Robotics and Autonomous Systems 141 (2021) 103761

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

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

A probabilistic framework for learning geometry-based robot manipulation skills

Fares J. Abu-Dakka [a,*], Yanlong Huang [b], João Silvério [c], Ville Kyrki [a]

[a] Intelligent Robotics Group, Department of Electrical Engineering and Automation (EEA), Aalto University, Espoo, Finland
[b] School of Computing, University of Leeds, Leeds, UK
[c] Idiap Research Institute, Martigny, Switzerland

ARTICLE INFO

Article history:
Received 7 July 2020
Received in revised form 27 November 2020
Accepted 1 March 2021
Available online 6 March 2021

Keywords:
Learning from demonstration
Variable impedance
Robot learning
Manipulability ellipsoids
Riemannian manifolds

ABSTRACT

Programming robots to perform complex manipulation tasks is difficult because many tasks require sophisticated controllers that may rely on data such as manipulability ellipsoids, stiffness/damping and inertia matrices. Such data are naturally represented as Symmetric Positive Definite (SPD) matrices to capture specific geometric characteristics of the data, which increases the complexity of hard-coding them. To alleviate this difficulty, the Learning from Demonstration (LfD) paradigm can be used in order to learn robot manipulation skills with specific geometric constraints encapsulated in SPD matrices.

Learned skills often need to be adapted when they are applied to new situations. While existing techniques can adapt Cartesian and joint space trajectories described by various desired points, *the adaptation of motion skills encapsulated in SPD matrices remains an open problem*. In this paper, we introduce a new LfD framework that can learn robot manipulation skills encapsulated in SPD matrices from expert demonstrations and adapt them to new situations defined by new start-, via- and end-matrices. The proposed approach leverages Kernelized Movement Primitives (KMPs) to generate SPD-based robot manipulation skills that smoothly adapt the demonstrations to conform to new constraints. We validate the proposed framework using a couple of simulations in addition to a real experiment scenario.

© 2021 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

Robots are entering human environments such as houses, hospitals and museums. Such environments are highly unstructured, dynamic and uncertain, making explicit programming of required robot skills infeasible. The difficulty is particularly apparent in manipulation tasks that require an encapsulation of its characteristics in specific geometry constraints data type. Some of these geometry constraints robotics data types are: (i) Orientation data [1], encapsulated in unit quaternions $\mathcal{S}^3$ or rotation matrices $\mathcal{SO}(3)$; (ii) Manipulability data [2–6], encapsulated in Symmetric Positive Definite (SPD) matrices $\mathcal{S}^m_{++}$; (iii) Impedance data (stiffness and damping matrices) [7–11], encapsulated in SPD matrices $\mathcal{S}^m_{++}$, *etc.*

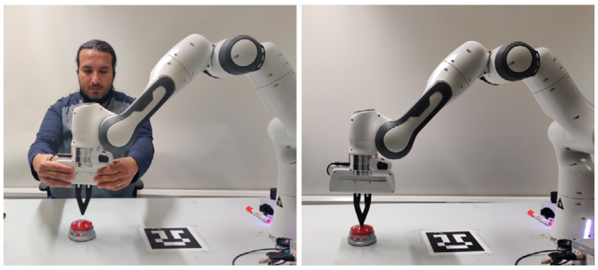

Fig. 1. *Left*: a human operator teaches a robot how to perform a push-button task. *Right*: a snapshot showing the *Franka Emika Panda* robot performing a push-button task. It is not always straightforward to estimate a stiffness profile from demonstrations in a way that all task constraints are accurately fulfilled. For instance, it is not rare that, due to limitations in the underlying skill representation, the robot stiffness becomes too high/low at a specific moment of the task. Our approach addresses this problem by adding new start-, via- and/or end-SPD points that represent desired stiffnesses.

The Learning from Demonstration (LfD) paradigm can be used to learn tasks from human demonstrations (Fig. 1-*left*) intuitively [14]. Several LfD approaches have been developed including Dynamic Movement Primitives (DMPs) [15], Gaussian Mixture

* Corresponding author.
E-mail addresses: fares.abu-dakka@aalto.fi (F.J. Abu-Dakka), y.l.huang@leeds.ac.uk (Y. Huang), joao.silverio@idiap.ch (J. Silvério), ville.kyrki@aalto.fi (V. Kyrki).

https://doi.org/10.1016/j.robot.2021.103761
0921-8890/© 2021 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Table 1
Comparison among the state-of-the-art and our approach.

	Probabilistic	New start-SPD point	New goal-SPD point	Via-SPD point/s
Jaquier et al. [6,12]	✓	–	–	–
Abu-Dakka et al. [10]	✓	–	–	–
Abu-Dakka et al. [13]	–	–	✓	–
Zeestraten et al. [1]	✓	✓[a]	✓[a]	–
Our approach	✓	✓	✓	✓

[a]Note that the proposed formulation in [1] is targeted at data on $\mathcal{S}^3$ and not $\mathcal{S}^m_{++}$. However we understand that the extension to $\mathcal{S}^m_{++}$ would be trivial and thus include it here as a point of comparison.

Model (GMM) [16], and more recently, Kernelized Movement Primitives (KMPs) [17]. In order to deal with orientation data, the idea of transforming data from $\mathcal{S}^3$ into $\mathbb{R}^3$ was used in orientation-DMP [18,19] and orientation-KMP [20]. For the case of learning SPD matrices, one can directly stack elements of each SPD matrix into a vector and subsequently learn motion patterns of the corresponding vectors by using DMPs, GMM or KMPs, whereas this treatment ignores the underlying data structure.

There are some previous works dealt with SPD data by taking into account the insight that demonstrations lie on a Riemannian manifold with associated metrics, e.g. learning manipulability ellipsoids [6] and impedance profiles [10]. These approaches have focused on synthesizing SPD matrices when a robot is presented with similar inputs to those observed during demonstrations. However, the adaptation of SPD-parameterized skills to unseen inputs remains an open problem in the LfD literature, as well as the adaptation to via-SPD data points.

In this paper, we go beyond previous works on learning *SPD-matrix-based robot skills*[1] [6,10,12] by considering both the learning and adaptation of SPD-based skills. The framework is based on first transforming SPD-matrix profiles into Euclidean space either by using Cholesky factorization or a Riemannian metric (Section 4). Subsequently, this Euclidean representation is probabilistically encoded using GMM, which is exploited later to retrieve the conditional distribution of the data using Gaussian Mixture Regression (GMR) (Section 5.1). This distribution is then exploited using a kernelized approach [17], which extends KMP [17] to the case of SPD profiles (Sections 5.2 and 5.3). This allows the model to generate desired SPD matrices for new inputs (so-called start-, via- or end-points).

As an illustrative example, consider a robot needs to perform a motion with learned control gains. When the task conditions change (e.g. the robot needs to go through a narrow opening that was not shown in the demonstrations), the control gains should be modified in some parts of the task (e.g. to be more precise), while keeping the rest unchanged. The proposed model allows smooth adaptation of the gains while ensuring their positive definiteness required for the stability of control (Fig. 1).

To the best of our knowledge, no previous approach for learning SPD-based manipulation skills from demonstrations has successfully tackled the adaptation problem to desired via-SPD-points. We here propose a framework that allows to elegantly tackle the aforementioned problems through the inclusion of *SPD-via-points*.

In summary, the main contribution of this paper is a probabilistic framework for learning and adapting SPD-based robot manipulation skills that allows

– learning different types of manipulation skills that rely on SPD matrices for correct execution, and
– adapting learned skills to new unseen inputs.
– adapting learned skills to new start-, via- and end-points represented as SPD matrices, which are unseen during demonstrations.

We also present a comparison between two different methods of pre-/post-processing the original SPD data: a Riemannian metric and the typically used Cholesky factorization [10,21–23], for the problem at hand.

We validate our framework by simulating different examples of two main robotic manipulation skills encapsulated in SPD matrix form: (*i*) learning variable impedance skills, and (*ii*) learning manipulability ellipsoids, Section 6. We discuss and conclude our framework in Sections 7 and 8, respectively.

2. Related work

In this section, we review related works on SPD learning algorithms as well as two primary applications of SPD-based skills: learning of variable impedance skills from demonstrations, and use of manipulability-based redundancy resolution. For the sake of comparison, the main contributions of the state-of-the-art approaches and our approach are summarized in Table 1.

Learning SPD profiles: Learning SPD profiles from demonstrations has received considerable attention in recent years. Jaquier et al. [12] proposed a tensor-based formulation of GMM and GMR on the $\mathcal{S}^m_{++}$, which is capable of learning and reproducing SPD-matrix-based skills without further reparametrization of the data. Although Jaquier et al. approach inherits from the classical GMM/GMR the capability of learning with multi-dimensional inputs, but it also inherits its limitation in adapting to different start-, via- and end-SPD-points on a SPD manifold.

Zeestraten et al. [1] proposed to model distributions of orientation data using the Riemannian manifold $\mathcal{S}^3$. The work extended the Task-Parameterized GMM (TP-GMM) to Riemannian manifolds, allowing the adaptation to new start- and goal-points. The formulation targets data in $\mathcal{S}^3$ but the method could potentially be extended – given that $\mathcal{S}^m_{++}$ is also a Riemannian manifold – to adaptation of SPD start- and end-SPD-points. However, TP-GMM adaptation does not support via-SPD-points adaptation.

Recently, Abu-Dakka and Kyrki [13] proposed to use Riemannian metrics to reformulate DMPs such that the resulting formulation can operate with SPD data in the SPD manifold. Their formulation is able to adapt to a new goal-SPD-point, but not to new via-SPD-points.

Learning variable impedance skills: Impedance control[2] specifies a dynamic relationship between position and force and plays an important role in tasks that require physical interaction between a robot and the environment [24]. Namely, impedance control helps overcome position uncertainties and subsequently avoid large impact forces, since robots are controlled to modulate their motion or compliance according to force perceptions. Note that the focus of this work is not on the learning of the controllers themselves but, generally, on learning SPD-profiles, of which

[1] Throughout the paper we will use this terminology to refer to manipulation skills that require some form of parameterization that relies on SPD matrices (e.g. impedance control, where stiffness and damping matrices are SPD).

[2] In this paper, we refer by variable impedance to variable stiffness, however, the proposed framework can be used to learn and adapt damping controllers.

