An Approach for Imitation Learning on Riemannian Manifolds

Martijn J.A. Zeestraten[1], Ioannis Havoutis[2,3], João Silvério[1], Sylvain Calinon[2,1], Darwin. G. Caldwell[1]

***Abstract*—In imitation learning, multivariate Gaussians are widely used to encode robot behaviors. Such approaches do not provide the ability to properly represent end-effector orientation, as the distance metric in the space of orientations is not Euclidean. In this work we present an extension of common imitation learning techniques to Riemannian manifolds. This generalization enables the encoding of joint distributions that include the robot pose. We show that Gaussian conditioning, Gaussian product and nonlinear regression can be achieved with this representation. The proposed approach is illustrated with examples on a 2-dimensional sphere, with an example of regression between two robot end-effector poses, as well as an extension of Task-Parameterized Gaussian Mixture Model (TP-GMM) and Gaussian Mixture Regression (GMR) to Riemannian manifolds.**

***Index Terms*—Learning and Adaptive Systems, Probability and Statistical Methods**

I. Introduction

The Gaussian is the most common distribution for the analysis of continuous data streams. Its popularity can be explained on both theoretical and practical grounds. The Gaussian is the *maximum entropy* distribution for data defined in Euclidean spaces: it makes the least amount of assumptions about the distribution of a dataset given its first two moments [1]. In addition, operations such as marginalization, conditioning, linear transformation and multiplication, all result in a Gaussian.

In imitation learning, multivariate Gaussians are widely used to encode robot motion [2]. Generally, one encodes a joint distribution over the motion variables (e.g. time and pose), and uses statistical inference to estimate an output variable (e.g. desired pose) given an input variable (e.g. time). The variance and correlation encoded in the Gaussian allow for interpolation and, to some extent, extrapolation of the original demonstration data. Regression with a mixture of Gaussians is fast compared to data-driven approaches, because it does not depend on the original training data. Multiple models relying on Gaussian properties have been proposed, including distributions over movement trajectories [3], and the encoding of movement from multiple perspectives [4].

Probabilistic encoding using Gaussians is restricted to variables that are defined in the Euclidean space. This typically excludes the use of end-effector orientation. Although one can approximate orientation locally in the Euclidean space [5], this approach becomes inaccurate when there is large variance in the orientation data.

The most compact and complete representation for orientation is the *unit quaternion*[1]. Quaternions can be represented as elements of the 3-sphere, a 3 dimensional Riemannian manifold. Riemannian manifolds allow various notions such as length, angles, areas, curvature or divergence to be computed, which is convenient in many applications including statistics, optimization and metric learning [6]–[9].

In this work we present a generalization of common probabilistic imitation learning techniques to Riemannian manifolds. Our contributions are twofold: (i) We show how to derive Gaussian conditioning and Gaussian product through likelihood maximization. The derivation demonstrates that *parallel transportation* of the covariance matrices is essential for Gaussian conditioning and Gaussian product. This aspect was not considered in previous generalizations of Gaussian conditioning to Riemannian manifolds [10], [11]; (ii) We show how the elementary operations of Gaussian conditioning and Gaussian product can be used to extend Task-Parameterized Gaussian Mixture Model (TP-GMM) and Gaussian Mixture Regression (GMR) to Riemannian manifolds.

This paper is organized as follows: Section II introduces Riemannian manifolds and statistics. The proposed methods are detailed in Section III, evaluated in Section IV, and discussed in relation to previous work in Section V. Section VI concludes the paper. This paper is accompanied by a video. Source code related to the work presented is available through http://www.idiap.ch/software/pbdlib/.

II. Preliminaries

Our objective is to extend common methods for imitation learning from Euclidean space to Riemannian manifolds. Unlike the Euclidean space, the Riemannian Manifold is not a vector space where sum and scalar multiplication are defined. Therefore, we cannot directly apply Euclidean methods to data defined on a Riemannian manifold. However, these methods can be applied in the Euclidean tangent spaces of the manifold, which provide a way to indirectly perform computation on the manifold. In this section we introduce the notions required to enable such indirect computations to generalize methods for imitation learning to Riemannian manifolds. We first introduce

Manuscript received: September, 10, 2016; Revised December, 8, 2016; Accepted January, 3, 2017.

This paper was recommended for publication by Editor Dongheui Lee upon evaluation of the Associate Editor and Reviewers' comments. The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Union's 7th Framework Programme FP7/2007-2013/ under REA grant agreement number 608022. It is also supported by the DexROV Project through the EC Horizon 2020 programme (Grant #635491) and by the Swiss National Science Foundation through the I-DRESS project (CHIST-ERA).

[1] Department of Advanced Robotics, Istituto Italiano di Tecnologia, Via Morego 30, 16163 Genova, Italy. name.surname@iit.it

[2] Idiap Research Institute, Rue Marconi 19, 1920 Martigny, Switzerland. name.surname@idiap.ch

[3] Oxford Robotics Institute, Department of Engineering Science, University of Oxford, United Kingdom. ihavoutis@robots.ox.ac.uk

Digital Object Identifier (DOI): see top of this page.

[1]A unit quaternion is a quaternion with unit norm. From here on, we will just use *quaternion* to refer to the *unit quaternion*.

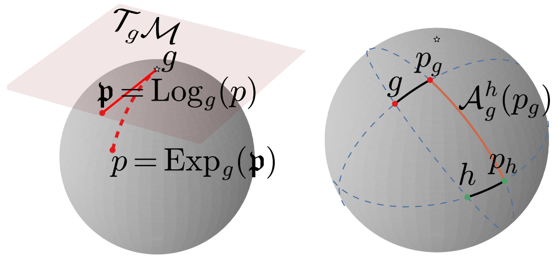

$\mathcal{T}_g\mathcal{M}$
g
$\mathfrak{p} = \mathrm{Log}_g(p)$
$p = \mathrm{Exp}_g(\mathfrak{p})$
p_g
g
$\mathcal{A}_g^h(p_g)$
h
p_h
(a)
(b)

Fig. 1: Manifold mappings and action function for $\mathcal{S}^2$. (a) The exponential and the logarithmic map provide local one-to-one mappings between the manifold and its tangent space at point $\boldsymbol{g}$. (b) An action $\mathcal{A}_{\boldsymbol{g}}^{\boldsymbol{h}}(\boldsymbol{p_g})$ maps $\boldsymbol{p_g}$ (a point defined relative to $\boldsymbol{g}$) to $\boldsymbol{p_h}$ by moving it along a geodesic (dotted lines) until it reaches a point such that the distance between $\boldsymbol{p_h}$ and $\boldsymbol{h}$ equals the distance between $\boldsymbol{p_g}$ and $\boldsymbol{g}$ (both distances visualized by black lines).

