accuracy of the proposed unified DMP formulation, we created 4 tests in 4 different manifolds, $\boldsymbol{\mathcal{P}} \in \mathcal{R}^2$, $\boldsymbol{\mathcal{Q}} \in \mathcal{S}^3$, $\boldsymbol{\mathcal{R}} \in \mathcal{SO}(3)$, and $\boldsymbol{\mathcal{C}} \in \mathcal{S}^2_{++}$. These are illustrated in Fig. 3 where each row corresponds to a particular manifold. The leftmost column of the figure represents the evolution of the elements of the profile over time.[2] Dashed black lines represent the demonstration and colored lines the reproduction of $\mathcal{G}$-DMP. The second column corresponds to the 1st-time-derivative of the profiles in each manifold, while the 3rd column shows the error or the distance between the $\mathcal{G}$-DMP profile and the demonstration profile for each manifold. The last column (rightmost) shows what the profile looks like in Cartesian space. In the case of $\mathcal{S}^3$, we rotate the 3D-frame of the 3D-Cartesian profile of the G-shape, while in $\mathcal{SO}(3)$ we show the frame rotating around $[0,0,0]^\top$. In the case of the $\mathcal{S}^2_{++}$, we illustrated the covariance matrices over the 2D-Cartesian profile of the G-shape. The results shown in this figure demonstrate the accuracy of the proposed $\mathcal{G}$-DMP to reproduce the desired trajectory profiles in different manifolds.

[2] As SPD matrices are symmetric, and for visualization purposes, in this figure we visualize the SPD by plotting the corresponding Mandel representation.

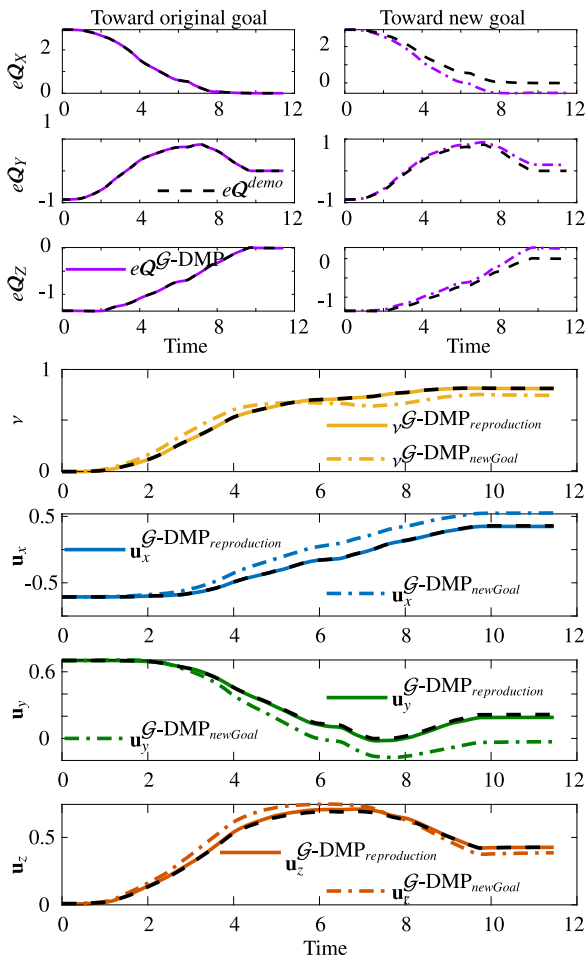


Fig. 4. $\mathcal{G}$-DMP execution of the same unit quaternion trajectory tested in [9]. The first three *rows* show the error between the current unit quaternion and the goal (*left*) and new goal (*right*). The *bottom* four *rows* show the evolution of each unit quaternion element, over time, toward the goal and new goal. Dashed black lines represent information related to the demonstration trajectory.

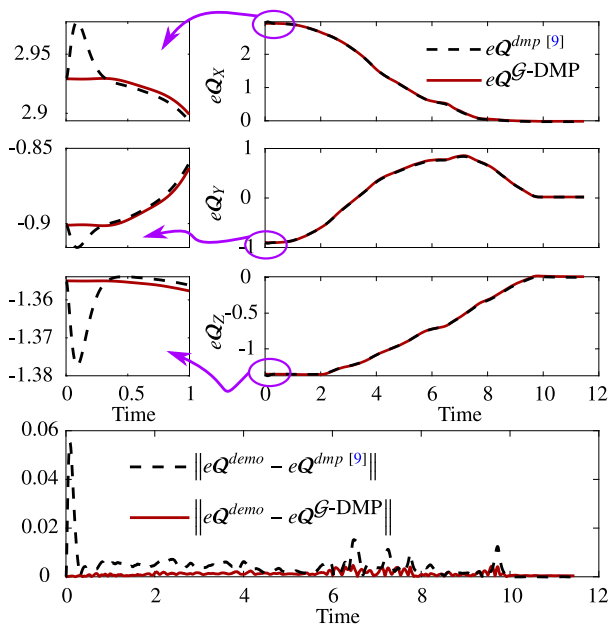


Fig. 5. Comparison between the proposed $\mathcal{G}$-DMP and [9]. The first three *rows* show more stable starting using $\mathcal{G}$-DMP. *Bottom:* Compares the mean error of $\mathcal{G}$-DMP (in red) and [9] (dashed black lines).

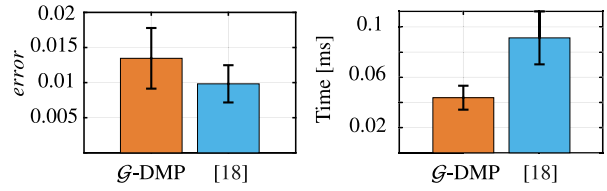


Fig. 6. Comparison between the proposed $\mathcal{G}$-DMP and our previous approach using parallel transport [17]. Both approaches executed 20 $\mathcal{S}^2_{++}$ trajectories of the modified Riemannian LASA dataset. *Left*: The error distance between the demonstration and the reproduction. *Right*: The computational cost in milliseconds per control cycle.

5.2. Comparison with [9]

The proposed $\mathcal{G}$-DMP is rigorously derived in Section 4.1 starting from a generic second-order dynamics evolving on a manifold. Therefore, our formulation is mathematically correct and it does not exhibit the oscillatory behaviors described in [9]. In addition to the mathematical derivation, we provide in this simulation an experimental comparison to support our claim.