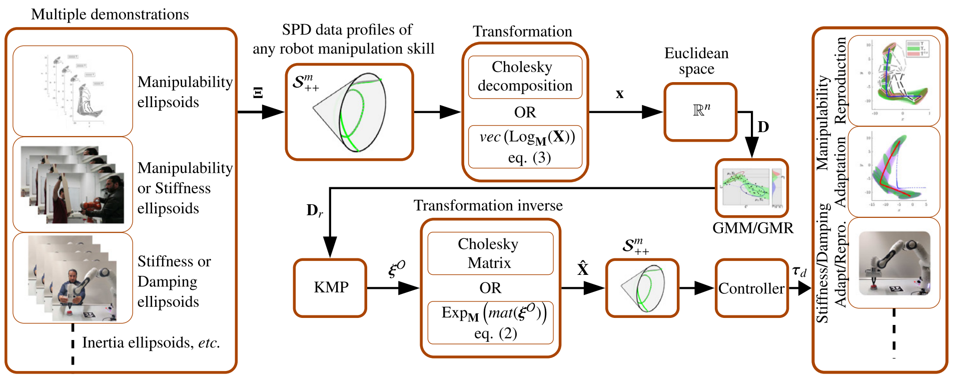
Multiple demonstrations
Manipulability ellipsoids
Manipulability or Stiffness ellipsoids
Stiffness or Damping ellipsoids
Inertia ellipsoids, *etc.*
Ξ
SPD data profiles of any robot manipulation skill
$\mathcal{S}^m_{++}$
Transformation
Cholesky decomposition
OR
$vec(\text{Log}_{\mathbf{M}}(\mathbf{X}))$ eq. (3)
$\mathbf{x}$
Euclidean space
$\mathbb{R}^n$
$\mathbf{D}$
GMM/GMR
$\mathbf{D}_r$
Transformation inverse
KMP
ξ^O
Cholesky Matrix
OR
$\text{Exp}_{\mathbf{M}}(mat(\xi^O))$ eq. (2)
$\hat{\mathbf{X}}$
$\mathcal{S}^m_{++}$
Controller
τ_d
Manipulability Reproduction
Manipulability Adaptation
Stiffness/Damping Adapt/Repro.

Fig. 2. Diagram of the proposed framework, from human demonstrations (leftmost box) to a robot performing the tasks (rightmost box).

impedance gains are only a particular case. For details on learning variable impedance control the reader is referred to [25].

Variable stiffness is a central use case of SPD-based skill representations and learning such skills has been of interest to several authors: Kronander et al. [23] proposed to use GMM in Euclidean space to directly encode the vectorized version of full stiffness matrices obtained by using Cholesky decomposition. The desired stiffnesses were physically demonstrated by a teacher wiggling the robot to indicate the desired stiffness. Abu-Dakka et al. [10] proposed a LfD framework to learn force-based variable stiffness skills. Both forces and stiffness profiles were probabilistically encoded using two different approaches: geometry-aware GMM/GMR [12] and Euclidean-based GMM/GMR [26]. However, none of the approaches presented above can adapt to new start-, end- or via-SPD-points in the SPD manifold.

Modeling manipulability ellipsoids: Manipulability analysis plays an important role in identifying the suitability of given configurations to execute specific tasks by providing a sort of performance measure known as manipulability index [2].

Manipulability measures have been used in many robotics applications. Chiu [27] proposed to use it to measure the compatibility of robot postures for fine versus coarse manipulation. Chiacchio [28] exploited manipulability ellipsoids within inverse kinematics approaches to compute optimal time trajectory planning. In order to obtain high task-space dexterity and singularity-free joint trajectories, designing tasks with high manipulability is often desired. However, solely maximizing the manipulability to achieve high dexterity in motions causes the reverse effect on the force capability [29].

Because of the complementarity of force capability and manipulability, a particular level of manipulability can be sought. Lee [29] showed that for required motion and force trajectories of a given task in dual arm manipulator, a profile of desired manipulability ellipsoids can be predetermined. Similarly, an optimization-based approach was developed by Lee et al. [5] for a humanoid robot to select a reaching posture by following manually-specified desired manipulability volumes. Note that both approaches [29] and [5] are task-oriented and need to manually predetermine the desired robot manipulability. Such procedure requires a careful motion analysis, which is often a burden on the user. Moreover, the aforementioned approaches overlooked the fact that the manipulability lies on the SPD manifold. Differently, Rozo et al. [6] applied the geometry-aware GMM/GMR [12] to solve the manipulability transfer problem between teacher and learner robots in order to learn and retrieve desired manipulability ellipsoids based on imitation learning. Despite the fact that Rozo's work tackles the manipulability on the manifold of SPD matrices, their approach suffers from the same limitations as [12]. In other words, the approach in [6] is still unable to adapt to new start-, via-, or end-SPD-points in the SPD manifold.

Table 2
Description of key notations.

m	$\triangleq$	matrix dimension
n	$\triangleq$	dimension of vector space
L	$\triangleq$	number of demonstrations
T	$\triangleq$	number of datapoints
G	$\triangleq$	number of Gaussian components
$\mathbf{X}$	$\triangleq$	an arbitrary SPD matrix
$\boldsymbol{\xi}^{\mathcal{I}}$	$\triangleq$	input vector
$\boldsymbol{\Xi} = \{\{\boldsymbol{\xi}^{\mathcal{I}}_{t,l}, \mathbf{X}_{t,l}\}_{t=1}^{T}\}_{l=1}^{L}$	$\triangleq$	set of SPD profiles
$\mathbf{D} = \{\{\boldsymbol{\xi}^{\mathcal{I}}_{t,l}, \mathbf{x}_{t,l}\}_{t=1}^{T}\}_{k=1}^{L}$	$\triangleq$	transformed data obtained from $\boldsymbol{\Xi}$, Sections 4.1 and 4.2
$\mathbf{D}_r = \{\boldsymbol{\xi}^{\mathcal{I}}_t, \hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t\}_{t=1}^{T}$	$\triangleq$	probabilistic reference trajectory extracted from $\mathbf{D}$
$\mathcal{M}$	$\triangleq$	a Riemannian manifold
$\mathcal{T}_{\mathbf{P}}\mathcal{M}$	$\triangleq$	a tangent space of $\mathcal{M}$ at point $\mathbf{P}$
$\mathcal{S}^m_{++}$	$\triangleq$	$m \times m$ SPD manifold
$\mathbf{Sym}^m$	$\triangleq$	$m \times m$ symmetric matrices space
$\mathbf{M}$	$\triangleq$	the mean of $\{\{\mathbf{X}_{t,l}\}_{t=1}^{T}\}_{l=1}^{L}$
$vec(\cdot)$	$\triangleq$	a function transforms symmetric matrices into vectors using Mandel's notation
$\boldsymbol{\xi}^{\mathcal{I}*}$	$\triangleq$	new test input
$k(\cdot,\cdot)$	$\triangleq$	kernel function
$\boldsymbol{\mu}, \boldsymbol{\Sigma}$	$\triangleq$	expanded vector and matrix

3. Proposed framework

The probabilistic distribution of multiple demonstrations often encapsulates important motion features [16,20] and further facilitates the design of optimal controllers [11]. For that reason, we propose to transform SPD data into Euclidean space, which hence enables the probabilistic modeling of transformed trajectories (see Section 4). Afterwards, the distribution of the transformed data (see Section 5.1) is exploited using a kernelized approach, whose predictions allow for the retrieval of proper SPD data (see Section 5.2). The whole pipeline of the proposed framework is illustrated in Fig. 2. We summarize key notations used throughout this paper in Table 2.

4. From the SPD manifold to Euclidean space

Let us define $\mathcal{S}^m_{++}$ as the set of $m \times m$ SPD matrices, $\mathbf{X} \in \mathcal{S}^m_{++}$ as an arbitrary SPD matrix, and $\boldsymbol{\Xi} = \{\{\boldsymbol{\xi}^{\mathcal{I}}_{t,l}, \mathbf{X}_{t,l}\}_{t=1}^{T}\}_{l=1}^{L}$ as a training data set containing inputs and SPD data. $\boldsymbol{\xi}^{\mathcal{I}}_{t,l} \in \mathbb{R}^b$ is the input vector (e.g., time, force, Cartesian position, *etc.*) at t-th time step from the l-th demonstration. Furthermore, let us denote $\mathbf{x}_{t,l} \in \mathbb{R}^n$

as the vectorized version of the projection of $\mathbf{X}_{t,l}$ onto Euclidean space, which leads to a new training set $\mathbf{D} = \{\{\boldsymbol{\xi}^{\mathcal{I}}_{t,l}, \mathbf{x}_{t,l}\}_{t=1}^{T}\}_{l=1}^{L}$, which will be probabilistically encoded later in Section 5. In order to compute the vectorization of $\mathbf{X}_{t,l}$, we implement two different approaches: (i) Cholesky-based approach, and (ii) Riemannian metric-based approach.

4.1. Cholesky-based projection

In order to reduce the dimensionality of the data $\mathbf{X}_{t,l}$ while ensuring that the resulting matrices are always SPD, $\mathbf{X}_{t,l}$ can be projected onto a Euclidean space variable $\mathbf{x}_{t,l}$ using Cholesky decomposition by first extracting the Cholesky factor $\boldsymbol{\mathcal{A}}_{t,l}$ (from $\mathbf{X}_{t,l} = \boldsymbol{\mathcal{A}}^{\mathsf{T}}_{t,l}\boldsymbol{\mathcal{A}}_{t,l}$) and then vectorizing the factor $\mathbf{a}_{t,l}$ [10]. This representation reduces the data space dimensionality to $m(m+1)/2$ and avoids replicating information due to the symmetry. For example, given $\mathbf{A} \in \boldsymbol{\mathcal{S}}^2_{++}$, then its Cholesky vector representation $\mathbf{a}$ is derived as follows

$$\mathbf{A} = \boldsymbol{\mathcal{A}}^{\mathsf{T}}\boldsymbol{\mathcal{A}}, \quad \boldsymbol{\mathcal{A}} = \begin{bmatrix} a_1 & a_2 \\ 0 & a_3 \end{bmatrix}, \qquad \therefore \mathbf{a} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^{\mathsf{T}} \tag{1}$$

Here, $\mathbf{x}_{t,l} = \mathbf{a}_{t,l}$ and $\mathbf{D} = \{\{\boldsymbol{\xi}^{\mathcal{I}}_{t,l}, \mathbf{a}_{t,l}\}_{t=1}^{T}\}_{l=1}^{L}$ which will be used to estimate the joint probability distribution $\mathcal{P}(\boldsymbol{\xi}^{\mathcal{I}}, \mathbf{a})$ (Section 5.1).