Riemannian manifolds in Section II-A, and then discuss the 'Riemannian Gaussian' in Section II-B.

The functions that will be introduced in this section are manifold specific. Table I provides an overview of their implementation for the manifolds considered in this work.

A. Riemannian Manifolds

A manifold $\mathcal{M}$ is a d-dimensional smooth space, for which each point $\boldsymbol{p} \in \mathcal{M}$ has a local bijective mapping with an open subset $\Omega \in \mathbb{R}^d$. This mapping is called a *(coordinate) chart*. Furthermore, there exists a tangent space $\mathcal{T}_{\boldsymbol{p}}\mathcal{M}$ of velocity vectors at each point $\boldsymbol{p} \in \mathcal{M}$. We indicate elements of the manifold in bold and elements on the tangent space in fraktur typeface, i.e. $\boldsymbol{p} \in \mathcal{M}$ and $\mathfrak{g} \in \mathcal{T}_{\boldsymbol{p}}\mathcal{M}$.

A manifold with a Riemannian metric—a positive definite inner product defined on the tangent space $\mathcal{T}_{\boldsymbol{p}}\mathcal{M}$—is called a Riemannian manifold. The metric introduces the notion of (minimum) distance: a notion that is essential in the definition of a Gaussian-like distribution. In Euclidean spaces, minimum distance paths lie on straight lines. Similarly, in Riemannian manifolds minimum distance paths lie on curves called *geodesics*: the generalization of straight lines.

The *exponential map* $\mathrm{Exp}_{\boldsymbol{g}}(\cdot) : \mathcal{T}_{\boldsymbol{g}}\mathcal{M} \to \mathcal{M}$ is a distance preserving map between the tangent space and the manifold. $\mathrm{Exp}_{\boldsymbol{g}}(\mathfrak{p})$ maps $\mathfrak{p}$ to $\boldsymbol{p}$ in such a way that $\boldsymbol{p}$ lies on the *geodesic* through $\boldsymbol{g}$ with direction $\mathfrak{p}$, and the distance between $\boldsymbol{g}$ and $\boldsymbol{p}$ is $\|\mathfrak{p}\| = \langle \mathfrak{p}, \mathfrak{p} \rangle_{\boldsymbol{g}}$, see Fig. 1a. The inverse mapping is called the *logarithmic map*, and exists if there is only one geodesic through $\boldsymbol{g}$ and $\boldsymbol{p}$. The distance preserving maps between the linear tangent space and the manifold allow to perform computations on the manifold indirectly.

In general, one exponential and logarithmic map is required for each tangent space. For homogeneous manifolds, however, their function can be moved from the origin $\boldsymbol{e}$ to other points on the manifold as follows

$$\mathrm{Exp}_{\boldsymbol{g}}(\mathfrak{p}_{\boldsymbol{g}}) = \mathcal{A}_{\boldsymbol{e}}^{\boldsymbol{g}}\left(\mathrm{Exp}_{\boldsymbol{e}}(\mathfrak{p}_{\boldsymbol{g}})\right), \tag{1}$$

$$\mathrm{Log}_{\boldsymbol{g}}(\boldsymbol{p}) = \mathrm{Log}_{\boldsymbol{e}}\left(\mathcal{A}_{\boldsymbol{g}}^{\boldsymbol{e}}(\boldsymbol{p})\right), \tag{2}$$

where $\mathcal{A}_{\boldsymbol{g}}^{\boldsymbol{h}}(\boldsymbol{p_g})$ is called the action function. It maps a point $\boldsymbol{p_g}$ along a geodesic to $\boldsymbol{p_h}$, in such a way that the distance between $\boldsymbol{p_g}$ and $\boldsymbol{g}$ equals the distance between $\boldsymbol{p_h}$ and $\boldsymbol{h}$ (see Fig. 1b).

Action functions remove the need to compute a specific exponential and logarithmic map for each point in the manifold at the cost of imposing a specific alignment of the tangent bases. Although this does not compromise the functions defined by (1) and (2), one must consider it while moving vectors from one tangent space to another.

Parallel transport moves vectors between tangent spaces while keeping them parallel to the geodesic that connects their bases. To achieve parallel transport between any two points on the manifold, we need to compensate for the relative rotation between $\mathcal{T}_{\boldsymbol{g}}\mathcal{M}$ and $\mathcal{T}_{\boldsymbol{h}}\mathcal{M}$. For d-spheres this rotation is given by

$$\boldsymbol{R}_{\|_{\boldsymbol{g}}}^{\boldsymbol{h}} = \boldsymbol{I}_{d+1} - \sin(m)\boldsymbol{g}\boldsymbol{u}^\top + (\cos(m)-1)\boldsymbol{u}\boldsymbol{u}^\top, \tag{3}$$

where $\boldsymbol{u} = [\mathfrak{v}^\top, 0]^\top$ gives the direction of transportation. It is constructed from $\boldsymbol{h}$ by mapping it into $\mathcal{T}_{\boldsymbol{g}}\mathcal{M}$, normalizing it, and finally rotating it to $\boldsymbol{g}$; i.e. $\mathfrak{v} = \boldsymbol{R}_{\boldsymbol{e}}^{\boldsymbol{g}} \mathrm{Log}_{\boldsymbol{g}}(\boldsymbol{h}) / m$ with $m = || \mathrm{Log}_{\boldsymbol{g}}(\boldsymbol{h}) ||$ the angle of transportation (See [12], Ch. 8). Notice that (3) is defined in the manifold ambient space $\mathbb{R}^{d+1}$, while we have defined our tangent spaces in $\mathbb{R}^d$. To achieve parallel transport between $\mathcal{T}_{\boldsymbol{g}}\mathcal{M}$ and $\mathcal{T}_{\boldsymbol{h}}\mathcal{M}$, we define the parallel action

$$\boldsymbol{\mathcal{A}}_{\|_{\boldsymbol{g}}}^{\boldsymbol{h}}(\mathfrak{p}) = \boldsymbol{B}^\top \boldsymbol{R}_{\boldsymbol{h}}^{\boldsymbol{e}} \boldsymbol{R}_{\|_{\boldsymbol{g}}}^{\boldsymbol{h}} \boldsymbol{R}_{\boldsymbol{e}}^{\boldsymbol{g}} \boldsymbol{B} \mathfrak{p}, \tag{4}$$

where $\boldsymbol{B}$ contains the direction of the bases at the origin (see Fig. 2). For the manifolds $\mathcal{S}^2$ and $\mathcal{S}^3$ we use

$$\boldsymbol{B}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}^\top, \text{ and } \boldsymbol{B}_3 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^\top. \tag{5}$$