More in detail, we compared our $\mathcal{G}$-DMP against the quaternion-based DMP proposed in[3] [9]. We used the same simulated unit quaternion trajectory, where the initial and final quaternions are $\boldsymbol{Q}_0 = [-0.0092 \;\; -0.7126 \;\; 0.7015 \;\; 0.0090]^\top$ and $\boldsymbol{Q}_g = [0.8104 \;\; 0.3364 \;\; 0.2141 \;\; 0.4293]^\top$. Moreover, we used the same DMP parameters, *e.g.*, $\alpha_z = 60$, $N = 60$, and $\alpha_x = 4.6052$. Top-left column of Fig. 4 shows the evolution of the quaternion error computed between the current (from $\mathcal{G}$-DMP) and goal quaternions through $e_{\mathcal{Q}} = 2\text{Log}_{\mathcal{Q}}(\boldsymbol{Q}_g)$. The top-right column shows the evolution of the error toward a new goal $\boldsymbol{Q}_g^{new} = [0.7442 \; 0.5414 \; -0.0343 \; 0.3897]^\top$. The bottom 4 plots, show the evolution of the trajectories of unit quaternion elements toward the original goal and the new one. This figure shows the accuracy of the proposed $\mathcal{G}$-DMP to encode and execute a challenging unit quaternion trajectory. Moreover, it is clear that $\mathcal{G}$-DMP successfully performs a goal-switching task.

Fig. 5 compares the accuracy of our $\mathcal{G}$-DMP with the approach proposed in [9]. The *bottom* plot shows that the proposed $\mathcal{G}$-DMP is more accurate.

Furthermore, the computational complexity during execution, particularly in terms of step time, remains compatible with control frequencies. Specifically, the means of the computational cost exhibited by [9] and $\mathcal{G}$-DMP at each control cycle are 0.04 ms and 0.1 ms, respectively. We also consider a baseline approach that uses the classical DMP and performs an extra normalization of the output. For the baseline, the mean computational cost for integrating and normalizing the output to reproduce a unit quaternion is 0.008 ms per time step. This indicates that all considered approaches can comfortably operate at frequencies exceeding 1 kHz, ensuring real-time responsiveness in robotic control applications.

5.3. Comparison with [17]

To illustrate the difference between our new formulation in (17)–(18) and our previous formulation described in [17], where parallel transport was employed, we have conducted an experiment where both approaches executed 20 $\mathcal{S}^2_{++}$ trajectories of the modified Riemannian LASA dataset (Section 5). Fig. 6 shows bar plots for computational time required for both approaches to learn and execute complete

[3] We thank Leonidas Koutras for sharing with us the implementation and test trajectory of their work in [9].

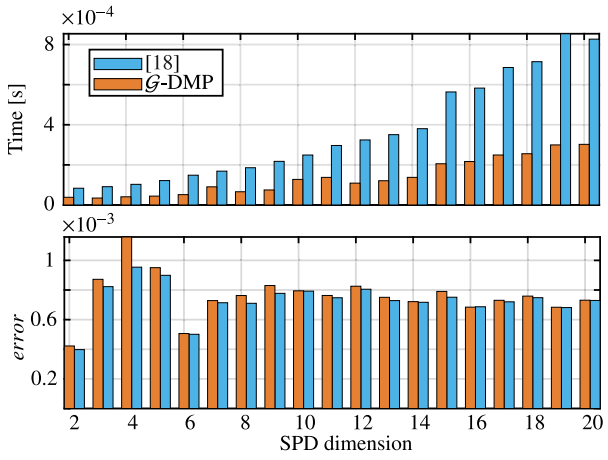


Fig. 7. Comparison between the proposed $\mathcal{G}$-DMP and our previous approach using parallel transport [17]. Both approaches executed 19 $\mathcal{S}^m_{++}$ trajectories, where $m = 2, \dots, 20$. *Top*: The computational cost in milliseconds per control cycle. *Bottom*: The error distance between the demonstration and the reproduction.

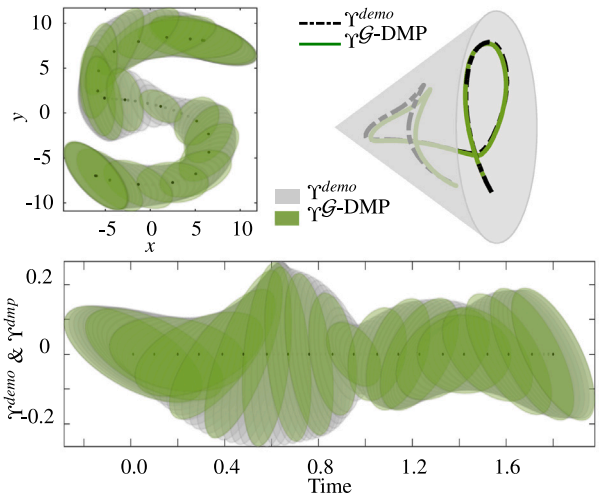


Fig. 8. *Top-Left*: The Cartesian trajectory (in centimeters) executed by the 5-DoF manipulator (black dots), the demonstrated manipulability profile (gray ellipses), and the manipulability profile learned by $\mathcal{G}$-DMP (green ellipses), shown at different times during the execution of the task. *Top-Right*: Representation of SPD manifold (gray cone) containing the demonstrated (dashed black line) and learned (green solid line) manipulability profiles. *Bottom*: Variation of demonstrated (gray ellipses) and learned (green ellipses) manipulability profiles over time.

trajectories, and the log-Euclidean distance [43] between the generated SPD profiles and the ground truth demonstrations.

Results in Fig. 6 show that employing parallel transport provides slightly more accurate results, as evidenced by the reduced log-Euclidean distance from the ground truth demonstrations. However, this improvement comes at a significant computational cost, as indicated by the increased computational time required for this approach. For instance, the mean of the computational cost exhibited by [17] and $\mathcal{G}$-DMP at each control cycle are 0.09 ms and 0.04 ms, respectively.

In Fig. 7 we observe how this computational cost increases exponentially with the approach in [17] as problem dimensions increase. Though [17] exhibits a slight improvement in accuracy, this must be weighed against its heightened computational demands. In this example, we executed both approaches, in [17] and $\mathcal{G}$-DMP, over 19 SPD trajectories with dimensions ranging from $\mathcal{S}^2_{++}$ to $\mathcal{S}^{20}_{++}$, providing a comprehensive comparison.

This trade-off between accuracy and computational efficiency is an important consideration in the selection of the appropriate formulation for specific applications. For tasks where computational resources are abundant and accuracy is paramount, the parallel transport approach may be preferred. However, the new formulation offers a more efficient alternative without penalizing the accuracy for real-time applications or scenarios with limited computational resources. Finally, it is important to note that, while the approach in [17] is specifically designed for SPD matrices, our $\mathcal{G}$-DMP framework is applicable to any Riemannian manifold.