4.2. Riemannian metric-based projection

In mathematics, a manifold is a topological space that locally resembles Euclidean space near each point. A Riemannian manifold $\mathcal{M}$ is a smooth manifold that is equipped with a Riemannian metric. For every point $\mathbf{P}$ in a manifold $\mathcal{M}$, i.g. $\mathbf{P} \in \mathcal{M}$, one can attach a tangent space $\mathcal{T}_{\mathbf{P}}\mathcal{M}$. For $\boldsymbol{\mathcal{S}}^m_{++}$, this tangent space corresponds to the space of symmetric matrices $\mathbf{Sym}^m$. The metric in the tangent space is flat, which allows the use of classical arithmetic tools despite the original data being $\boldsymbol{\mathcal{S}}^m_{++}$. Note that the space of $m \times m$ SPD matrices $\boldsymbol{\mathcal{S}}^m_{++}$ can be represented as the interior of a convex cone embedded in its tangent space.

To operate on the tangent space, a mapping system is required to switch between $\mathcal{T}_{\mathbf{P}}\mathcal{M}$ and $\mathcal{M}$. These mapping operators are:

- The exponential map $\mathrm{Exp}_{\mathbf{P}}(\boldsymbol{\varGamma})\colon \mathcal{T}_{\mathbf{P}}\mathcal{M} \mapsto \mathcal{M}$ is a function that maps a point $\boldsymbol{\varGamma}$ in the tangent space to a point $\mathbf{Q} \in \mathcal{M}$, so that it lies on the geodesic starting from $\mathbf{P}$ in the direction of $\boldsymbol{\varGamma}$. In the case of $\boldsymbol{\mathcal{S}}^m_{++}$, $\mathrm{Exp}_{\mathbf{P}}(\boldsymbol{\varGamma})$ is defined as [30]:
$$\mathrm{Exp}_{\mathbf{P}}(\boldsymbol{\varGamma}) = \mathbf{P}^{\frac{1}{2}}\mathrm{expm}(\mathbf{P}^{-\frac{1}{2}}\boldsymbol{\varGamma}\,\mathbf{P}^{-\frac{1}{2}})\mathbf{P}^{\frac{1}{2}}. \tag{2}$$
- The logarithmic map $\mathrm{Log}_{\mathbf{P}}(\mathbf{Q})\colon \mathcal{M} \mapsto \mathcal{T}_{\mathbf{P}}\mathcal{M}$ is the inverse of the exponential map and defined as follows for the case of $\boldsymbol{\mathcal{S}}^m_{++}$ manifold [30]:
$$\mathrm{Log}_{\mathbf{P}}(\mathbf{Q}) = \mathbf{P}^{\frac{1}{2}}\mathrm{logm}(\mathbf{P}^{-\frac{1}{2}}\mathbf{Q}\,\mathbf{P}^{-\frac{1}{2}})\mathbf{P}^{\frac{1}{2}}. \tag{3}$$

where $\mathrm{expm}(\cdot)$ and $\mathrm{logm}(\cdot)$ are the matrix exponential and logarithmic functions.

In order to project the demonstrated data $\{\{\mathbf{X}_{t,l}\}_{t=1}^{T}\}_{l=1}^{L}$ onto Euclidean space, we need first to introduce an auxiliary SPD matrix $\mathbf{M} \in \boldsymbol{\mathcal{S}}^m_{++}$. For a proper selection of $\mathbf{M}$, we can use [12] to estimate it as the mean of $\{\{\mathbf{X}_{t,l}\}_{t=1}^{T}\}_{l=1}^{L}$. Subsequently, we project the demonstration data $\{\{\mathbf{X}_{t,l}\}_{t=1}^{T}\}_{l=1}^{L}$ into the tangent space $\mathcal{T}_{\mathbf{M}}\mathcal{M}$ using (3), where $\mathrm{Log}_{\mathbf{M}}(\mathbf{X}_{t,l}) = \boldsymbol{\mathcal{H}}_{t,l}$, $\boldsymbol{\mathcal{H}}_{t,l} \in \mathbf{Sym}^m$. Afterwards, Mandel's representation of a symmetric matrix can be used to reduce the data space dimensionality by vectorizing $\boldsymbol{\mathcal{H}}_{t,l}$, such that

$$vec(\boldsymbol{\mathcal{H}}_{t,l}) = \mathbf{h}_{t,l}, \quad \mathbf{h}_{t,l} \in \mathbb{R}^n, \tag{4}$$

where $vec(\cdot)$ is a function that transforms symmetric matrices into vectors using Mandel's notation. For example, given a 2×2 symmetric matrix, the transformation is

$$vec\left(\begin{bmatrix} a & b \\ b & d \end{bmatrix}\right) = \begin{bmatrix} a \\ d \\ \sqrt{2}b \end{bmatrix}. \tag{5}$$

This representation reduces the data space dimensionality to $m(m+1)/2$ and avoids replicating information due to the symmetry.

In this case, $\mathbf{x}_{t,l} = \mathbf{h}_{t,l}$ which leads to a training dataset $\mathbf{D} = \{\{\boldsymbol{\xi}^{\mathcal{I}}_{t,l}, \mathbf{h}_{t,l}\}_{t=1}^{T}\}_{l=1}^{L}$ so the joint probability distribution $\mathcal{P}(\boldsymbol{\xi}^{\mathcal{I}}, \mathbf{h})$ can be estimated in Section 5.1.

5. Probabilistic learning of SPD profiles

The training dataset $\mathbf{D}$ can be encoded using different probabilistic models [17]. In this paper, a mixture of G Gaussians components is used to estimate the joint probability distribution $\mathcal{P}(\boldsymbol{\xi}^{\mathcal{I}}, \boldsymbol{\xi}^{\mathcal{O}})$, with $\boldsymbol{\xi}^{\mathcal{O}} = \mathbf{x}$ through expectation–maximization [26]. Subsequently, we use GMR [26] to retrieve the trajectory distribution that initializes KMP (Section 5.2).

5.1. Probabilistic modeling of SPD data

Given the training dataset $\mathbf{D}$, we use GMM to model the joint probability distribution $\mathcal{P}(\boldsymbol{\xi}^{\mathcal{I}}, \boldsymbol{\xi}^{\mathcal{O}})$, leading to

$$\mathcal{P}(\boldsymbol{\xi}^{\mathcal{I}}, \boldsymbol{\xi}^{\mathcal{O}}) \sim \sum_{g=1}^{G} \pi_g \mathcal{N}\Big(\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g\Big), \tag{6}$$

where $\boldsymbol{\pi}_g$, $\boldsymbol{\mu}_g = \begin{bmatrix} \boldsymbol{\mu}^{\mathcal{I}}_g \\ \boldsymbol{\mu}^{\mathcal{O}}_g \end{bmatrix}$, $\boldsymbol{\Sigma}_g = \begin{bmatrix} \boldsymbol{\Sigma}^{\mathcal{I}}_g & \boldsymbol{\Sigma}^{\mathcal{IO}}_g \\ \boldsymbol{\Sigma}^{\mathcal{OI}}_g & \boldsymbol{\Sigma}^{\mathcal{O}}_g \end{bmatrix}$ correspond to the prior, mean, and covariance of the g-th Gaussian component, respectively. G denotes the number of Gaussian components. Later, we employ GMR to retrieve the conditional distribution $\mathcal{P}(\boldsymbol{\xi}^{\mathcal{O}}_t|\boldsymbol{\xi}^{\mathcal{I}}_t)$ as

$$\mathcal{P}\left(\boldsymbol{\xi}^{\mathcal{O}}_t|\boldsymbol{\xi}^{\mathcal{I}}_t\right) = \sum_{g=1}^{G} h_g\left(\boldsymbol{\xi}^{\mathcal{I}}_t\right)\mathcal{N}\Big(\hat{\boldsymbol{\mu}}_g\left(\boldsymbol{\xi}^{\mathcal{I}}_t\right), \hat{\boldsymbol{\Sigma}}_g\Big), \tag{7}$$

where

$$h_g\left(\boldsymbol{\xi}^{\mathcal{I}}_t\right) = \frac{\pi_g \mathcal{N}\left(\boldsymbol{\xi}^{\mathcal{I}}_t|\boldsymbol{\mu}^{\mathcal{I}}_g, \boldsymbol{\Sigma}^{\mathcal{I}}_g\right)}{\sum_{j=1}^{G} \pi_j \mathcal{N}\left(\boldsymbol{\xi}^{\mathcal{I}}_t|\boldsymbol{\mu}^{\mathcal{I}}_j, \boldsymbol{\Sigma}^{\mathcal{I}}_j\right)} \tag{8}$$

$$\hat{\boldsymbol{\mu}}_g\left(\boldsymbol{\xi}^{\mathcal{I}}_t\right) = \boldsymbol{\mu}^{\mathcal{O}}_g + \boldsymbol{\Sigma}^{\mathcal{OI}}_g \boldsymbol{\Sigma}^{\mathcal{I}^{-1}}_g\left(\boldsymbol{\xi}^{\mathcal{I}}_t - \boldsymbol{\mu}^{\mathcal{I}}_g\right) \tag{9}$$

$$\hat{\boldsymbol{\Sigma}}_g = \boldsymbol{\Sigma}^{\mathcal{O}}_g - \boldsymbol{\Sigma}^{\mathcal{OI}}_g \boldsymbol{\Sigma}^{\mathcal{I}^{-1}}_g \boldsymbol{\Sigma}^{\mathcal{IO}}_g. \tag{10}$$

As shown in [17,26], we can approximate the distribution in (7) by a single Gaussian distribution, i.e., $\boldsymbol{\xi}^{\mathcal{O}}_t|\boldsymbol{\xi}^{\mathcal{I}}_t \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t)$. Finally, for a given input sequence $\{\boldsymbol{\xi}^{\mathcal{I}}_t\}_{t=1}^{T}$, we can derive a probabilistic reference trajectory $\mathbf{D}_r = \{\boldsymbol{\xi}^{\mathcal{I}}_t, \hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t\}_{t=1}^{T}$, which can be exploited directly by using KMP, since $\mathbf{D}_r$ encapsulates the distribution of training trajectories in $\mathbf{D}$.