Furthermore, $\boldsymbol{R}_{\boldsymbol{e}}^{\boldsymbol{g}}$ and $\boldsymbol{R}_{\boldsymbol{h}}^{\boldsymbol{e}}$ represent rotations between $\boldsymbol{e}$ and $\boldsymbol{g}$, and $\boldsymbol{h}$ and $\boldsymbol{e}$, respectively. Note that no information is lost through the projection $\boldsymbol{B}$, which can be understood by realizing that the parallel action is invertible.

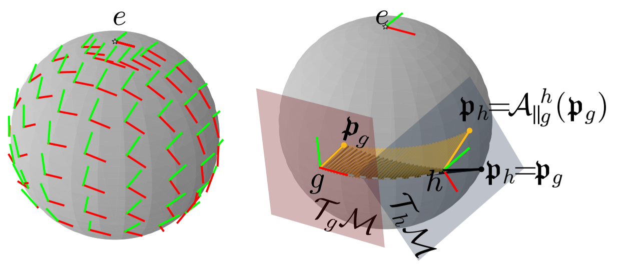
Finally, we note that the Cartesian product of two Riemannian manifolds is again a Riemannian manifold. This property allows us to define joint distributions on any combination of Riemannian manifolds. e.g., a robot pose is represented by the Cartesian product of a 3 dimensional Euclidean space and a hypersphere, i.e. $\boldsymbol{p} \in \mathbb{R}^3 \times \mathcal{S}^3$. The corresponding Exp(), Log(), $\mathcal{A}$(), and parallel transport of the Cartesian product are obtained by concatenating the individual functions, e.g.

$$\mathrm{Log}_{\left[\begin{smallmatrix} \boldsymbol{e}_a \\ \boldsymbol{e}_b \end{smallmatrix}\right]}\left(\begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix}\right) = \begin{bmatrix} \mathrm{Log}_{\boldsymbol{e}_a}(\boldsymbol{a}) \\ \mathrm{Log}_{\boldsymbol{e}_b}(\boldsymbol{b}) \end{bmatrix}.$$

B. Riemannian Statistics

In [7], Pennec shows that the maximum entropy distribution given the first two moments—mean point and covariance—is a distribution of the exponential family. Although the exact solution is computationally impractical, it is often approximated by

$$\mathcal{N}_{\mathcal{M}}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x})^\top \boldsymbol{\Sigma}^{-1} \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x})}, \tag{6}$$



e
e
$\mathfrak{p}_g$
$\mathfrak{p}_h = \mathcal{A}_{\Vert g}^{h}(\mathfrak{p}_g)$
g
h
$\mathfrak{p}_h = \mathfrak{p}_g$
$\mathcal{T}_g\mathcal{M}$
$\mathcal{T}_h\mathcal{M}$

(a) (b)

Fig. 2: (a) Tangent space alignment with respect to $\boldsymbol{e}$. (b) Even though the tangent bases are aligned with respect to $\boldsymbol{e}$, base misalignment exists between $\mathcal{T}_{\boldsymbol{g}}\mathcal{M}$ and $\mathcal{T}_{\boldsymbol{h}}\mathcal{M}$ when one of them does not lie on a geodesic through $\boldsymbol{e}$. In such cases, parallel transport of $\mathfrak{p}$ from $\mathcal{T}_{\boldsymbol{g}}\mathcal{M}$ to $\mathcal{T}_{\boldsymbol{h}}\mathcal{M}$ requires a rotation $\boldsymbol{\mathcal{A}}_{\Vert \boldsymbol{g}}^{\boldsymbol{h}}(\mathfrak{p})$ to compensate for the misalignment of the tangent spaces.

$\mathcal{M}$:	$\mathbb{R}^d$	$\mathcal{S}^2$	$\mathcal{S}^3$
$\boldsymbol{g} \in \mathcal{M}$	$\begin{pmatrix} g_{x_1} \\ \vdots \\ g_{x_d} \end{pmatrix}$	$\begin{pmatrix} g_x \\ g_y \\ g_z \end{pmatrix}$	$\begin{pmatrix} q0 \\ \boldsymbol{q} \end{pmatrix}$
$\mathrm{Exp}_{\boldsymbol{e}}(\mathfrak{g})$	$\boldsymbol{e} + \begin{bmatrix} \mathfrak{g}_{x_1} \\ \vdots \\ \mathfrak{g}_{x_d} \end{bmatrix}$	$\begin{cases} \begin{pmatrix} \mathrm{s}(\Vert\mathfrak{g}\Vert) \frac{\mathfrak{g}}{\Vert\mathfrak{g}\Vert} \\ \mathrm{c}(\Vert\mathfrak{g}\Vert) \end{pmatrix}, & \Vert\mathfrak{g}\Vert \neq 0 \\ \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, & \Vert\mathfrak{g}\Vert = 0 \end{cases}$	$\begin{cases} \begin{pmatrix} \mathrm{c}(\Vert\mathfrak{g}\Vert) \\ \mathrm{s}(\Vert\mathfrak{g}\Vert) \frac{\mathfrak{g}}{\Vert\mathfrak{g}\Vert} \end{pmatrix}, & \Vert\mathfrak{g}\Vert \neq 0 \\ \begin{pmatrix} 1 \\ \boldsymbol{0} \end{pmatrix}, & \Vert\mathfrak{g}\Vert = 0 \end{cases}$
$\mathrm{Log}_{\boldsymbol{e}}(\boldsymbol{g})$	$\begin{bmatrix} g_{x_1} \\ \vdots \\ g_{x_d} \end{bmatrix}$	$\begin{cases} \mathrm{ac}(g_z) \frac{[g_x, g_y]^\top}{\Vert [g_x, g_y]^\top \Vert}, & g_z \neq 1 \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, & g_z = 1 \end{cases}$	$\begin{cases} \mathrm{ac}_*(q_0) \frac{\boldsymbol{q}}{\Vert\boldsymbol{q}\Vert}, & q_0 \neq 1 \\ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, & q_0 = 1 \end{cases}$
$\mathcal{A}_{\boldsymbol{g}}^{\boldsymbol{h}}(\boldsymbol{p})$	$\boldsymbol{p}_g - \boldsymbol{g} + \boldsymbol{h}$	$\boldsymbol{R}_{\boldsymbol{e}}^{\boldsymbol{h}} \boldsymbol{R}_{\boldsymbol{g}}^{\boldsymbol{e}} \boldsymbol{p}_{\boldsymbol{g}}$	$\boldsymbol{g} * \boldsymbol{h} * \boldsymbol{g}^{-1} * \boldsymbol{p}_{\boldsymbol{g}}$
$\boldsymbol{\mathcal{A}}_{\Vert \boldsymbol{g}}^{\boldsymbol{h}}(\mathfrak{p})$	$\boldsymbol{I}_d \mathfrak{p}$	$\boldsymbol{B}_2^\top \boldsymbol{R}_{\boldsymbol{h}}^{\boldsymbol{e}} \boldsymbol{R}_{\Vert \boldsymbol{g}}^{\boldsymbol{h}} \boldsymbol{R}_{\boldsymbol{e}}^{\boldsymbol{g}} \boldsymbol{B}_2 \mathfrak{p}$	$\boldsymbol{B}_3^\top \boldsymbol{Q}_{\boldsymbol{h}}^\top \boldsymbol{R}_{\Vert \boldsymbol{g}}^{\boldsymbol{h}} \boldsymbol{Q}_{\boldsymbol{g}} \boldsymbol{B}_3 \mathfrak{p}$