5.4. Learning manipulability ellipsoids

The manipulability of a robotic arm provides an analytical way to evaluate the manipulator's ability to change its end-effector pose from a certain joint configuration. Manipulability can be illustrated as an ellipsoid in 2- or 3-D Euclidean space. Mathematically, the manipulability of a robotic arm is computed from the forward kinematics

$$\dot{\boldsymbol{\mathcal{P}}} = \mathbb{J}\dot{\boldsymbol{\mathcal{J}}}, \tag{35}$$

that relates task velocity $\dot{\boldsymbol{\mathcal{P}}} \in \mathcal{R}^m$ and the joint velocity $\dot{\boldsymbol{\mathcal{J}}} \in \mathcal{R}^n$ through the Jacobian matrix $\mathbb{J} \in \mathcal{R}^{m\times n}$. By considering, in (35), only the joint velocity with unit norm, *i.e.*, $\|\dot{\boldsymbol{\mathcal{J}}}\| = \dot{\boldsymbol{\mathcal{J}}}^\top\dot{\boldsymbol{\mathcal{J}}} = 1$, we obtain

$$\dot{\boldsymbol{\mathcal{J}}}^\top\dot{\boldsymbol{\mathcal{J}}} = \dot{\boldsymbol{\mathcal{P}}}^\top(\mathbb{J}^\dagger)^\top\mathbb{J}^+\dot{\boldsymbol{\mathcal{P}}} = \dot{\boldsymbol{\mathcal{P}}}^\top\left(\mathbb{J}\mathbb{J}^\top\right)^\dagger\dot{\boldsymbol{\mathcal{P}}}, \tag{36}$$

which defines a point on the surface of an ellipsoid in the end-effector velocity space. The SPD matrix $\boldsymbol{\mathcal{Y}} = \left(\mathbb{J}\mathbb{J}^\top\right)^\dagger \in \mathcal{S}^m_{++}$, called manipulability ellipsoid, gives an intuition of the directions where the manipulator can move its end-effector at large/small velocities.

Here we propose to use a toy example similar to the one in [44] to evaluate our $\mathcal{G}$-DMP formulation while operating SPD data profiles. One demonstration $\boldsymbol{\Xi} = \{t_l, \boldsymbol{\mathcal{Y}}_l\}_{l=1}^T$ is obtained by performing a tracking task with a 3-DoF manipulator. Let us call $\boldsymbol{\mathcal{P}}$ the Cartesian position trajectory of the robot end-effector. The desired position trajectory $\hat{\boldsymbol{\mathcal{P}}}$ is then tracked by a 5-DoF robot. The force $\boldsymbol{\mathcal{F}}$ needed to perform the tracking task is computed using the following control law originally proposed in [44]

$$\boldsymbol{\tau}_d = \mathbb{J}^\top\boldsymbol{\mathcal{F}} - \left(\boldsymbol{I} - \mathbb{J}^\top\bar{\mathbb{J}}^\top\right)\alpha\nabla g_t(\boldsymbol{\mathcal{J}}); \quad \alpha > 0, \tag{37}$$

where $\bar{\mathbb{J}}$ is the inertia-weighted pseudo-inverse of $\mathbb{J}$ and $\boldsymbol{\tau}_d$ is the desired joint torque. The cost function $g_t(\boldsymbol{\mathcal{J}})$ is defined as

$$g_t(\boldsymbol{\mathcal{J}}) = \log\left(\det\left(\frac{\hat{\boldsymbol{\mathcal{Y}}}_t + \boldsymbol{\mathcal{Y}}_{a,t}(\boldsymbol{\mathcal{J}})}{2}\right)\right) - \frac{1}{2}\log\left(\det\left(\hat{\boldsymbol{\mathcal{Y}}}_t\boldsymbol{\mathcal{Y}}_{a,t}(\boldsymbol{\mathcal{J}})\right)\right), \tag{38}$$

where $\boldsymbol{\mathcal{Y}}_{a,t}(\boldsymbol{\mathcal{J}})$ are the actual and $\hat{\boldsymbol{\mathcal{Y}}}_t$ the desired manipulability ellipsoids, respectively. $\hat{\boldsymbol{\mathcal{Y}}}_t$ are generated using the proposed $\mathcal{G}$-DMP.

The results of this procedure, applied to track a 2-D S-shape Cartesian trajectory, are shown in Fig. 8. Fig. 8(*top-left*) shows that the desired manipulability profile (green ellipses) smoothly and accurately follows the demonstrated manipulability profile (gray ellipses) while the 5-DoF robot was performing the tracking task. Similar results are shown in Fig. 8(*bottom*), but considering the time evolution of desired and demonstrated manipulability ellipsoids. Fig. 8(*top-right*) depicts the SPD manifold (a cone) and the geodesic curve of the desired and demonstrated manipulability profiles. The $\mathcal{G}$-DMP successfully and accurately followed the demonstrated Cartesian trajectory along with the manipulability profile, in its composite Riemannian form $\mathcal{R}^2 \times \mathcal{S}^2_{++}$, and converged to the goal.

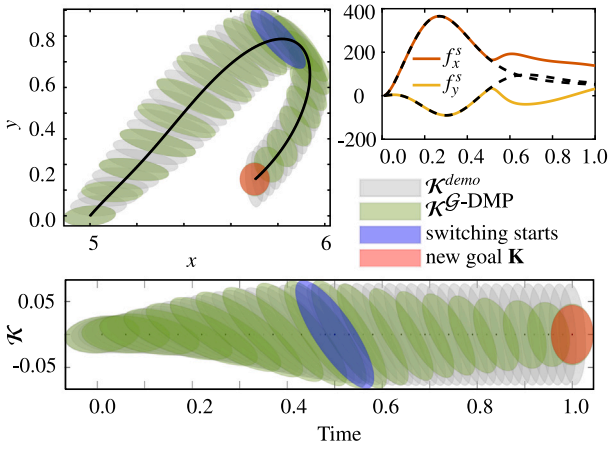


Fig. 9. $\mathcal{G}$-DMP adapts the stiffness profile to a new goal using the mechanism of goal switching (28). Gray ellipsoids represent the demonstrated stiffness profile, green ones are the result of $\mathcal{G}$-DMP, the blue one indicates the instant where goal switching occurred, and the red one denotes the new goal ellipsoid. *Top-Left*: The evolution of $\mathcal{G}$-DMP over a Cartesian trajectory. *Bottom*: The evolution of $\mathcal{G}$-DMP over time. *Top-Right*: The evolution of the spring forces while tracking the Cartesian trajectory.

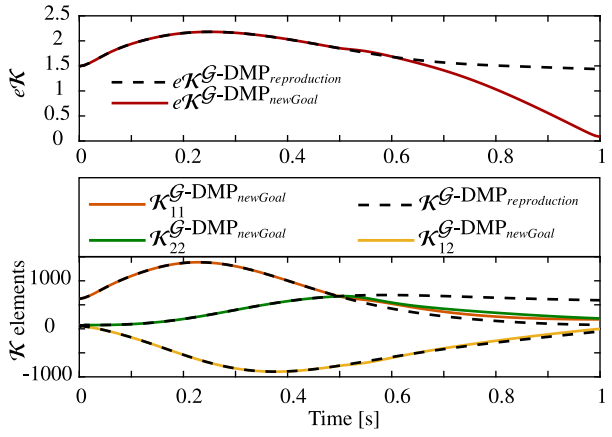


Fig. 10. *Top*: The Log-Euclidean distance between $\mathcal{G}$-DMP evolution and the goal in both cases; reproduction (dashed black lines), adaptation using goal switching (red solid line). *Bottom*: The element of stiffness profile in reproduction (dashed black lines) and adaptation using goal switching (colored solid lines).