Noteworthy, the upper triangle matrix $\hat{\boldsymbol{\mathcal{A}}}$ (Section 4.1) or the symmetric matrix $\hat{\boldsymbol{\mathcal{H}}}$ (Section 4.2) can be re-constructed from the output $\hat{\boldsymbol{\mu}}_t$ of GMR, and accordingly the desired $\hat{\mathbf{X}}$ can be computed. However, in this paper, we will introduce further treatment to GMR output $\{\hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t\}_{t=1}^{T}$ as explained in the next section.

Algorithm 1 *Learning and adaptation of SPD profiles using kernelized approach*

```
1: Prepare reference trajectory $\mathbf{D}_r = \{\boldsymbol{\xi}_t^{\mathcal{I}}, \hat{\boldsymbol{\mu}}_t, \hat{\boldsymbol{\Sigma}}_t\}_{t=1}^T$ from
   expert demonstrations (Section 5.1)
     – Collect demonstrations $\boldsymbol{\Xi} = \{\{\boldsymbol{\xi}_{t,l}^{\mathcal{I}}, \mathbf{X}_{t,l}\}_{t=1}^T\}_{l=1}^L$
     – Transform $\boldsymbol{\Xi}$ into Euclidean space using Cholesky
       decomposition (Section 4.1) or Riemannian metrics
       (Section 4.2)
     – Extract $\mathbf{D}_r$ through (7)
2: Transform the desired SPD data $\tilde{\boldsymbol{\Xi}} = \{\tilde{\boldsymbol{\xi}}_l^{\mathcal{I}}, \tilde{\mathbf{X}}_l\}_{l=1}^L$ into desired
   Euclidean data $\tilde{\mathbf{D}} = \{\tilde{\boldsymbol{\xi}}_l^{\mathcal{I}}, \tilde{\boldsymbol{\xi}}_l^{\mathcal{O}}\}_{l=1}^L$ and update the reference
   trajectory $\mathbf{D}_r$ (Section 5.3)
3: Prediction of new SPD data
     – Define $\lambda$ and $k(\cdot,\cdot)$
     – *Input*: query point $\boldsymbol{\xi}^{\mathcal{I}*}$
     – Compute Eq. (11) for prediction
     – Project predicted data to SPD manifold (Sections
       4.1/4.2)
     – *Output*: $\hat{\mathbf{X}}$
```

5.2. Kernelized learning approach

In this section, we exploit $\mathbf{D}_r$ in KMP [17]. For a new test point $\boldsymbol{\xi}^{\mathcal{I}*}$, we predict the corresponding output $\boldsymbol{\xi}^{\mathcal{O}}(\boldsymbol{\xi}^{\mathcal{I}*})$ as [17]

$$\boldsymbol{\xi}^{\mathcal{O}}(\boldsymbol{\xi}^{\mathcal{I}*}) = \mathbf{k}^*(\boldsymbol{\mathcal{K}} + \lambda\boldsymbol{\Sigma})^{-1}\boldsymbol{\mu}. \tag{11}$$

where

$$\mathbf{k}^* = \left[\mathbf{k}(\boldsymbol{\xi}^{\mathcal{I}*}, \boldsymbol{\xi}_1^{\mathcal{I}}) \quad \mathbf{k}(\boldsymbol{\xi}^{\mathcal{I}*}, \boldsymbol{\xi}_2^{\mathcal{I}}) \quad \cdots \quad \mathbf{k}(\boldsymbol{\xi}^{\mathcal{I}*}, \boldsymbol{\xi}_T^{\mathcal{I}})\right] \tag{12}$$

$$\boldsymbol{\mathcal{K}} = \begin{bmatrix} \mathbf{k}(\boldsymbol{\xi}_1^{\mathcal{I}}, \boldsymbol{\xi}_1^{\mathcal{I}}) & \mathbf{k}(\boldsymbol{\xi}_1^{\mathcal{I}}, \boldsymbol{\xi}_2^{\mathcal{I}}) & \cdots & \mathbf{k}(\boldsymbol{\xi}_1^{\mathcal{I}}, \boldsymbol{\xi}_T^{\mathcal{I}}) \\ \mathbf{k}(\boldsymbol{\xi}_2^{\mathcal{I}}, \boldsymbol{\xi}_1^{\mathcal{I}}) & \mathbf{k}(\boldsymbol{\xi}_2^{\mathcal{I}}, \boldsymbol{\xi}_2^{\mathcal{I}}) & \cdots & \mathbf{k}(\boldsymbol{\xi}_2^{\mathcal{I}}, \boldsymbol{\xi}_T^{\mathcal{I}}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{k}(\boldsymbol{\xi}_T^{\mathcal{I}}, \boldsymbol{\xi}_1^{\mathcal{I}}) & \mathbf{k}(\boldsymbol{\xi}_T^{\mathcal{I}}, \boldsymbol{\xi}_2^{\mathcal{I}}) & \cdots & \mathbf{k}(\boldsymbol{\xi}_T^{\mathcal{I}}, \boldsymbol{\xi}_T^{\mathcal{I}}) \end{bmatrix} \tag{13}$$

is a matrix evaluating a chosen kernel function $k(.,.)$ and $\mathbf{k}(\boldsymbol{\xi}_i^{\mathcal{I}}, \boldsymbol{\xi}_j^{\mathcal{I}}) = k(\boldsymbol{\xi}_i^{\mathcal{I}}, \boldsymbol{\xi}_j^{\mathcal{I}})\mathbf{I}_n$, where $i,j = 1,2,\ldots,T$. λ is an empirically chosen hyperparameter. $\boldsymbol{\mu} = [\hat{\boldsymbol{\mu}}_1^{\mathsf{T}}\hat{\boldsymbol{\mu}}_2^{\mathsf{T}}\cdots\hat{\boldsymbol{\mu}}_T^{\mathsf{T}}]^{\mathsf{T}}$ and $\boldsymbol{\Sigma} = \text{blockdiag}(\hat{\boldsymbol{\Sigma}}_1\hat{\boldsymbol{\Sigma}}_2\cdots\hat{\boldsymbol{\Sigma}}_T)$.

The kernel function depends on the characteristics of the training data. In this paper, we use the squared-exponential kernel

$$k(\boldsymbol{\xi}_i^{\mathcal{I}}, \boldsymbol{\xi}_j^{\mathcal{I}}) = \sigma_f^2 \exp\left(-r\,\|\boldsymbol{\xi}_i^{\mathcal{I}} - \boldsymbol{\xi}_j^{\mathcal{I}}\|^2\right), \tag{14}$$

a common choice in the literature, with hyperparameters $\{r, \sigma_f^2\}$.

In summary, given a query input $\boldsymbol{\xi}^{\mathcal{I}*}$, we predict its corresponding output $\boldsymbol{\xi}^{\mathcal{O}}(\boldsymbol{\xi}^{\mathcal{I}*})$ through (11). The predicted result is used to retrieve the corresponding SPD data by constructing the Cholesky matrix as described in Section 4.1, or by using Riemannian metrics as shown in 4.2. For the sake of clarity, we follow either of the following options: (*i*) when the predicted vector from (11) $\boldsymbol{\xi}^{\mathcal{O}}$ is a Cholesky vector (Section 4.1), then we can reconstruct the Cholesky factor $\boldsymbol{\mathcal{A}}$ as in (1), which allows us to compute the Cholesky Matrix (the SPD matrix) $\mathbf{A}$ through $\mathbf{A} = \boldsymbol{\mathcal{A}}^{\mathsf{T}}\boldsymbol{\mathcal{A}}$, however, (*ii*) when $\boldsymbol{\xi}^{\mathcal{O}}$ corresponds to Mandel's representation of a symmetric matrix (Section 4.2), then by applying the inverse of (5), we can reconstruct its correspondent symmetric matrix $\boldsymbol{\mathcal{H}}$ in the tangent space $\mathcal{T}_{\mathbf{M}}\mathcal{M}$. Later we project this matrix back to the $\mathcal{S}_{++}^m$ manifold using $\text{Exp}_{\mathbf{M}}(\boldsymbol{\mathcal{H}})$ operator as in (2) with respect to the same auxiliary SPD matrix $\mathbf{M}$. See Fig. 2 for an illustration of the pipeline of the framework.

5.3. Adaptation of SPD profiles

In this section, we extend the original capabilities of KMP to adapt to new desired start-, via-, and end-points (unseen in the demonstrations) to SPD data. Consider $\tilde{\boldsymbol{\Xi}}^0 = \{\tilde{\boldsymbol{\xi}}_l^{\mathcal{I},0}, \tilde{\mathbf{X}}_l^0\}_{l=1}^L$ to be the set of desired SPD data, where $\tilde{\boldsymbol{\xi}}^{\mathcal{I},0} \in \mathbb{R}^p$ and $\tilde{\mathbf{X}}^0 \in \mathcal{S}_{++}^m$. The transformation of this desired set into Euclidean space (using Sections 4.1 or 4.2) is denoted as $\tilde{\mathbf{D}} = \{\tilde{\boldsymbol{\xi}}_l^{\mathcal{I}}, \tilde{\boldsymbol{\xi}}_l^{\mathcal{O}}\}_{l=1}^L$. In addition, (11) requires the assignment of covariance matrices $\tilde{\boldsymbol{\Sigma}}^{\mathcal{O}}$ for each new desired point $\tilde{\boldsymbol{\xi}}^{\mathcal{O}}$ to control the adaptation precision (the smaller the covariance the higher the precision)[3]. Therefore, we have an additional reference trajectory $\tilde{\mathbf{D}}_r = \{\tilde{\boldsymbol{\xi}}_l^{\mathcal{I}}, \tilde{\boldsymbol{\xi}}_l^{\mathcal{O}}, \tilde{\boldsymbol{\Sigma}}_l^{\mathcal{O}}\}_{l=1}^L$ which can be concatenated with $\mathbf{D}_r$, yielding an extended reference trajectory $\bar{\mathbf{D}}_r = \{\bar{\boldsymbol{\xi}}_t^{\mathcal{I}}, \bar{\boldsymbol{\xi}}_t^{\mathcal{O}}, \bar{\boldsymbol{\Sigma}}_t^{\mathcal{O}}\}_{t=1}^{T+L}$ defined by

$$\begin{cases} \begin{cases} \bar{\boldsymbol{\xi}}_t^{\mathcal{I}} = \boldsymbol{\xi}_t^{\mathcal{I}}, \\ \bar{\boldsymbol{\xi}}_t^{\mathcal{O}} = \hat{\boldsymbol{\mu}}_t^{\mathcal{O}}, & \text{if } \; t \le T, \\ \bar{\boldsymbol{\Sigma}}_t^{\mathcal{O}} = \hat{\boldsymbol{\Sigma}}_t^{\mathcal{O}}, \end{cases} \\ \begin{cases} \bar{\boldsymbol{\xi}}_t^{\mathcal{I}} = \tilde{\boldsymbol{\xi}}_{t-T}^{\mathcal{I}}, \\ \bar{\boldsymbol{\xi}}_t^{\mathcal{O}} = \tilde{\boldsymbol{\xi}}_{t-T}^{\mathcal{O}}, & \text{if } \; T < t \le T+L. \\ \bar{\boldsymbol{\Sigma}}_t^{\mathcal{O}} = \tilde{\boldsymbol{\Sigma}}_{t-T}^{\mathcal{O}}, \end{cases} \end{cases} \tag{15}$$