TABLE I: Overview of the exponential and logarithmic maps, and the action and parallel transport functions for all Riemannian manifolds considered in this work. $\mathrm{s}(\cdot), \mathrm{c}(\cdot), \mathrm{ac}_*(\cdot)$ are short notations for the sine, cosine, and a modified version of the arccosine[2], respectively. The elements of $\mathcal{S}^3$ are quaternions, $*$ defines their product, and $^{-1}$ a quaternion inverse. $\boldsymbol{Q}_{\boldsymbol{g}}$ and $\boldsymbol{Q}_{\boldsymbol{h}}$ represent the quaternion matrices of $\boldsymbol{g}$ and $\boldsymbol{h}$.

where $\boldsymbol{\mu} \in \mathcal{M}$ is the Riemannian center of mass, and $\boldsymbol{\Sigma}$ the covariance defined in the tangent space $\mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}$ (see e.g. [10], [11], [13]).

The Riemannian center of mass for a set of datapoints $\{\boldsymbol{x_0}, \dots, \boldsymbol{x_N}\}$ can be estimated through a Maximum Likelihood Estimate (MLE). The corresponding log-likelihood function is

$$L(x) = c - \frac{1}{2} \sum_{i=1}^{N} h_i \, \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_i)^\top \, \boldsymbol{\Sigma}^{-1} \, \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_i) \,, \tag{7}$$

where h_i are weights that can be assigned to individual datapoints. For a single Gaussian one can set $h_i = 1$, but different weights are required when estimating the likelihood of a mixture of Gaussians.

[2]The space of unit quaternions, $\mathcal{S}^3$, provides a double covering over rotations. To ensure that the distance between two antipodal rotations is zero we define $\arccos_*(\rho) \begin{cases} \arccos(\rho) - \pi & , -1 \leq \rho < 0 \\ \arccos(\rho) & , 0 \leq \rho \leq 1 \end{cases}$.

Likelihood maximization can be achieved through Gauss-Newton optimization [7], [13]. We review its derivation because the proposed generalizations of Gaussian conditioning and Gaussian product to Riemannian manifolds rely on it.

First, the likelihood function is rearranged as

$$f(\boldsymbol{\mu}) = -\frac{1}{2} \epsilon(\boldsymbol{\mu})^\top \boldsymbol{W} \epsilon(\boldsymbol{\mu}), \tag{8}$$

where c is omitted because it has no role in the optimization, and

$$\epsilon(\boldsymbol{\mu}) = \left[\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_0)^\top, \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_1)^\top, \dots, \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_N)^\top \right]^\top, \tag{9}$$

is a stack of tangent space vectors in $\mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}$ which give the distance and direction of $\boldsymbol{x}_i$ with respect to $\boldsymbol{\mu}$. $\boldsymbol{W}$ is a weight matrix with a block diagonal structure (see Table II).

The gradient of (8) is

$$\Delta = -\left(\boldsymbol{J}^\top \boldsymbol{W} \boldsymbol{J} \right)^{-1} \boldsymbol{J}^\top \boldsymbol{W} \epsilon(\boldsymbol{x}), \tag{10}$$

with $\boldsymbol{J}$ the Jacobian of $\epsilon(\boldsymbol{x})$ with respect to the tangent basis of $\mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}$. It is a concatenation of individual Jacobians $\boldsymbol{J}_i$ corresponding to $\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_i)$ which have the simple form $\boldsymbol{J}_i = -\boldsymbol{I}_d$.

Δ provides an estimate of the optimal value mapped into the tangent space $\mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}$. The optimal value is obtained by mapping Δ onto the manifold

$$\boldsymbol{\mu} \leftarrow \mathrm{Exp}_{\boldsymbol{\mu}}(\Delta) \,. \tag{11}$$

The computation of (10) and (11) is repeated until Δ reaches a predefined convergence threshold. We observed fast convergence in our experiments for MLE, Conditioning, Product, and GMR (typically 2-5 iterations). After convergence, the covariance $\boldsymbol{\Sigma}$ is computed in $\mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}$.

Note that the presented Gauss-Newton algorithm performs optimization over a domain that is a Riemannian manifold, while standard Gauss-Newton methods consider a Euclidean domain.

III. Probabilistic Imitation Learning on Riemannian Manifolds

Many probabilistic imitation learning methods rely on some form of Gaussian Mixture Model (GMM), and inference through GMR. These include both non-autonomous (time-driven) and autonomous systems [11], [14]. The generalization capability of these methods can be improved by encoding the transferred skill from multiple perspectives [4], [15]. The elementary operations required in this framework are: MLE, Gaussian conditioning, and Gaussian product. Table II provides an overview of the parameters required to perform these operations on a Riemannian manifold using the likelihood maximization presented in Section II-B.