5.5. Goal switching

In order to evaluate the proposed $\mathcal{G}$-DMP formulation characteristics under goal switching, we used it to drive an virtual-Mass Spring-Damper (MSD), with a designed variable stiffness profile, along a specific Cartesian trajectory. The variable stiffness profile is designed, such that, it starts with, horizontally-aligned stiffness ellipsoid, [622.9934 39.9577; 39.9577 79.5444], then we rotated it gradually 90°, through $\mathbf{R}^\top \boldsymbol{\mathcal{K}}\mathbf{R}$ ($\mathbf{R}$ is a rotation matrix), until it ends up with, vertically-aligned stiffness ellipsoid, [79.5444 −39.9577; −39.9577 588.2443]. This stiffness profile $\boldsymbol{\mathcal{K}} \in \mathcal{S}^2_{++}$ is our demonstration, the gray ellipsoids in Fig. 9(*top-left*), along with the Cartesian trajectory $\boldsymbol{\mathcal{P}} \in \mathcal{R}^2$, solid black curve. In this simulation, $\mathcal{G}$-DMP encodes the composite Riemannian manifolds $\mathcal{R}^2 \times \mathcal{S}^2_{++}$.

During the execution, we estimated the spring forces $\boldsymbol{f}^s$ while tracking the Cartesian trajectory. The $\mathcal{G}$-DMP reproduction, in the first execution, has been successfully converged to the original goal, dashed lines in Fig. 10(*bottom*). In the second execution, we switched to a new stiffness goal [200 0; 0 200], red ellipsoid in Fig. 9, at the middle of the execution. From Fig. 10(*top*), we can see the error between $\mathcal{G}$-DMP stiffness result, at each time step, and the new stiffness goal converges to zero (the solid red line), which indicates that the $\mathcal{G}$-DMP converges accurately to the new stiffness goal.

5.6. Robot experiments

We evaluated the proposed approach on a 7 DoF Franka Emika Panda robot with two experiments, namely picking from different boxes and refilling a watering can. In order to perform these tasks, the robot had to continuously modulate its position, orientation, stiffness, and/or manipulability. In real settings, orientation trajectories are often collected from demonstrations with a real robot. This requires a preprocessing step to extract unit quaternions from a trajectory of rotation matrices. The step is needed because the robot's forward kinematics is typically expressed as a homogeneous transformation matrix [45]. Numerical approaches to continuously compute quaternions from rotation matrices may return a quaternion at time t and its antipodal at $t+1$, since antipodal quaternions represent the same rotation. The resulting discontinuity can be avoided by checking that the dot product $\mathbf{q}_t \cdot \mathbf{q}_{t+1} > 0$ and replacing $\mathbf{q}_{t+1}$ with $-\mathbf{q}_{t+1}$ otherwise.

5.6.1. Refilling a watering can

In this experiment, the robot had to refill a watering can by immersing it in a tray full of water (see Fig. 11). To perform the task, the robot was controlled using the Cartesian impedance control law

$$\begin{aligned} \boldsymbol{\mathcal{F}}_p &= \boldsymbol{\mathcal{K}}_p \left(\boldsymbol{\mathcal{P}}^{dmp} - \boldsymbol{\mathcal{P}}\right) + \boldsymbol{\mathcal{D}}_p \left(\dot{\boldsymbol{\mathcal{P}}}^{dmp} - \dot{\boldsymbol{\mathcal{P}}}\right), \\ \boldsymbol{\mathcal{F}}_o &= \boldsymbol{\mathcal{K}}_o \operatorname{Log}_{\mathcal{Q}} \left(\mathcal{Q}^{dmp}\right) + \boldsymbol{\mathcal{D}}_o \left(\mathcal{W}^{dmp} - \mathcal{W}\right), \end{aligned} \tag{39}$$

where the subscript p indicates position and o orientation. The measured end-effector position and orientation (unit quaternion) are indicated by $\boldsymbol{\mathcal{P}}$ and $\mathcal{Q}$ respectively, and the corresponding linear and angular velocities are $\dot{\boldsymbol{\mathcal{P}}}$ and $\mathcal{W}$. The desired trajectories $\boldsymbol{\mathcal{P}}^{dmp}$ and $\mathcal{Q}^{dmp}$, as well as the variable stiffness matrix $\boldsymbol{\mathcal{K}}_p$ and the desired velocities $\left(\dot{\boldsymbol{\mathcal{P}}}^{dmp}\right.$ and $\left.\mathcal{W}^{dmp}\right)$, were generated with the proposed $\mathcal{G}$-DMP. The orientation stiffness was kept constant at $\boldsymbol{\mathcal{K}}_o = 150\,\boldsymbol{I}$ Nm/rad. The damping matrices $\boldsymbol{\mathcal{D}}_p$ and $\boldsymbol{\mathcal{D}}_o$ were computed from the respective stiffness matrices using the double diagonalization approach [46]. The robot was controlled at 1 KHz using the joint torques

$$\tau_d = \mathbb{J}^\top \begin{bmatrix} \boldsymbol{\mathcal{F}}_p \\ \boldsymbol{\mathcal{F}}_o \end{bmatrix}, \tag{40}$$

where $\mathbb{J}^\top$ is the transpose of the manipulator Jacobian and the Cartesian forces $\boldsymbol{\mathcal{F}}_p$ and $\boldsymbol{\mathcal{F}}_o$ are defined as in (39).

Desired position, velocity, and stiffness profiles were learned using the proposed $\mathcal{G}$-DMP. In order to estimate a variable stiffness profile, we collected 5 kinesthetic demonstrations containing end-effector positions, velocities, accelerations, and sensed forces. These data were used through the interaction model proposed in [15] to estimate the variable stiffness profile shown in Fig. 11 (bottom). Positions and unit quaternion trajectories were learned from a single demonstration, obtained by averaging the 5 used to obtain the stiffness profile.

The results in Fig. 11 show that the proposed $\mathcal{G}$-DMP formulation is capable of learning complex trajectories evolving on composite Riemannian manifolds $\mathcal{R}^3 \times \mathcal{S}^3 \times \mathcal{S}^3_{++}$ while fulfilling the underlying geometric constraints, *i.e.,* unit norm in variable orientation and symmetry and positive definiteness in variable stiffness profiles.