Subsequently, the newly obtained trajectory $\bar{\mathbf{D}}_r$ can be used to generate n-dimensional trajectories in Euclidean space via (11) and retrieve the corresponding SPD profiles capable of passing through the desired points $\tilde{\boldsymbol{\Xi}}^0$.

The entire proposed framework of adapting SPD data profiles is summarized in Algorithm 1.

6. Experiments

We evaluated the proposed imitation learning framework using a couple of simulation setups in addition to a real experiment. The simulations and the real experiment serve as a proof of concept and consist of the following:

- Learning and adaptation of manipulability ellipsoids in trajectory tracking problem.
- Learning and adaptation of variable impedance skills.
- Adaptation of variable stiffness profile towards a via-stiffness point to perform push button task.

All algorithms have been implemented in MATLAB® using a workstation running Ubuntu 18.04 LTS with Intel Core i9–8950HK CPU @ 2.90 GHz × 12, 16 GB of RAM. The experiment is performed using the *Franka Emika Panda* robot.

6.1. Adaptation of manipulability ellipsoids

Manipulability is encapsulated as an SPD matrix and represented as an ellipsoid in a b-dimensional Euclidean space. The manipulability of a given configuration of a robot defines the capacity of change in pose of its end-effector. The manipulability

[3] The prediction in (11) can be interpreted as the maximization of the posterior given the distribution of training data, thus the covariance $\tilde{\boldsymbol{\Sigma}}_l^{\mathcal{O}}$ determines the adaptation error corresponding to each desired point $\{\tilde{\boldsymbol{\xi}}_l^{\mathcal{I}}, \tilde{\boldsymbol{\xi}}_l^{\mathcal{O}}\}$, see [31] for more details.

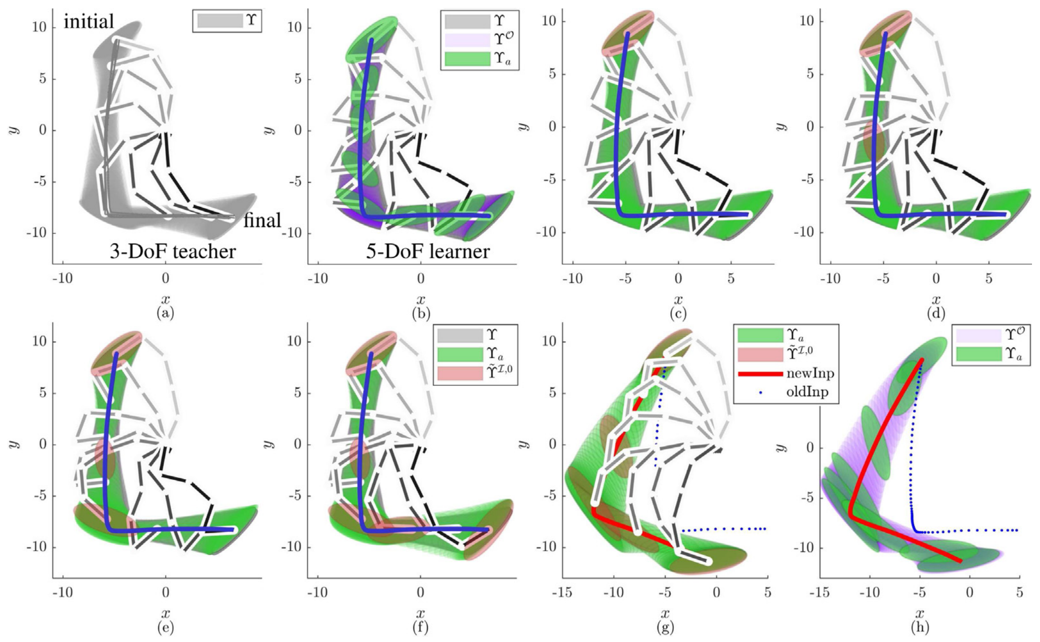

Fig. 3. Reproduction and adaptation of a L-shape trajectory-following task. Gray ellipsoids represent the demonstrated manipulability ellipsoids $\boldsymbol{\Upsilon}$, violet ones are for the desired manipulability ellipsoids $\boldsymbol{\Upsilon}^{\mathcal{O}}$ generated by the proposed framework, green ellipsoids represent the real measured robot manipulability $\boldsymbol{\Upsilon}_a$ (shown for a subset of points). Red ellipsoids correspond to new via-manipulability-ellipsoids $\tilde{\boldsymbol{\Upsilon}}^{\mathcal{I},0}$ which were not presented in the training data. *'initial'* and *'final'* indicate the starting and ending points of the tracking task, respectively. (a) A 3-DoFs teacher robot builds the set of training data $\boldsymbol{\Xi}$ to be used in the learning framework. (b) A 5-DoFs learner robot reproduces a manipulability profile using the proposed framework while tracking a Cartesian trajectory obtained using GMR. (c)–(f) A 5-DoFs learner robot tracks a reference Cartesian trajectory (similar to the demonstrations) and adapts its manipulability ellipsoids profile to new via-manipulability-ellipsoids. In each plot, different numbers of via-manipulability-ellipsoids are considered and we show one complete robot motion. Notice the different kinematic configurations of the robot due to the tracking of new via-manipulability-ellipsoids. (g) A 5-DoFs learner robot tracks a completely new reference Cartesian trajectory and adapts its manipulability ellipsoids profile to via-manipulability-ellipsoids. (h) Shows the matching between the desired manipulability profile (violet) and the current ones (green). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

of a robotic arm is derived from the relation between joint and task velocities ($\dot{\mathbf{q}}$ and $\dot{\mathbf{s}}$ respectively),

$$\dot{\mathbf{q}} = \mathbf{J}^{+}\dot{\mathbf{s}}, \tag{16}$$

where $\mathbf{q} \in \mathbb{R}^d$ and $\mathbf{J} \in \mathbb{R}^{6\times d}$ are the joint position and Jacobian, respectively, of a d-DoFs arm robot. The superscript $^+$ denotes the pseudo-inverse of a matrix. In addition, the set of joint velocities satisfying $\|\dot{\mathbf{q}}\| = 1$ corresponds to a point (in Cartesian space) lying on the surface of an ellipsoid. By substituting the unit norm constraint in (16),

$$\dot{\mathbf{q}}^{\mathsf{T}}\dot{\mathbf{q}} = \dot{\mathbf{s}}^{\mathsf{T}}\left(\mathbf{J}^{+}\right)^{\mathsf{T}}\mathbf{J}^{+}\dot{\mathbf{s}} = \dot{\mathbf{s}}^{\mathsf{T}}\left(\mathbf{J}\mathbf{J}^{\mathsf{T}}\right)^{+}\dot{\mathbf{s}}, \tag{17}$$

the manipulability ellipsoid $\boldsymbol{\Upsilon} = \left(\mathbf{J}\mathbf{J}^{\mathsf{T}}\right)^{+}$, where $\boldsymbol{\Upsilon} \in \mathcal{S}^m_{++}$ provides an intuitive indicator of the direction of possible movement of a configuration.

The objective here is to learn manipulability ellipsoids from demonstrations (from a teaching robot) and transfer them to a learner robot with higher DoFs. Unlike the work proposed by [6], we are not only aiming to reproduce the task by tracking a desired Cartesian trajectory, but also to *adapt* to new via-manipulability-ellipsoids and track a new Cartesian trajectory. In this simulation, we will use the toy example implemented in [6] to evaluate our framework.