We present below Gaussian conditioning and Gaussian product in more details, which are then used to extend GMR and TP-GMM to Riemannian manifolds.

	$\epsilon(\boldsymbol{\mu})$	$\boldsymbol{W}$	$\boldsymbol{J}$	Δ	$\boldsymbol{\Sigma}$
MLE	$\begin{bmatrix}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n)\\ \vdots \\ \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n)\end{bmatrix}$	$\mathrm{diag}\begin{pmatrix}\boldsymbol{\Sigma}^{-1}\\ \vdots \\ \boldsymbol{\Sigma}^{-1}\end{pmatrix}$	$\begin{bmatrix}-\boldsymbol{I}\\ \vdots \\ -\boldsymbol{I}\end{bmatrix}$	$\frac{1}{\sum_i^N h_i}\sum_{n=1}^{N} h_i\,\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n)$	$\frac{1}{\sum_i^N h_i}\sum_{n=1}^{N} h_i\,\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n)\,\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n)^{\top}$
Product	$\begin{bmatrix}-\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{\mu}_1)\\ \vdots \\ -\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{\mu}_P)\end{bmatrix}$	$\mathrm{diag}\begin{pmatrix}\boldsymbol{\Sigma}_{\parallel 1}^{-1}\\ \vdots \\ \boldsymbol{\Sigma}_{\parallel P}^{-1}\end{pmatrix}$	$\begin{bmatrix}-\boldsymbol{I}\\ \vdots \\ -\boldsymbol{I}\end{bmatrix}$	$\left(\sum_{p=1}^{P}\boldsymbol{\Sigma}_{\parallel p}^{-1}\right)^{-1}\sum_{p=1}^{P}\boldsymbol{\Sigma}_{\parallel p}^{-1}\,\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{\mu}_p)$	$\left(\sum_{p=1}^{P}\boldsymbol{\Sigma}_{\parallel p}^{-1}\right)^{-1}$
Condition	$\begin{bmatrix}\mathrm{Log}_{\boldsymbol{x}_{\mathcal{I}}}(\boldsymbol{\mu}_{\mathcal{I}})\\ \mathrm{Log}_{\boldsymbol{x}_{\mathcal{O}}}(\boldsymbol{\mu}_{\mathcal{O}})\end{bmatrix}$	$\boldsymbol{\Lambda}_{\parallel}$	$\begin{bmatrix}\boldsymbol{0}\\ -\boldsymbol{I}\end{bmatrix}$	$\mathrm{Log}_{\boldsymbol{x}_{\mathcal{O}}}(\boldsymbol{\mu}_{\mathcal{O}}) + \boldsymbol{\Lambda}_{\parallel\mathcal{O}\mathcal{O}}^{-1}\boldsymbol{\Lambda}_{\parallel\mathcal{O}\mathcal{I}}^{\top}\,\mathrm{Log}_{\boldsymbol{x}_{\mathcal{I}}}(\boldsymbol{\mu}_{\mathcal{I}})$	$\boldsymbol{\Lambda}_{\parallel\mathcal{O}\mathcal{O}}^{-1}$

TABLE II: Overview of the parameters used in the likelihood maximization procedure presented in Sections II-B, III-A and III-B.

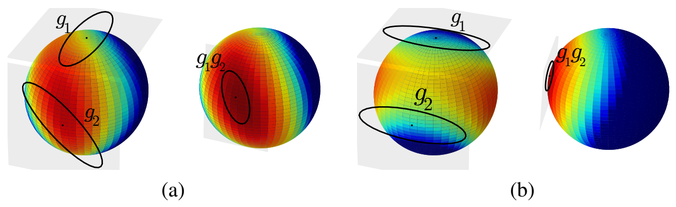


Fig. 3: Log-likelihood for the product of two Gaussians on the manifold $\mathcal{S}^2$. The Gaussians are visualized on their tangent spaces by the black ellipsoids; their center and contour represent the mean and covariance. The color of the sphere corresponds to the value of the log-likelihood (high=red, low=blue). The true log-likelihood (equation (12) with $P = 2$) is displayed on the left of each subfigure, while the log-likelihood approximated by the product is displayed on the right. The configuration of the Gaussians in (a) results in a log-likelihood with a single mode, while (b) shows the special case of multiple modes. Note the existence of a saddle point to which the gradient descent could converge.

A. Gaussian Product

In Euclidean space, the product of P Gaussians is again a Gaussian. This is not generally the case for Riemannian Gaussians. This can be understood by studying its log-likelihood function

$$L(\boldsymbol{x}) = c - \frac{1}{2}\sum_{p=1}^{P}\mathrm{Log}_{\boldsymbol{\mu}_p}(\boldsymbol{x})^T\,\boldsymbol{\Sigma}_p^{-1}\,\mathrm{Log}_{\boldsymbol{\mu}_p}(\boldsymbol{x})\,, \qquad (12)$$

where P represents the number of Gaussians to multiply, and $\boldsymbol{\mu}_p$ and $\boldsymbol{\Sigma}_p$ their parameters. The appearance of $\mathrm{Log}_{\boldsymbol{\mu}_p}(\boldsymbol{x})$ makes the log-likelihood potentially nonlinear. As a result the product of Gaussians on Riemannian manifolds is not guaranteed to be Gaussian.

Figure 3 shows comparisons on $\mathcal{S}^2$ between the true log-likelihood of a product of two Gaussians, and the log-likelihood obtained by approximating it using a single Gaussian. The neighborhood in which the approximation of the product by a single Gaussian is reasonable will vary depending on the values of $\boldsymbol{\mu}_p$ and $\boldsymbol{\Sigma}_p$. In our experiments we demonstrate that the approximation is suitable for movement regeneration in the task-parameterized framework.