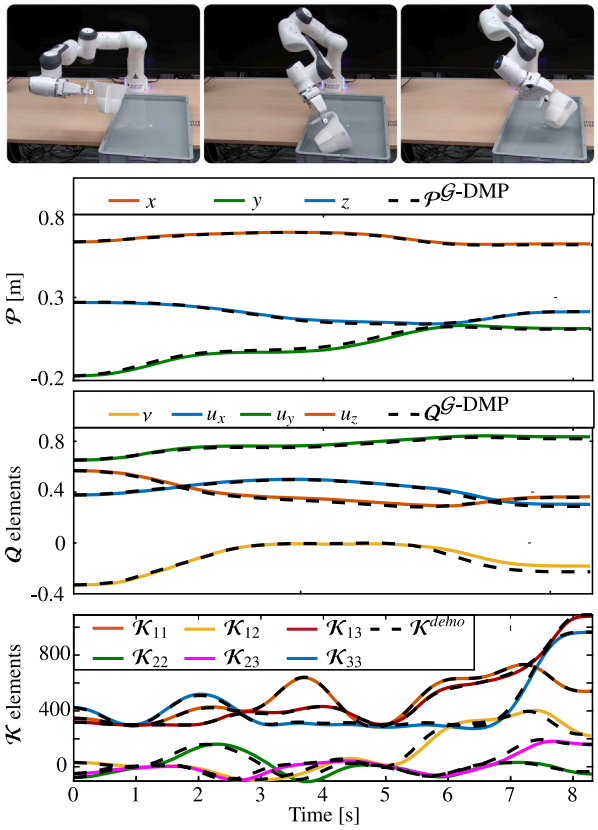


Fig. 11. Results for the refill of a watering can experiment. *Top*: The robot correctly performs the task. *Bottom*: Position, orientation, and stiffness profiles.

5.6.2. Pick from different boxes

In this experiment, the robot had to enter 3 boxes placed at different locations, mimicking a pick from each of the boxes (see Fig. 12). The experiment was designed to show that geometry-aware DMPs can (i) effectively encode manipulability profiles and (ii) change the goal after the learning.

We provided a kinesthetic demonstration to make the robot enter box 1 while collecting end-effector position and joint trajectories. As detailed in Section 5.4, collected trajectories were used to learn position and manipulability profiles using geometry-aware DMPs. At run time, the robot was controlled using the control law (37) to track the DMP position as main task and to exploit its redundant DoF to follow the desired manipulability profile. As shown in Fig. 12 (top), the robot followed accurately both position and manipulability profiles and successfully entered box 1.

In order to experimentally verify the generalization capabilities of geometry-aware DMPs, we repeated the experiment by entering two boxes placed at different locations wrt box 1. To measure the new goal, we manually placed the robot inside the boxes and stored its end-effector position. As shown in Fig. 12 (middle)–(bottom), the robot reached the new position goals inside box 2 and 3. As already mentioned, the manipulability profile was tracked in the null-space of the position task, which introduces an error between the planned and executed manipulability profiles. However, in this task, null-space tracking was sufficient to preserve a joint configuration that let the robot enter boxes 2 and 3 without collision.

Overall, the results in Fig. 12 show that the proposed $\mathcal{G}$-DMP formulation is capable of learning complex trajectories evolving on the composite Riemannian manifold $\mathcal{R}^3 \times \mathcal{S}^3_{++}$ while fulfilling the underlying geometric constraints, *i.e.,* symmetry and positive definiteness in variable manipulability profiles.

6. Conclusion

In this paper, we have exploited Riemannian geometry to derive a new formulation of DMP that is capable of learning and reproducing robot skills evolving on any Riemannian manifold. Our new formulation, Geometry-aware DMP ($\mathcal{G}$-DMP), is manifold independent and allows us to treat data belonging to different manifolds in a unified manner. It also preserves the underlying geometric constraints during both learning and reproduction without pre- or post-processing of the data. Moreover, it preserves the properties of the classical DMP formulation such as convergence to a given target and the possibility to change the target at run-time (goal switching).

$\mathcal{G}$-DMP has been extensively validated through multiple simulation examples and two experiments on a real robotic manipulator. For simulation, we augmented two Euclidean datasets (2D-Letters and LASA handwriting) with data samples from three Riemannian manifolds ($\mathcal{S}^3$, $\mathcal{SO}(3)$, and $\mathcal{S}^2_{++}$). We showed that $\mathcal{G}$-DMP can accurately learn profiles evolving on such manifolds while converging to a (possibly changing) goal. Moreover, a comparison with a baseline approach was conducted on a unit quaternion trajectory. In this case, $\mathcal{G}$-DMP shows improvement by avoiding slight jumps at the beginning of the trajectories. Finally, real experiments show the effectiveness of $\mathcal{G}$-DMP in encoding data from manifolds such as orientation, and SPD matrices.

In the future, we propose to integrate our approach with iterative learning algorithms—for example iterative learning control—in order to adapt to different situations and perform more complex tasks such as physical interaction control. Moreover, extending exploration-based learning methods to Riemannian manifolds is an open research problem. These methods are crucial when a robot needs to significantly adapt its behavior to a new situation by considering the data directly on its corresponding manifold. This will allow us to successfully exploit $\mathcal{G}$-DMPs in a large diversity of task situations.

CRediT authorship contribution statement

Fares J. Abu-Dakka: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Matteo Saveriano:** Writing – review & editing, Validation, Investigation, Funding acquisition, Data curation, Conceptualization. **Ville Kyrki:** Writing – review & editing, Funding acquisition, Conceptualization.

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

I have shared the link to my data/code in the manuscript.

Acknowledgments

This work is supported in part by Basque Government (ELKARTEK), Spain projects Proflow KK-2022/00024 and HELDU KK-2023/00055, in part by the European Union project INVERSE, Italy (GA No. 101136067), and in part by CHIST-ERA project IPALM, Finland (Academy of Finland decision 326304). Real experiments were conducted at the Department of Computer Science, University of Innsbruck, Austria.

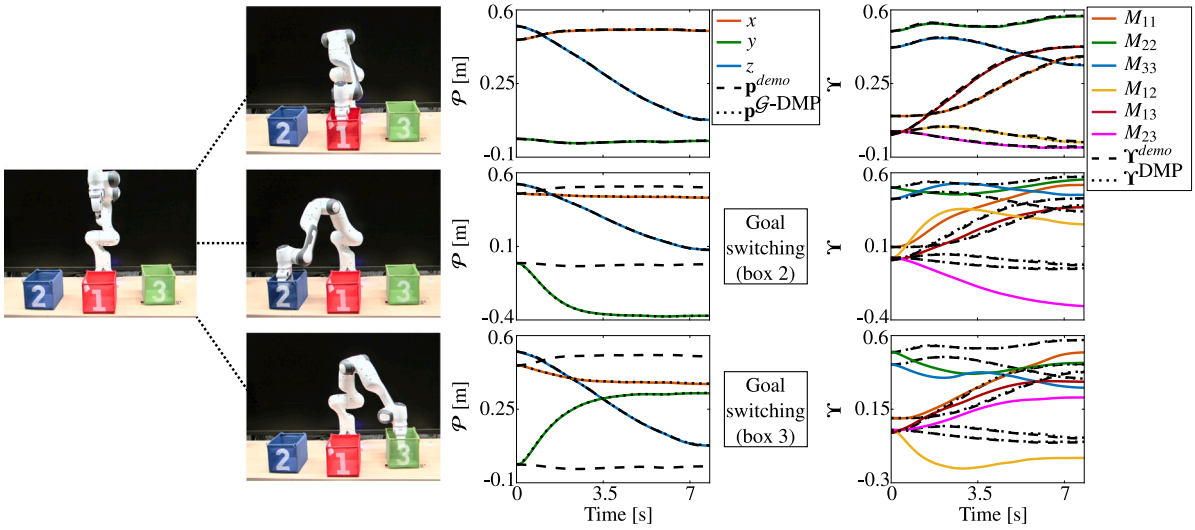


Fig. 12. Results for the pick from different boxes experiment. *Top*: Picking from the demonstrated box 1. *Middle*: Goal switching is used to pick from a new box 2. *Bottom*: Goal switching is used to pick from a new box 3. In the 3 cases, manipulability is controlled in the null-space of the position task to maintain a certain joint configuration during the motion.