A set of training data $\boldsymbol{\Xi} = \{\{\boldsymbol{\xi}^{\mathcal{I}}_{t,l}, \boldsymbol{\Upsilon}_{t,l}\}_{t=1}^{T}\}_{l=1}^{4}$ from a trajectory-following task is collected when a 3-DoFs robot tracks an L-shape Cartesian trajectory, see Fig. 3-(a). $\boldsymbol{\xi}^{\mathcal{I}}_{t,l} = \mathbf{s}_{t,l}$ is the Cartesian position of the robot end-effector. Let us define $\boldsymbol{\upsilon}$ as the vectorized form of $\boldsymbol{\Upsilon}$ after projecting it onto Euclidean space using either Cholesky factorization (Section 4.1) or Riemannian metrics (Section 4.2). Afterwards, we train a 4-states GMM over $\mathbf{D} = \{\{\boldsymbol{\xi}^{\mathcal{I}}_{t,l}, \boldsymbol{\upsilon}_{t,l}\}_{t=1}^{T}\}_{l=1}^{L}$ to estimate the joint probability distribution $\mathcal{P}(\boldsymbol{\xi}^{\mathcal{I}}, \boldsymbol{\upsilon})$. Then, the desired reference trajectory $\mathbf{D}_r$ is computed by GMR as explained in Section 5.1, which will be used as a reference trajectory in KMP (Section 5.2). During reproduction and adaptation, using KMP with hyperparameters $\sigma_f^2 = 1$, $r = 50$ and $\lambda = 1$, a 5-DoFs robot tracks unseen desired Cartesian position trajectory $\hat{\mathbf{s}}$. Note that, due to the different kinematic chain, without manipulability tracking the 5-DoF robot would likely exhibit a different manipulability profile than the 3-DoF one. The robot estimates its desired force $\mathbf{F}$ at the end-effector by obeying the following controller [6],

$$\boldsymbol{\tau}_d = \mathbf{J}^{\mathsf{T}}\mathbf{F} - \left(\mathbf{I} - \mathbf{J}^{\mathsf{T}}\bar{\mathbf{J}}^{\mathsf{T}}\right)\alpha\nabla g_t(\mathbf{q}); \quad \alpha > 0 \tag{18}$$

where $\bar{\mathbf{J}}$ is the inertia-weighted pseudo-inverse of $\mathbf{J}$, $\boldsymbol{\tau}_d$ is the desired joint torque, and the cost function $g_t(\mathbf{q})$ is defined as in [6],

$$\begin{aligned} g_t(\mathbf{q}) = \log&\left(\det\left(\frac{\boldsymbol{\Upsilon}^{\mathcal{O}}_t + \boldsymbol{\Upsilon}_{a,t}(\mathbf{q})}{2}\right)\right) \\ &- \frac{1}{2}\log\left(\det\left(\boldsymbol{\Upsilon}^{\mathcal{O}}_t\,\boldsymbol{\Upsilon}_{a,t}(\mathbf{q})\right)\right) \end{aligned} \tag{19}$$

where $\boldsymbol{\Upsilon}^{\mathcal{O}}$ denotes desired manipulability ellipsoids obtained from our framework. $\boldsymbol{\Upsilon}_a(\mathbf{q})$ represents actual manipulability ellipsoids of the current robot configuration.

Fig. 3 illustrates reproduction and adaptation of manipulability ellipsoids while the 5-DoFs robot end-effector is tracking an L-shape trajectory. Moreover, it illustrates how the desired manipulability ellipsoids $\boldsymbol{\Upsilon}^{\mathcal{O}}\left(\boldsymbol{\xi}^{\mathcal{I}*}\right)$ (in violet) and the current robot manipulability (in green) are smoothly following the demonstration ellipsoids $\boldsymbol{\Upsilon}$ (in gray), Fig. 3-(b). In addition, it shows the matching between the actual and desired manipulability ellipsoids.

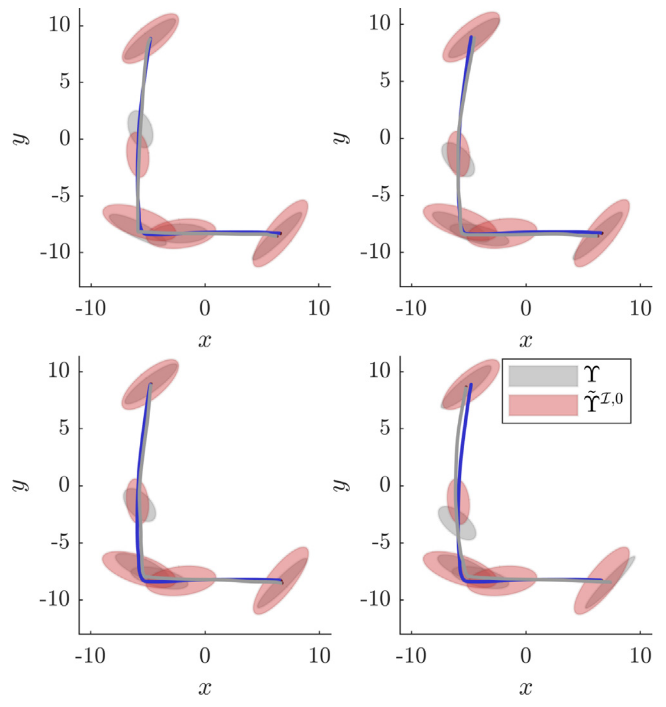

Fig. 4. Difference between the five designed via-manipulability-ellipsoids (red) and the ones from the four demonstrations (gray), at the corresponding time step. Each plot corresponds to one demonstration, in a total of four. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Figs. 3-(c)–(f) show the ability of the proposed framework to adapt the manipulability profile to new via-manipulability-ellipsoids $\tilde{\boldsymbol{\Upsilon}}^{\mathcal{I},0}$ (in red). This kind of adaptation is not achievable using GMR and to the best of our knowledge no previous approach tackled this adaptation problem. The new via-manipulability-points (see Fig. 4) have been designed in order to be different from the demonstration data and to maximize the manipulability of the robot at a specific Cartesian position. The figures illustrate how the manipulability profile smoothly adapt to $\tilde{\boldsymbol{\Upsilon}}^{\mathcal{I},0}$. These via-manipulability-ellipsoids gradually affect the evolution of the robot configuration, starting by one in (c) and ending with five in (f).

Fig. 3-(g) shows the adaptation of the manipulability profile to a completely new Cartesian trajectory and five via-manipulability-ellipsoids. For this last run, the matching between $\boldsymbol{\Upsilon}^{\mathcal{O}}(\boldsymbol{\xi}^{\mathcal{I}*})$ (in violet) and $\boldsymbol{\Upsilon}_a$ (in green) is shown in Fig. 3-(h).

6.2. Retrieving variable impedance skills

In this section, we propose to use a toy example of a 2-dimensional virtual Mass Spring–Damper system (MSD) in order to evaluate our framework by learning and adapting force-based variable impedance skills. Inspired by [10], the MSD system starts from the rest position with a horizontally-aligned stiffness ellipsoid $\mathbf{K}^{\mathcal{P}}$. To stimulate the MSD dynamics, external forces $\mathbf{f}^e$ are applied while $\mathbf{K}^{\mathcal{P}}$ is rotating through $\mathbf{R}^{\mathsf{T}}\mathbf{K}^{\mathcal{P}}\mathbf{R}$ until it ends up with a vertically-aligned ellipsoid as shown in Fig. 5-*bottom*. Afterwards, we transform these time-varying stiffness profiles into Euclidean space as mentioned in Sections 4.1 or 4.2. Subsequently, we use these transformed profiles along with $\mathbf{f}^e$ as a training dataset for 4-states GMM by defining $\boldsymbol{\xi}^{\mathcal{I}} = \mathbf{f}^e$ and $\boldsymbol{\xi}^{\mathcal{O}} = \mathbf{v}^{\mathcal{P}}$, where $\mathbf{v}^{\mathcal{P}}$ is the vectorized form of $\mathbf{K}^{\mathcal{P}}$. Then, the desired reference trajectory $\mathbf{D}_r$ is computed as explained in Section 5.1, which is used to initialize the KMP (Section 5.2). The KMP

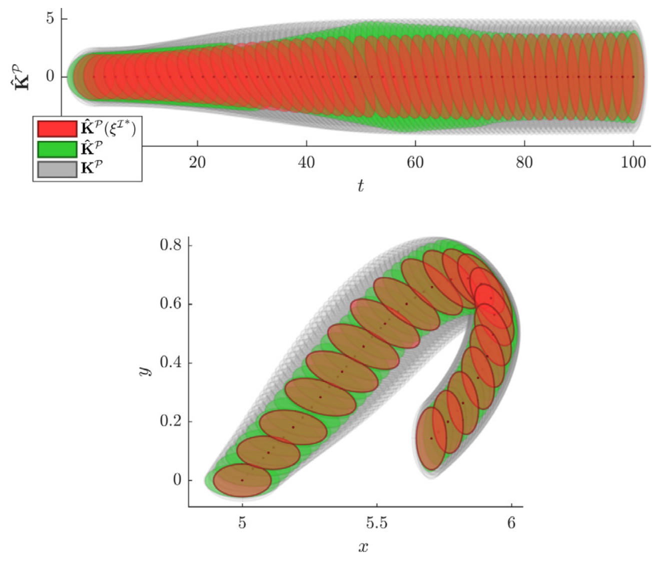

Fig. 5. Learning and reproduction of variable stiffness profile using the kernelized approach. The demonstrated stiffness ellipsoids are shown in gray. They start from the rest position at (5,0) with a horizontally-aligned ellipsoid and end with a vertically-aligned one. Green ellipsoids represent the reproduction using GMR, while the red ones are the result of the kernelized approach. *Top:* Stiffness profiles over time. *Bottom:* Stiffness profiles over the Cartesian trajectory of the MSD mass. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

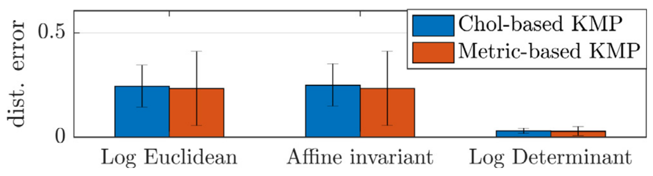

Fig. 6. Distance error comparison between Cholesky decomposition (Section 4.1) and Riemannian metrics (Section 4.2) on the reproduction of stiffness profile.

hyperparameters used in this simulation are $\sigma_f^2 = 1$, $r = 30$ and $\lambda = 1$.

Fig. 5 shows the reproduction case of the proposed framework where Cholesky factorization has been used to project full stiffness matrices into/from Euclidean space. Fig. 5-*top* shows the resulting stiffness profile over time, while Fig. 5-*bottom* shows it over the Cartesian trajectory of the MSD. We observe an accurate tracking of the initial stiffness profile, validating our approach for reproducing stiffness profiles.

Fig. 6 compares the average difference between all demonstrations and the stiffness profile obtained by the proposed framework, based on Cholesky decomposition and based on Riemannian metrics. In order to carry out the computation of distances in $\mathcal{S}_{++}^m$, we used different metrics, e.g. (*i*) log-Euclidean, (*ii*) Affine invariant, and finally (*iii*) Jensen–Bregman log-determinant. Readers may consult [32] for more details regarding these metrics.

The distance between each stiffness demonstration profile and the generated dataset from GMR and the proposed framework are analyzed in Fig. 7. Here we used the log-Euclidean metric to calculate the distances. It is clear that both approaches perform similarly in the reproduction case. However, the proposed approach results are smoother, as a result of the kernel treatment of KMP.