Parameter estimation for the approximated Gaussian $\mathcal{N}\left(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}\right)$ can be achieved through likelihood maximization. Ignoring the constant c, we can rewrite (12) into the form (8) by defining

$$\epsilon(\tilde{\boldsymbol{\mu}}) = \left[\mathrm{Log}_{\boldsymbol{\mu}_1}(\tilde{\boldsymbol{\mu}})^{\top}, \mathrm{Log}_{\boldsymbol{\mu}_2}(\tilde{\boldsymbol{\mu}})^{\top}, \ldots, \mathrm{Log}_{\boldsymbol{\mu}_P}(\tilde{\boldsymbol{\mu}})^{\top}\right]^{\top}, \qquad (13)$$

and $\boldsymbol{W} = \mathrm{diag}\left(\boldsymbol{\Sigma}_1^{-1}, \boldsymbol{\Sigma}_2^{-1}, \ldots, \boldsymbol{\Sigma}_P^{-1}\right)$, where $\tilde{\boldsymbol{\mu}}$ is the mean of the Gaussian we are approximating. The vectors $\mathrm{Log}_{\boldsymbol{\mu}_p}(\tilde{\boldsymbol{\mu}})$ of $\epsilon(\tilde{\boldsymbol{\mu}})$ in (13) are not defined in the $\mathcal{T}_{\tilde{\boldsymbol{\mu}}}\mathcal{M}$, but in P different tangent spaces. In order to perform the likelihood maximization we need to switch the base and argument of Log() while ensuring that the original likelihood function (12) remains unchanged. This implies that the Mahalanobis distance should remain unchanged, i.e.

$$\mathrm{Log}_{\boldsymbol{\mu}_p}(\tilde{\boldsymbol{\mu}})^{\top}\,\boldsymbol{\Sigma}_p^{-1}\,\mathrm{Log}_{\boldsymbol{\mu}_p}(\tilde{\boldsymbol{\mu}}) = \mathrm{Log}_{\tilde{\boldsymbol{\mu}}}(\boldsymbol{\mu}_p)^{\top}\,\boldsymbol{\Sigma}_{\parallel p}^{-1}\,\mathrm{Log}_{\tilde{\boldsymbol{\mu}}}(\boldsymbol{\mu}_p)\,, \qquad (14)$$

where $\boldsymbol{\Sigma}_{\parallel p}$ is a modified weight matrix that ensures an equal distance measure. It is computed through parallel transportation of $\boldsymbol{\Sigma}_p$ from $\boldsymbol{\mu}_p$ to $\tilde{\boldsymbol{\mu}}$ with

$$\boldsymbol{\Sigma}_{\parallel p} = \mathcal{A}\|_{\boldsymbol{\mu}_p}^{\tilde{\boldsymbol{\mu}}}(\boldsymbol{L}_p)^{\top}\,\mathcal{A}\|_{\boldsymbol{\mu}_p}^{\tilde{\boldsymbol{\mu}}}(\boldsymbol{L}_p)\,, \qquad (15)$$

where $\boldsymbol{L}_p$ is obtained through a symmetric decomposition of the covariance matrix, i.e. $\boldsymbol{\Sigma}_p = \boldsymbol{L}_p^{\top}\boldsymbol{L}_p$. This operation transports the eigencomponents of the covariance matrix [16]. It has to be performed at each iteration of the gradient descent because it depends on the changing $\tilde{\boldsymbol{\mu}}$. For spherical manifolds, parallel transport is the linear operation (4), and (15) simplifies to $\boldsymbol{\Sigma}_{\parallel p} = \boldsymbol{R}^{\top}\boldsymbol{\Sigma}_p\boldsymbol{R}$ with $\boldsymbol{R} = \mathcal{A}\|_{\boldsymbol{\mu}_p}^{\tilde{\boldsymbol{\mu}}}(\boldsymbol{I}_d)$.

Using the transported covariances we can compute the gradient required to estimate $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\Sigma}}$. The result is presented in Table II.

The Gaussian product arises in different fields of probabilistic robotics. Generalizations of the Extended Kalman Filter [17] and the Unscented Kalman Filter [18] to Riemannian manifolds required a similar procedure.

B. Gaussian Conditioning

In Gaussian conditioning, we compute the probability $\mathcal{P}\left(\boldsymbol{x}_{\mathcal{O}}|\boldsymbol{x}_{\mathcal{I}}\right) \sim \mathcal{N}(\boldsymbol{\mu}_{\mathcal{O}|\mathcal{I}}, \boldsymbol{\Sigma}_{\mathcal{O}|\mathcal{I}})$ of a Gaussian that encodes the joint probability density of $\boldsymbol{x}_{\mathcal{O}}$ and $\boldsymbol{x}_{\mathcal{I}}$. The log-likelihood of the conditioned Gaussian is given by

$$L(\boldsymbol{x}) = c - \frac{1}{2}\,\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x})^{\top}\,\boldsymbol{\Lambda}\,\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x})\,, \qquad (16)$$

with

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_{\mathcal{I}} \\ \boldsymbol{x}_{\mathcal{O}} \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_{\mathcal{I}} \\ \boldsymbol{\mu}_{\mathcal{O}} \end{bmatrix}, \quad \boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{\Lambda}_{\mathcal{I}\mathcal{I}} & \boldsymbol{\Lambda}_{\mathcal{I}\mathcal{O}} \\ \boldsymbol{\Lambda}_{\mathcal{O}\mathcal{I}} & \boldsymbol{\Lambda}_{\mathcal{O}\mathcal{O}} \end{bmatrix}, \tag{17}$$

where the subscripts $\mathcal{O}$ and $\mathcal{I}$ indicate respectively the input and the output, and the precision matrix $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ is introduced to simplify the derivation.

Equation (16) is in the form of (8). Here we want to estimate $\boldsymbol{x}_{\mathcal{O}}$ given $\boldsymbol{x}_{\mathcal{I}}$ (i.e. $\boldsymbol{\mu}_{\mathcal{O}|\mathcal{I}}$). Similarly to the Gaussian product, we cannot directly optimize (16) because the dependent variable, $\boldsymbol{x}_{\mathcal{O}}$, is in the argument of the logarithmic map. This is again resolved by parallel transport, namely

$$\boldsymbol{\Lambda}_{\parallel} = \mathcal{A}\|_{\boldsymbol{\mu}}^{\boldsymbol{x}}(\boldsymbol{V})^{\top} \, \mathcal{A}\|_{\boldsymbol{\mu}}^{\boldsymbol{x}}(\boldsymbol{V}), \tag{18}$$

where $\boldsymbol{V}$ is obtained through a symmetric decomposition of the precision matrix: $\boldsymbol{\Lambda} = \boldsymbol{V}^{\top}\boldsymbol{V}$. Using the transformed precision matrix, the values for $\epsilon(\boldsymbol{x}_t)$, and $\boldsymbol{J}$ (both found in Table II), we can apply (11) to obtain the update rule. The covariance obtained through Gaussian conditioning is given by $\boldsymbol{\Lambda}_{\parallel}^{-1}$. Note that we maximize the likelihood only with respect to $\boldsymbol{x}_{\mathcal{O}}$, and therefore $\boldsymbol{0}$ appears in the Jacobian.