Appendix A. Characterization of used manifolds

A.1. The SPD manifold $\mathcal{S}^m_{++}$

As early mentioned, SPD matrices is important in robotics as it encapsulate different types of data. The space $\mathcal{S}^m_{++}$ is defined as the space of $m \times m$ Symmetric Positive Definite matrices. This space is not closed under scalar product and addition [34], thus, we cannot use classical Euclidean arithmetic operators to manipulate these matrices. Alternatively, we can equip SPD matrices with A Riemannian metric in order to form a Riemannian manifold [34].

Note that the space $\mathcal{S}^m_{++}$ can be represented as the interior of a convex cone embedded in its tangent space of symmetric $m \times m$ matrices $\mathcal{SYM}^m$.

For $\boldsymbol{Q}, \boldsymbol{U} \in \mathcal{S}^m_{++}$ and $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{U}}\mathcal{S}^m_{++}$, the logarithmic and exponential maps (8) and (9) can be defined as in [34]

$$\boldsymbol{v} = \mathrm{Log}_{\boldsymbol{U}}(\boldsymbol{Q}) = \boldsymbol{U}^{\frac{1}{2}} \mathrm{logm}\left(\boldsymbol{U}^{-\frac{1}{2}} \boldsymbol{Q} \boldsymbol{U}^{-\frac{1}{2}}\right) \boldsymbol{U}^{\frac{1}{2}}, \tag{A.1}$$

$$\boldsymbol{Q} = \mathrm{Exp}_{\boldsymbol{U}}(\boldsymbol{v}) = \boldsymbol{U}^{\frac{1}{2}} \mathrm{expm}\left(\boldsymbol{U}^{-\frac{1}{2}} \boldsymbol{v} \boldsymbol{U}^{-\frac{1}{2}}\right) \boldsymbol{U}^{\frac{1}{2}}, \tag{A.2}$$

where logm$(\cdot)$ and expm$(\cdot)$ are the matrix logarithm and exponential functions.

A.2. The unit m-sphere manifold $\mathcal{S}^m$

$\mathcal{S}^m$ is a topological space embedded in $\mathcal{R}^{m+1}$ Cartesian space, where $\mathcal{S}^m = \left\{\boldsymbol{X} \in \mathcal{R}^{m+1} : \|\boldsymbol{X}\| = 1\right\}$. For $\boldsymbol{Q}, \boldsymbol{U} \in \mathcal{S}^m$ and $\boldsymbol{v}, \boldsymbol{r} \in \mathcal{T}_{\boldsymbol{U}}\mathcal{S}^m$ then, the logarithmic and exponential maps (8) and (9) are defined as in [47]

$$\boldsymbol{v} = \mathrm{Log}_{\boldsymbol{U}}(\boldsymbol{Q}) = \frac{\boldsymbol{Q} - (\boldsymbol{U}^\top \boldsymbol{Q})\boldsymbol{U}}{\left\|\boldsymbol{Q} - (\boldsymbol{U}^\top \boldsymbol{Q})\boldsymbol{U}\right\|} d(\boldsymbol{U}, \boldsymbol{Q}), \tag{A.3}$$

$$\boldsymbol{Q} = \mathrm{Exp}_{\boldsymbol{U}}(\boldsymbol{v}) = \boldsymbol{U}\cos(\|\boldsymbol{v}\|) + \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}\sin(\|\boldsymbol{v}\|), \tag{A.4}$$

where $d(\boldsymbol{U}, \boldsymbol{Q}) \equiv \arccos(\boldsymbol{Q}^\top \boldsymbol{U})$ defines the geodesic distance between $\boldsymbol{Q}$ and $\boldsymbol{U}$.

A.3. The unit quaternions group $\mathcal{S}^3$

One way to describe the robot's end-effector orientation, in 3D-space, is to use unit quaternion representation. For $\boldsymbol{Q}, \boldsymbol{U} \in \mathcal{S}^3$ and $\boldsymbol{v}, \boldsymbol{r} \in \mathcal{T}_{\boldsymbol{U}}\mathcal{S}^3 \equiv \mathcal{R}^3$, where $\mathcal{S}^3$ is a unit sphere in $\mathcal{R}^4$, $\boldsymbol{Q} = v_q + \boldsymbol{u}_q$, $v_q \in \mathcal{R}$, and $\boldsymbol{u}_q \in \mathcal{R}^3$. The logarithmic and exponential maps (8) and (9) are

$$\begin{aligned} \boldsymbol{v} &= \mathrm{Log}_{\boldsymbol{U}}(\boldsymbol{Q}) = \mathrm{Log}\left(\boldsymbol{Q} * \bar{\boldsymbol{U}}\right) \\ &= \begin{cases} \arccos(v)\frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}, & \boldsymbol{u} \neq \boldsymbol{0} \\ [0 \;\; 0 \;\; 0]^\top, & \text{otherwise.} \end{cases} \end{aligned} \tag{A.5}$$

$$\begin{aligned} \boldsymbol{Q} &= \mathrm{Exp}_{\boldsymbol{U}}(\boldsymbol{v}) \\ &= \begin{cases} \left[\cos(\|\boldsymbol{v}\|) + \sin(\|\boldsymbol{v}\|)\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}\right] * \boldsymbol{U}, & \boldsymbol{v} \neq \boldsymbol{0} \\ \left[1 + [0 \;\; 0 \;\; 0]^\top\right] * \boldsymbol{U}, & \text{otherwise.} \end{cases} \end{aligned} \tag{A.6}$$

where $\boldsymbol{Q} * \bar{\boldsymbol{U}} = v + \boldsymbol{u} \in \mathcal{S}^3$, and $\boldsymbol{v} \in \mathcal{R}^3$ is treated as a quaternion with $v = 0$.