6.3. Adaptation of variable impedance skills

Another test is shown in Fig. 8 which illustrates the ability of the proposed approach to start from a new stiffness ellipsoid

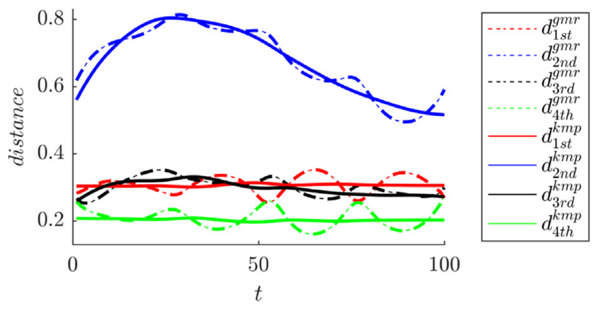

Fig. 7. Comparison of the distance between demonstrations and both results from GMR and the proposed approach. The distance is computed using log-Euclidean metric.

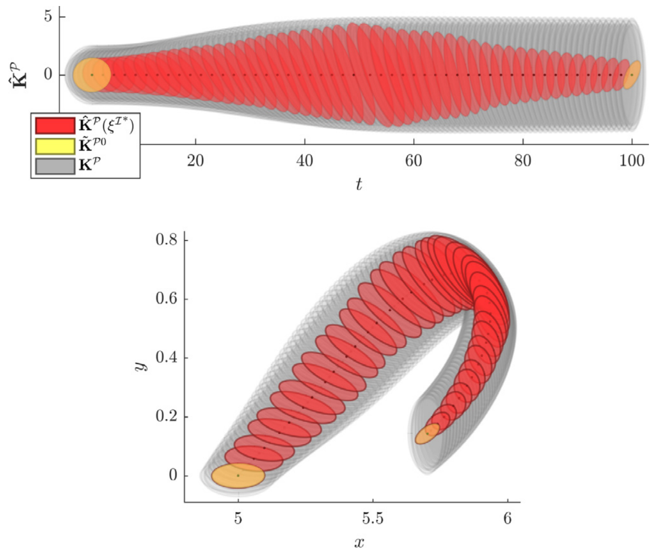

Fig. 8. Adaptation of variable stiffness profile using the proposed framework. Gray ellipsoids represent the demonstrated stiffness, while the output of the proposed approach are colored in red. The two yellow stiffness ellipsoids at the beginning and end of the trajectory are the start- and end-points of the stiffness profile. *Top:* Stiffness profiles over time. *Bottom:* Stiffness profiles over the Cartesian trajectory of the MSD mass. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and smoothly adapt to follow the demonstrations. Moreover, the figure shows how smoothly the algorithm adapts to a new goal ellipsoid, which is different from the demonstrations in terms of size and orientation. The two yellow ellipsoids in the trajectory represent the new desired stiffness $\tilde{\mathbf{K}}^{\mathcal{P},0}$. The set of red ellipsoids represents the output of KMP transformed into SPD manifold $\hat{\mathbf{K}}^{\mathcal{P}}(\boldsymbol{\xi}^{\mathcal{I}*})$, using Cholesky decomposition. The design of the desired stiffness ellipsoids is based on the direction of the desired stiffness.

6.4. Push-button task

In unstructured hostile environments such as industrial floors or other similar scenarios dangerous for humans, robots can be used for pushing objects, operating valves and doors, and other tasks that require application of forces to environments. In the same vein, we demonstrate the applicability of the proposed approach in a task where a robot without a force sensor needs to push a button.

In order to teach the robot how to perform the push button-task, a human teacher provides several demonstrations by kinesthetic teaching, see Fig. 1. The user holds the robot end-effector

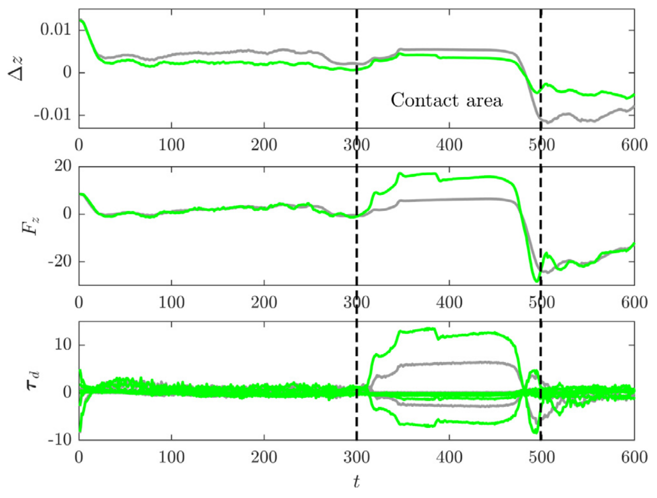

Fig. 9. *Top*: tracking errors in z-axis during the execution of the push-button task; *Middle*: desired force at the end-effector; *Bottom*: desired torques. Gray curves correspond to the reproduction, while the green curves correspond to the adaptation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and guides it along the desired trajectory (in this case, to push the button) in such a way that the desired task is successfully executed. During the demonstration we recorded only the Cartesian trajectories $\mathbf{s}$ as no interaction forces can be recorded without a force sensor. To train the GMM model, we used five Gaussian components with five demonstrations, each demonstration takes Cartesian position as input $\boldsymbol{\xi}^{\mathcal{I}} = \mathbf{s}$ and the Cholesky vector $\mathbf{v}^{\mathcal{P}}$ of a constant stiffness profile with value $\mathbf{K}^{\mathcal{P}} = [200, 50; 50, 200]$ for each datapoint as output ($\boldsymbol{\xi}^{\mathcal{O}} = \mathbf{v}^{\mathcal{P}}$). This model is used later by GMR to obtain $\mathbf{D}_r$ which subsequently will be exploited by KMP.

We tested both the reproduction of push-button task and the adaptation to via-SPD points.[4] In both cases we used a Cartesian impedance controller to execute the task by computing the desired forces $\mathbf{F}_d$ from the resulting virtual mass spring-damping system at the tip of the end-effector

$$\boldsymbol{\tau}_d = \mathbf{J}^{\mathsf{T}}\mathbf{F}_d + f(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \tag{20}$$

where $\mathbf{F}_d = -\hat{\mathbf{K}}^{\mathcal{P}}\Delta\mathbf{s} - \hat{\mathbf{K}}^{\mathcal{V}}(\mathbf{J}\dot{\mathbf{q}})$ and $f(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ is the robot dynamics model.

Reproduction. We reproduced the push-button task without any adaptation or force feedback information. In this case, the robot was unable to push the button, because the estimated stiffness profiles did not result in a high enough stiffness at the final point and thus failing to produce an enough desired force at the robot's tip to push the button. Fig. 9 shows the evolution of the robot execution in gray color for the reproduction. These results show that it is not always straightforward to design an appropriate stiffness profile in a way that all task constraints are fulfilled, even though the robot might be capable of tracking a desired trajectory. Note that this is not to say that an LfD stategy is a bad choice – the stiffness profiles up to the end-point in the push-button task were accurate enough to drive the robot to the end configuration. However, in some cases, additional human inputs are required for task success. In this paper, we do this by adding new via-SPD points.

[4] A video of the push-button experiment can be found at https://irobotics.aalto.fi/video-geometry-based-manipulation/.

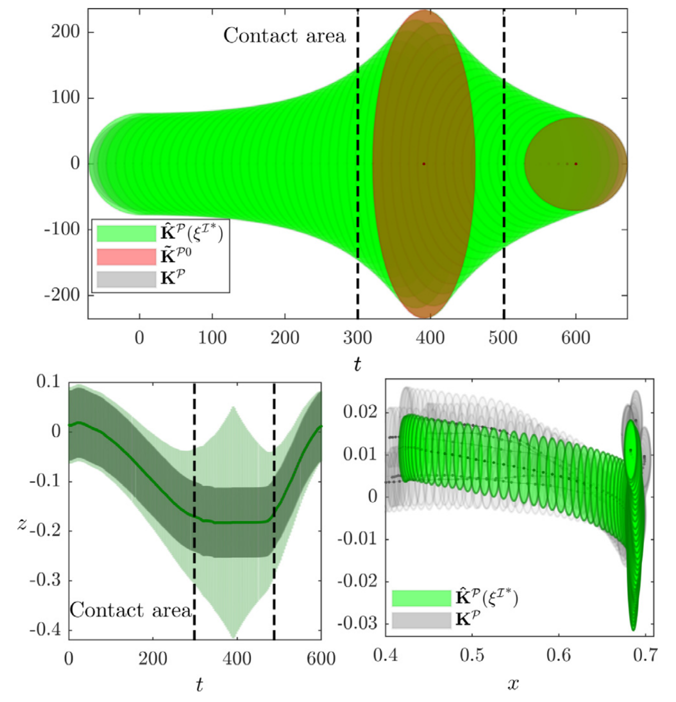

Fig. 10. Adaptation of the stiffness profile towards various desired via-SPD stiffness points. *Top:* The adaptation over time. *Bottom-left:* Adapted stiffness profile over z-axis (green) and demonstrations (gray). *Bottom-right:* Demonstration profiles (gray) and the adapted profile (green) over the Cartesian trajectory. The interaction between the tip of the robot and the button is shown between the dashed black lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Adaptation. Here we introduced two via-SPD points: one to make the robot stiffer in the direction of task evolution (z-axis) and the second one to make it compliant by reaching the end of the task. The green curves in Fig. 9 illustrate the tracking errors, force in z-axis at the tip of the robot, and the desired torques (20) during a successful execution of the task. Fig. 10 shows the evolution of the stiffness adaptation towards the via-SPD stiffness points over time (top). By including via-SPD points, the robot was able to accomplish the task. The interaction between the robot's tip and the button is bounded by the dashed black lines in both figures.

7. Discussion

The experimental results show that our approach allows for proper reproduction and adaptation of demonstrated datasets of SPD matrices. Note that due to the kernel treatment and the choice of a squared-exponential kernel (14), the adapted trajectories are smooth. Other kernels can alternatively be employed, depending on the properties of the data to be handled (e.g. a periodic kernel could be used for cyclic SPD-matrices in robot locomotion scenarios). We also saw that transforming the data into Euclidean space using the Riemannian metric resulted in more accurate reproductions than those from the Cholesky decomposition. This stems from the fact that $\mathcal{S}_{++}^m$ is a Riemannian manifold, hence its geometry is better captured by the logarithmic and exponential maps than the Cholesky factorization.