C. Gaussian Mixture Regression

Similarly to a Gaussian Mixture Model (GMM) in Euclidean space, a GMM on a Riemannian manifold is defined by a weighted sum of Gaussians

$$\mathcal{P}(\boldsymbol{x}) = \sum_{i=1}^{K} \pi_i \, \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\Sigma}_{\mathrm{i}}),$$

where π_i are the priors, with $\sum_i^K \pi_i = 1$. In imitation learning, they are used to represent nonlinear behaviors in a probabilistic manner. Parameters of the GMM can be estimated by Expectation Maximization (EM), an iterative process in which the data are given weights for each cluster (Expectation step), and in which the clusters are subsequently updated using a weighted MLE (Maximization step). We refer to [10] for a detailed description of EM for Riemannian GMMs.

A popular regression technique for Euclidean GMM is Gaussian Mixture Regression (GMR) [4]. It approximates the conditioned GMM using a single Gaussian, i.e.

$$\mathcal{P}(\boldsymbol{x}_{\mathcal{O}}|\boldsymbol{x}_{\mathcal{I}}) \approx \mathcal{N}\left(\hat{\boldsymbol{\mu}}_{\mathcal{O}}, \hat{\boldsymbol{\Sigma}}_{\mathcal{O}}\right).$$

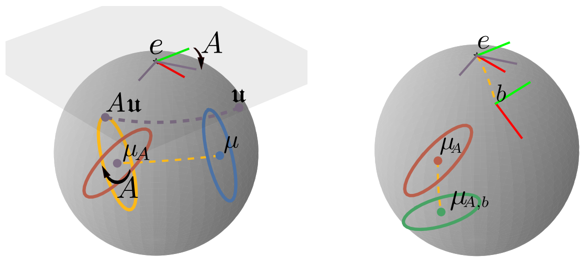
In Euclidean space, the parameters of this Gaussian are formed by a weighted sum of the expectations and covariances, namely,

$$\hat{\boldsymbol{\mu}}_{\mathcal{O}} = \sum_{i=1}^{K} h_i \, \mathbb{E}\left[\mathcal{N}(\boldsymbol{x}_{\mathcal{O}}|\boldsymbol{x}_{\mathcal{I}}; \boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\Sigma}_{\mathrm{i}})\right], \tag{19}$$

$$\hat{\boldsymbol{\Sigma}}_{\mathcal{O}} = \sum_{i=1}^{K} h_i \, \mathrm{cov}[\mathcal{N}(\boldsymbol{x}_{\mathcal{O}}|\boldsymbol{x}_{\mathcal{I}}; \boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\Sigma}_{\mathrm{i}})], \tag{20}$$

$$\text{with} \quad h_i = \frac{\mathcal{N}\left(\boldsymbol{x}_{\mathcal{I}}; \boldsymbol{\mu}_{\mathrm{i},\mathcal{I}}, \boldsymbol{\Sigma}_{\mathrm{i},\mathcal{I}\mathcal{I}}\right)}{\sum_{j=1}^{K} \mathcal{N}\left(\boldsymbol{x}_{\mathcal{I}}; \boldsymbol{\mu}_{\mathrm{j},\mathcal{I}}, \boldsymbol{\Sigma}_{\mathrm{j},\mathcal{I}\mathcal{I}}\right)}. \tag{21}$$

We cannot directly apply (19) and (20) to a GMM defined on a Riemannian manifold. First, because the computation of (19) would require a weighted sum of manifold elements—an operation that is not available on the manifold. Secondly, because directly applying (20) would incorrectly assume the alignment of the tangent spaces in which $\boldsymbol{\Sigma}_i$ are defined.



(a) Transformation $\boldsymbol{A}$ (b) Translation $\boldsymbol{b}$

Fig. 4: Application of task-parameters $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b} \in \mathcal{S}^2$ to a Gaussian defined on $\mathcal{S}^2$.

To remedy the computation of the mean, we employ the approach for MLE given in Section II-B, and compute the weighted mean iteratively in the tangent space $\mathcal{T}_{\hat{\boldsymbol{\mu}}_{\mathcal{O}}}\mathcal{M}$, namely

$$\Delta = \sum_{i=1}^{K} h_i \, \mathrm{Log}_{\hat{\boldsymbol{\mu}}_{\mathcal{O}}}(\mathbb{E}[\mathcal{N}(\boldsymbol{x}_{\mathcal{O}}|\boldsymbol{x}_{\mathcal{I}}; \boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\Sigma}_{\mathrm{i}})]), \tag{22}$$

$$\hat{\boldsymbol{\mu}}_{\mathcal{O}} \leftarrow \mathrm{Exp}_{\hat{\boldsymbol{\mu}}_{\mathcal{O}}}(\Delta). \tag{23}$$

Here, $\mathbb{E}[\mathcal{N}(\boldsymbol{x}_{\mathcal{O}}|\boldsymbol{x}_{\mathcal{I}}; \boldsymbol{\mu}_{\mathrm{i}}, \boldsymbol{\Sigma}_{\mathrm{i}})]$ refers to the conditional mean point obtained using the procedure described in Section III-B. Note that the computation of the responsibilities h_i is straightforward using the Riemannian Gaussian (6).

After convergence of the mean, the covariance is computed in the tangent space defined at $\hat{\boldsymbol{\mu}}_{\mathcal{O}}$. First, $\boldsymbol{\Sigma}_i$ are parallel transported from $\mathcal{T}_{\boldsymbol{\mu}_i}\mathcal{M}$ to $\mathcal{T}_{\hat{\boldsymbol{\mu}}_{\mathcal{O}}}\mathcal{M}$, and then summed similarly to (20) with

$$\hat{\boldsymbol{\Sigma}}_{\mathcal{O}} = \sum_{i}^{K} h_i \, \boldsymbol{\Sigma}_{\parallel i}, \quad \text{where} \tag{24}$$

$$\boldsymbol{\Sigma}_{\parallel i} = \mathcal{A}\|_{\boldsymbol{\mu}_i}^{\hat{\boldsymbol{\mu}}_{\mathcal{O}}}(\boldsymbol{L}_i)^{\top} \, \mathcal{A}\|_{\boldsymbol{\mu}_i}^{\hat{\boldsymbol{\mu}}_{\mathcal{O}}}(\boldsymbol{L}_i), \tag{25}$$

and $\boldsymbol{\Sigma}_i = \boldsymbol{L}_i \boldsymbol{L}_i^{\top}$.