A.4. The special orthogonal group $\mathcal{SO}(m)$

$\mathcal{SO}(m)$ is a subgroup of the orthogonal group $\mathcal{O}(m)$ where its determinant is 1. Let us define $\boldsymbol{R}_1, \boldsymbol{R}_2 \in \mathcal{SO}(m)$ and $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{R}_1}\mathcal{SO}(m)$, then the logarithmic and exponential maps (8) and (9) are defined as in [47]

$$\boldsymbol{v} = \mathrm{Log}_{\boldsymbol{R}_1}(\boldsymbol{R}_2) = \mathrm{logm}\left({\boldsymbol{R}_1}^\top \boldsymbol{R}_2\right), \tag{A.7}$$

$$\boldsymbol{R}_2 = \mathrm{Exp}_{\boldsymbol{R}_1}(\boldsymbol{v}) = \mathrm{expm}(\boldsymbol{v})\,\boldsymbol{R}_1. \tag{A.8}$$

A.5. The rotation group $\mathcal{SO}(3)$

Traditionally, orientations, in 3D-space, were represented through rotation matrices in $\mathcal{SO}(3) = \{\boldsymbol{R} \in \mathcal{R}^{3\times3} : |\boldsymbol{R}| = 1, \boldsymbol{R}^\top\boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^\top = \boldsymbol{I}\}$ which are widely used in robotics. Let us define $\boldsymbol{R}_1, \boldsymbol{R}_2 \in \mathcal{SO}(3)$ and $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{R}_1}\mathcal{SO}(3)$, then (8) will be [48]

$$\begin{aligned} \boldsymbol{v} &= \mathrm{Log}_{\boldsymbol{R}_1}\left(\boldsymbol{R}_2\right) = \mathrm{Log}\left(\boldsymbol{R}_2{\boldsymbol{R}_1}^\top\right) = \mathrm{Log}(\boldsymbol{R}) \\ &= \begin{cases} [0,0,0]^\top, & \boldsymbol{R} = \boldsymbol{I} \\ \omega = \theta\boldsymbol{n}, & \text{otherwise,} \end{cases} \end{aligned} \tag{A.9}$$

where

$$\theta = \arccos\left(\frac{\text{trace}(\boldsymbol{R})-1}{2}\right),\ \boldsymbol{n} = \frac{1}{2\sin(\theta)}\begin{bmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12}\end{bmatrix}$$

and (9) will be

$$\begin{aligned}\boldsymbol{R}_2 &= \text{Exp}_{\boldsymbol{R}_1}\left([\boldsymbol{v}]_\times\right) \\ &= \left(\boldsymbol{I} + \sin(\theta)\frac{[\boldsymbol{v}]_\times}{\|\boldsymbol{v}\|} + (1-\cos(\theta))\frac{[\boldsymbol{v}]_\times^2}{\|\boldsymbol{v}\|^2}\right)\boldsymbol{R}_1,\end{aligned} \tag{A.10}$$

Note that the mappings in (A.6)–(A.7) and in (A.9)–(A.10) are computed using Lie group theory as unit quaternions and rotation matrices form a Lie group [42]. In particular, the mappings are based on the tangent space placed at the identity element (the so-called Lie algebra), and the product operations are used to parallel transport vectors from the Lie algebra to the tangent space placed at a different point ($\boldsymbol{U}$ or $\boldsymbol{R}_1$). We used the term Riemannian through the paper since every Lie group equipped with a Riemannian metric is a Riemannian manifold, but not vice versa.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.neucom.2024.128056.

References

[1] S. Schaal, Is imitation learning the route to humanoid robots? Trends in Cognitive Sciences 3 (6) (1999) 233–242.

[2] A. Billard, S. Calinon, R. Dillmann, Learning from humans, in: Springer Handbook of Robotics, 2016, pp. 1995–2014.

[3] H. Ravichandar, A.S. Polydoros, S. Chernova, A. Billard, Recent advances in robot learning from demonstration, Annual Rev. Control Robot. Autonomous Syst. 3 (1) (2020) 297–330.

[4] 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.

[5] S.M. Khansari-Zadeh, A. Billard, Learning stable non-linear dynamical systems with Gaussian mixture models, IEEE Trans. Robot. 27 (5) (2011) 943–957.

[6] 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.

[7] A.J. Ijspeert, J. Nakanishi, S. Schaal, Learning rhythmic movements by demonstration using nonlinear oscillators, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, 2002, pp. 958–963.

[8] A. Ude, B. Nemec, T. Petrić, J. Morimoto, Orientation in cartesian space dynamic movement primitives, in: IEEE International Conference on Robotics and Automation, Hong Kong, China, 2014, pp. 2997–3004.

[9] L. Koutras, Z. Doulgeri, A correct formulation for the orientation dynamic movement primitives for robot control in the cartesian space, in: Conference on Robot Learning, Osaka, Japan, 2020, pp. 293–302.

[10] Y. Huang, F.J. Abu-Dakka, J. Silvério, D.G. Caldwell, Toward orientation learning and adaptation in cartesian space, IEEE Trans. Robot. 37 (1) (2020) 82–98.

[11] S. Traversaro, S. Brossette, A. Escande, F. Nori, Identification of fully physical consistent inertial parameters using optimization on manifolds, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Daejeon, Korea, 2016, pp. 5446–5451.

[12] T. Yoshikawa, Manipulability of robotic mechanisms, Int. J. Robot. Res. 4 (2) (1985) 3–9.

[13] F.J. Abu-Dakka, Y. Huang, J. Silvério, V. Kyrki, A probabilistic framework for learning geometry-based robot manipulation skills, Robot. Auton. Syst. 141 (2021) 103761.

[14] R. Ikeura, H. Inooka, Variable impedance control of a robot for cooperation with a human, in: IEEE International Conference on Robotics and Automation, Nagoya, Japan, 1995, pp. 3097–3102.

[15] F.J. Abu-Dakka, L. Rozo, D.G. Caldwell, Force-based variable impedance learning for robotic manipulation, Robot. Auton. Syst. 109 (2018) 156–167.

[16] 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.

[17] F.J. Abu-Dakka, V. Kyrki, Geometry-aware dynamic movement primitives, in: IEEE International Conference on Robotics and Automation, Paris, France, 2020, pp. 4421–4426.

[18] M. Saveriano, F. Franzel, D. Lee, Merging position and orientation motion primitives, in: IEEE International Conference on Robotics and Automation, Montreal, QC, Canada, 2019, pp. 7041–7047.

[19] M. Saveriano, F.J. Abu-Dakka, A. Kramberger, L. Peternel, Dynamic movement primitives in robotics: A tutorial survey, Int. J. Robot. Res. 42 (13) (2023) 1133–1184.

[20] A. Paraschos, C. Daniel, J.R. Peters, G. Neumann, Probabilistic movement primitives, in: Advances in Neural Information Processing Systems, Nevada, USA, 2013, pp. 2616–2624.

[21] S. Calinon, D. Bruno, D.G. Caldwell, A task-parameterized probabilistic model with minimal intervention control, in: IEEE International Conference on Robotics and Automation, Hong Kong, China, 2014, pp. 3339–3344.

[22] P. Pastor, H. Hoffmann, T. Asfour, S. Schaal, Learning and generalization of motor skills by learning from demonstration, in: IEEE International Conference on Robotics and Automation, Kobe, Japan, 2009, pp. 763–768.

[23] F.J. Abu-Dakka, M. Saveriano, L. Peternel, Periodic DMP formulation for quaternion trajectories, in: IEEE International Conference of Advanced Robotics, Ljubljana, Slovenia, 2021, pp. 658–663.

[24] P.-Y. Gousenbourger, E. Massart, P.-A. Absil, Data fitting on manifolds with composite bézier-like curves and blended cubic splines, J. Math. Imaging Vision 61 (5) (2019) 645–671.