One difficulty that might arise in practical scenarios is the definition by the user of the new stiffness/damping matrices or manipulability ellipsoids that the robot should go through. After all it can be unintuitive for a human experimenter to reason in a $\mathcal{S}_{++}^m$ manifold as we are conditioned to think in Euclidean terms. In order to mitigate these practical limitations, interactive devices could be used to measure muscle activity from a human and map it to desired impedances [33,34]. In the same way, computer vision tools [35] can be used to track the human kinematic chain, compute manipulability ellipsoids at the end-effectors and add them to the reference database of the KMP on demand. Additionally, the structure provided by KMP for adaptation is prone to be used in Reinforcement Learning (RL) where one can potentially define a reward function that the robot would aim to maximize by placing start-/via-/end-points in different locations of the space. In this case, the generation of desired SPD matrices would be done by the RL algorithm. It should, however, be noted that RL can be used to solve the problem of learning SPD matrices, but its solution is different from one proposed here. RL typically relies on exploration and exploitation (i.e., errors and trials) to find the optimal solution, usually hundreds or thousands of trials are required to find a proper policy. Our solution does not need any extra learning of trials and errors as well as the definition of reward function, i.e., the optimal skill (encoded by SPD matrices) can be obtained immediately given a new inquiry point, allowing for straightforward applications in an online setting. As an example, RL could potentially have been used to fulfill the *push-button* task by letting the robot discover what was the appropriate stiffness that pressed the button (by defining the task using a proper reward), as opposed to defining the via-SPD point manually.

Finally it is important to note that, while here we limited the dimension of the inputs to 2, the kernelized structure of KMP allows for inputs of higher dimension. For instance, in [11] we used the position of a human partner to teach collaborative tasks to robots. This opens up several possibilities for further applications of our approach.

8. Conclusion

In this paper, we proposed a new learning-from-demonstration framework, using a kernelized approach, for learning SPD data profiles associated with multi-dimensional inputs. This approach is not only capable of reproducing new SPD data, but also capable of adapting them to towards arbitrary desired via-/end-SPD points. We exploited our approach in robotics manipulations tasks, where the skills to be learned are encapsulated in SPD matrices, e.g. tasks that require variable impedance skills, or others require high arm manipulability in the workspace. As KMP operates in Euclidean space, we proposed two different approaches to transform SPD data from/into Euclidean space. The first one is based on Cholesky decomposition while the second one is based on Riemannian metrics.

We evaluated the proposed approach using two simulated scenarios. The reported results show the ability of the kernelized learning approach to learn and adapt SPD data.

In future work, it will be interesting to develop tensor-based formulation of kernelized movement primitives on Riemannian manifold $\mathcal{S}_{++}^m$ which will allow for direct learning and adaptation of SPD data without any further reparametrization (e.g. without using Cholesky decomposition).

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.

Acknowledgments

This work is partially supported by CHIST-ERA project IPALM (Academy of Finland decision 326304).

References

[1] 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.

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

[3] L. Guilamo, J. Kuffner, K. Nishiwaki, S. Kagami, Manipulability optimization for trajectory generation, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2006, pp. 2017–2022.

[4] N. Vahrenkamp, T. Asfour, G. Metta, G. Sandini, R. Dillmann, Manipulability analysis, in: Proceedings of the 12th IEEE/RAS International Conference on Humanoid Robots, Humanoids, 2012, pp. 568–573.

[5] I. Lee, J.-H. Oh, Humanoid posture selection for reaching motion and a cooperative balancing controller, J. Intell. Robot. Syst. 81 (3–4) (2016) 301–316.

[6] 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.

[7] R. Ikeura, H. Inooka, Variable impedance control of a robot for cooperation with a human, in: IEEE International Conference on Robotics and Automation, Vol. 3, Nagoya, Japan, 1995, pp. 3097–3102.

[8] T. Tsumugiwa, R. Yokogawa, K. Hara, Variable impedance control based on estimation of human arm stiffness for human-robot cooperative calligraphic task, in: IEEE International Conference on Robotics and Automation, Vol. 1, Washington, DC, USA, 2002, pp. 644–650.

[9] K. Kronander, A. Billard, Online learning of varying stiffness through physical human-robot interaction, in: IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 2012, pp. 1842–1849.

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

[11] J. Silvério, Y. Huang, F.J. Abu-Dakka, L. Rozo, D.G. Caldwell, Uncertainty-aware imitation learning using kernelized movement primitives, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Macau, China, 2019, pp. 90–97.

[12] 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.

[13] 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.

[14] S. Schaal, Is imitation learning the route to humanoid robots? Trends Cogn. Sci. 3 (6) (1999) 233–242.

[15] 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.

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

[17] Y. Huang, L. Rozo, J. Silvério, D.G. Caldwell, Kernelized movement primitives, Int. J. Robot. Res. 38 (7) (2019) 833–852.

[18] 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.

[19] A. Ude, B. Nemec, T. Petric, J. Morimoto, Orientation in Cartesian Space Dynamic Movement Primitives, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2014, pp. 2997–3004.

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

[21] Z. Wang, B.C. Vemuri, Y. Chen, T.H. Mareci, A constrained variational principle for direct estimation and smoothing of the diffusion tensor field from complex DWI, IEEE Trans. Med. Imaging 23 (8) (2004) 930–939.

[22] H. Zhu, Y. Chen, J.G. Ibrahim, Y. Li, C. Hall, W. Lin, Intrinsic regression models for positive-definite matrices with applications to diffusion tensor imaging, J. Amer. Statist. Assoc. 104 (487) (2009) 1203–1212.

[23] K. Kronander, A. Billard, Learning compliant manipulation through kinesthetic and tactile human-robot interaction, IEEE Trans. Haptics 7 (3) (2014) 367–380.

[24] N. Hogan, Impedance control: An approach to manipulation. Part I–Theory, Part II–Implementation, and Part III–Applications, ASME Trans. J. Dyn. Syst. Measur. Control B 107 (1985) 1–7, 8–16, 17–24.

[25] F.J. Abu-Dakka, M. Saveriano, Variable impedance control and learning – a review, Front. Robot. AI 7 (2020) 177.

[26] S. Calinon, A tutorial on task-parameterized movement learning and retrieval, Intell. Serv. Robot. 9 (1) (2016) 1–29.

[27] S. Chiu, Control of redundant manipulators for task compatibility, in: Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 4, 1987, pp. 1718–1724.

[28] P. Chiacchio, Exploiting redundancy in minimum-time path following robot control, in: American Control Conference, 1990, pp. 2313–2318.

[29] S. Lee, Dual redundant arm configuration optimization with task-oriented dual arm manipulability, IEEE Trans. Robot. Autom. 5 (1) (1989) 78–97.

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

[31] Y. Huang, L. Rozo, J. Silvério, D.G. Caldwell, Non-parametric imitation learning of robot motor skills, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2019, pp. 5266–5272.

[32] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, M. Harandi, Kernel methods on the Riemannian manifold of symmetric positive definite matrices, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 73–80.

[33] A. Ajoudani, Transferring Human Impedance Regulation Skills to Robots, Springer, 2016.

[34] L. Peternel, T. Petrič, J. Babič, Robotic assembly solution by human-in-the-loop teaching method based on real-time stiffness modulation, Auton. Robots 42 (1) (2018) 1–17.

[35] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, Y. Sheikh, OpenPose: Realtime multi-person 2D pose estimation using part affinity fields, IEEE Trans. Pattern Anal. Mach. Intell. 43 (1) (2021) 172–186.

Fares J. Abu-Dakka received his B.Sc. degree in mechanical engineering from Birzeit University, Palestine, in 2003, and his M.Sc. and Ph.D. degrees in robotics motion planning from the Polytechnic University of Valencia, Spain, in 2006 and 2011, respectively. Currently, he is a senior researcher at Intelligent Robotics Group at the Department of Electrical Engineering and Automation (EEA), Aalto University, Finland. Before that he was researching at ADVR, Istituto Italiano di Tecnologia (IIT). Between 2013 and 2016 he was holding a visiting professor position at the Department of Systems Engineering and Automation of the Carlos III University of Madrid, Spain. His research activities include robot control and learning, human–robot interaction, impedance control, and robot motion planning. Webpage: https://sites.google.com/view/abudakka/.

Yanlong Huang received the B.Sc. degree in automatic control, the M.Sc. degree in control theory and control engineering from the Nanjing University of Science and Technology, Nanjing, China, in 2008 and 2010, respectively, and the Ph.D. degree in robotics from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2013.

He is currently a University Academic Fellow with the school of computing, University of Leeds, Leeds, U.K. From 2013 to 2019, he carried out his research as a Postdoctoral Researcher with the Max-Planck Institute for Intelligent Systems and Italian Institute of Technology. His interests include imitation learning, optimal control, reinforcement learning, motion planning and their applications to robotic systems.

João Silvério is a postdoctoral researcher at the Idiap Research Institute since July 2019. He received his M.Sc in Electrical and Computer Engineering (2011) from Instituto Superior Técnico (Lisbon, Portugal) and Ph.D in Robotics (2017) from the University of Genoa (Genoa, Italy) and the Italian Institute of Technology, where he was also a postdoctoral researcher until May 2019. He is interested in machine learning for robotics, particularly imitation learning and control. Webpage: http://joaosilverio.eu.

Ville Kyrki received his M.Sc. in 1999 and Ph.D. in 2002, both from Lappeenranta University of Technology (Finland). Since 2012 he is an Associate Professor (tenured in 2016) and leader of the Intelligent Robotics group at AU/EEA. Previously he was a Professor in computer science with specialization in intelligent robot systems at Lappeenranta University of Technology (Finland). He has been a PI in two FP7 projects GRASP and RECONFIG. He is the coordinator of national Strategic Research Council project ROSE and has lead successfully several national research projects financed by Academy of Finland and Finnish Funding Agency for Technology and Innovation. He is an Associate Editor of IEEE Transactions on Robotics and was the publicity chair for ICRA 2016. His research interests lie in the intersection of robotic perception, learning and manipulation.