[25] S. Kim, R. Haschke, H. Ritter, Gaussian mixture model for 3-dof orientations, Robot. Auton. Syst. 87 (2017) 28–37.

[26] 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.

[27] L. Dodero, H.Q. Minh, M. San Biagio, V. Murino, D. Sona, Kernel-based classification for brain connectivity graphs on the Riemannian manifold of positive definite matrices, in: IEEE International Symposium on Biomedical Imaging, Brooklyn, NY, USA, 2015, pp. 42–45.

[28] S. Herath, M. Harandi, F. Porikli, Learning an invariant hilbert space for domain adaptation, in: IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, Hawaii, 2017, pp. 3845–3854.

[29] D.C. Alexander, C. Pierpaoli, P.J. Basser, J.C. Gee, Spatial transformations of diffusion tensor magnetic resonance images, IEEE Trans. Med. Imaging 20 (11) (2001) 1131–1139.

[30] S. Calinon, Gaussians on Riemannian manifolds: Applications for robot learning and adaptive control, IEEE Robot. Autom. Mag. 27 (2) (2020) 33–45.

[31] N. Jaquier, S. Calinon, Gaussian mixture regression on symmetric positive definite matrices manifolds: Application to wrist motion estimation with sEMG, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Vancouver, Canada, 2017, pp. 59–64.

[32] X. Pennec, Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements, J. Math. Imaging Vision 25 (1) (2006) 127–154.

[33] P.-A. Absil, R. Mahony, R. Sepulchre, Optimization algorithms on matrix manifolds, Princeton University Press, 2009.

[34] X. Pennec, P. Fillard, N. Ayache, A Riemannian framework for tensor computing, Int. J. Comput. Vis. 66 (1) (2006) 41–66.

[35] M. Fréchet, Les éléments aléatoires de nature quelconque dans un espace distancié, in: Ann. Inst. H. Poincaré, 10, (4) 1948, pp. 215–310.

[36] S. Fiori, I. Cervigni, M. Ippoliti, C. Menotta, Synthetic nonlinear second-order oscillators on Riemannian manifolds and their numerical simulation, Discrete Contin. Dyn. Syst. Ser. B 27 (3) (2022) 1227–1262.

[37] N. Boumal, P.-A. Absil, A discrete regression method on manifolds and its application to data on SO(n), IFAC Proc. Vol. 44 (1) (2011) 2284–2289, 18th IFAC World Congress.

[38] L. Markus, Asymptotically autonomous differential systems, in: S. Lefschetz (Ed.), Contributions To the Theory of Nonlinear Oscillations III, Princeton University Press, 1956, pp. 17–30.

[39] S. Fiori, Manifold calculus in system theory and control–fundamentals and first-order systems, Symmetry 13 (11) (2021).

[40] J. Slotine, W. Li, Applied nonlinear control, Prentice-Hall Englewood Cliffs, 1991.

[41] 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, Atlanta, GA, USA, 2010, pp. 6277–6280.

[42] J. Sola, J. Deray, D. Atchuthan, A micro Lie theory for state estimation in robotics, 2018, arXiv preprint arXiv:1812.01537.

[43] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, M. Harandi, Kernel methods on Riemannian manifolds with Gaussian RBF kernels, IEEE Trans. Pattern Anal. Mach. Intell. 37 (12) (2015) 2464–2477.

[44] L. Rozo, N. Jaquier, S. Calinon, D.G. Caldwell, Learning manipulability ellipsoids for task compatibility in robot manipulation, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Vancouver, Canada, 2017, pp. 3183–3189.

[45] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics: Modelling, Planning and Control, Springer, 2009.

[46] A. Albu-Schaffer, C. Ott, U. Frese, G. Hirzinger, Cartesian impedance control of redundant robots: Recent results with the DLR-light-weight-arms, in: IEEE International Conference on Robotics and Automation, Taipei, Taiwan, 2003, pp. 3704–3709.
[47] Q. Rentmeesters, A gradient method for geodesic data fitting on some symmetric Riemannian manifolds, in: IEEE Conference on Decision and Control and European Control Conference, Orlando, FL, USA, 2011, pp. 7141–7146.
[48] R.M. Murray, Z. Li, S.S. Sastry, A mathematical introduction to robotic manipulation, CRC Press, 2017.

Fares J. Abu-Dakka received his B.Sc. degree in Mechanical Engineering from Birzeit University, Palestine (2003), and advanced degrees (DEA and Ph.D.) in robotics motion planning from the Polytechnic University of Valencia, Spain (2006, 2011). His postdoctoral journey began at the Jozef Stefan Institute, Slovenia, in 2012. From 2013 to 2016, he was a Visiting Professor at the Carlos III University of Madrid, Spain, followed by a postdoctoral role at the Istituto Italiano di Tecnologia (IIT) from 2016 to 2019. He was a Research Fellow at Aalto University (2019-2022) before joining the Technical University of Munich as a Senior Scientist in 2022, where he led the Robot Learning group at MIRMI. Currently, he is a Lecturer and Researcher at the Faculty of Engineering, Mondragon Unibertsitatea, Spain. His research spans control theory, differential geometry, and machine learning, with a focus on improving robot manipulation performance and safety. He also serves as an Associate Editor for IEEE Robotics and Automation Letters (RA-L) and EEE Transactions on Robotics (T-RO). For more details, visit his https://sites.google.com/view/abudakka/webpage.

Matteo Saveriano received his B.Sc. and M.Sc. degrees in Automatic Control Engineering from the University of Naples, Italy, in 2008 and 2011, respectively. He earned his 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. He has previously served as an Assistant Professor at the University of Innsbruck and as a Postdoctoral Researcher at the German Aerospace Center (DLR). He is an Associate Editor for RA-L and IJRR. His research focuses on robot learning, human–robot interaction, and the understanding and interpretation of human activities. For more details, visit his https://matteosaveriano.weebly.com/webpage.

Ville Kyrki received his M.Sc. and Ph.D. degrees in Computer Science from Lappeenranta University of Technology, Finland, in 1999 and 2002, respectively. He was a Postdoctoral Fellow at the Royal Institute of Technology, Stockholm, Sweden, from 2003 to 2004, before returning to Lappeenranta University of Technology, where he held various positions from 2003 to 2009. From 2009 to 2012, he served as a Professor of Computer Science at Lappeenranta University of Technology. Since 2012, he has been an Associate Professor of Intelligent Mobile Machines at Aalto University, Helsinki, Finland. His primary research interests are in robotic perception, decision-making, and learning. Dr. Kyrki is a Fellow of the Academy of Engineering Sciences (Finland) and a member of the Finnish Robotics Society and the Finnish Society of Automation. He has held various leadership roles within the IEEE Finland Section, including Chair and Vice Chair of the Jt. Chapter of CS, RA, and SMC Societies, Treasurer, and CoChair of the IEEE RAS TC in Computer and Robot Vision. He served as an Associate Editor for the IEEE Transactions on Robotics from 2014 to 2017.