D. Task-Parameterized Models

One of the challenges in imitation learning is to generalize skills to previously unseen situations, while keeping a small set of demonstrations. In task-parameterized representations [4], this challenge is tackled by considering the robot end-effector motion from different perspectives (frames of reference such as objects, robot base or landmarks). These perspectives are defined in a global frame of reference through the *task parameters* $\boldsymbol{A}$ and $\boldsymbol{b}$, representing a linear transformation and translation, respectively. In Euclidean spaces this allows data to be projected to the global frame of reference through the linear operation $\boldsymbol{A}\boldsymbol{u} + \boldsymbol{b}$.

This linear operation can also be applied to the Riemannian Gaussian, namely

$$\boldsymbol{\mu}_{\boldsymbol{A},\boldsymbol{b}} = \mathrm{Exp}_{\boldsymbol{b}}(\boldsymbol{A}\,\mathrm{Log}_{\boldsymbol{e}}(\boldsymbol{\mu})), \tag{26}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{A},\boldsymbol{b}} = (\boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^{\top})\|_{\boldsymbol{b}}^{\boldsymbol{\mu}_{\boldsymbol{A}\boldsymbol{b}}}, \tag{27}$$

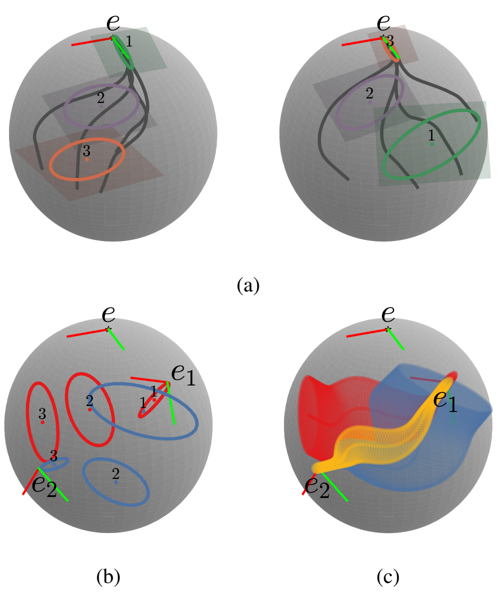


Fig. 5: Proposed approach for task parameterization (see Sec. IV).

with $(\cdot)_{\|_{b}^{\mu_{Ab}}}$ the parallel transportation of the covariance matrix from $\boldsymbol{b}$ to $\boldsymbol{\mu}_{\boldsymbol{A},\boldsymbol{b}}$.

We illustrate the individual effect of $\boldsymbol{A}$ and $\boldsymbol{b}$ in Fig. 4. The rotation $\boldsymbol{A}$ is expressed in the tangent space of $\boldsymbol{e}$. Each center $\boldsymbol{\mu}$ of the Gaussian is expressed as $\boldsymbol{u} = \mathrm{Log}_{\boldsymbol{e}}(\boldsymbol{\mu})$ in this tangent space. After applying the rotation $\boldsymbol{u}' = \boldsymbol{A}\boldsymbol{u}$, and by exploiting the property of homogeneous space in (1), the result is moved to $\boldsymbol{b}$ with $\mathrm{Exp}_{\boldsymbol{b}}(\boldsymbol{u}') = \mathcal{A}_{\boldsymbol{e}}^{\boldsymbol{b}}\left(\mathrm{Exp}_{\boldsymbol{e}}(\boldsymbol{u}')\right)$, see (26). To maintain the relative orientation between the local frame and the covariance, we parallel transport the covariance $\boldsymbol{\Sigma}_{\boldsymbol{A}}$ from the $\boldsymbol{b}$ to $\boldsymbol{\mu}_{\boldsymbol{A},\boldsymbol{b}}$ and obtain $\boldsymbol{\Sigma}_{\boldsymbol{A},\boldsymbol{b}}$. The transport compensates the tangent space misalignment caused by moving the local frame away from the origin. Fig. 4b shows the application of $\boldsymbol{b}$ to the result of Fig. 4a.

IV. Evaluation

The abilities of GMR and TP-GMM on Riemannian manifolds are briefly demonstrated using a toy example. We use four demonstrations of a point-to-point movement on $\mathcal{S}^2$, which are observed from two different frames of reference (start and end points). In each frame the spatial and temporal data, defined on the manifold $\mathcal{S}^2 \times \mathbb{R}$, are jointly encoded in a GMM ($K\!=\!3$). The estimated GMMs together with the data are visualized in Fig. 5a. To reproduce the encoded skill, we set new locations of the start and end points, and move the frames with their GMM accordingly, using the procedure described in Section III-D. Fig. 5b shows the results in red and blue. To obtain the motion in this new situation, we first perform time-based GMR to compute $\mathcal{P}\left(\boldsymbol{x}|t\right)$ separately for each GMM, which results in the two ‘tubes’ of Gaussians visualized in Fig. 5c. The combined distribution, displayed in yellow, is obtained by evaluating the Gaussian product between the two Gaussians obtained at each time step t. We can observe smooth regression results for the individual frames, as well as a smooth combination of the frames in the retrieved movement.

The Cartesian product allows us to combine a variety of Riemannian manifolds. Rich behavior can be encoded on such manifolds using relatively simple statistical models. We demonstrate this by encoding a bi-manual pouring skill with a single multivariate Gaussian, and reproduce it using the presented Gaussian conditioning.

We transfer the pouring task using 3 kinesthetic demonstrations on 6 different locations in Baxter’s workspace while recording the two end-effector poses (positions and orientations, 18 demonstrations in total). The motion consists of an adduction followed by a abduction of both arms, while holding the left hand horizontal and tilting the right hand (left and right seen from the robot’s perspective). Snapshots of a demonstration from two users (each moving one arm) are shown in Fig. 6. The demonstrated data lie on the Riemannian manifold $\mathcal{B} = \mathbb{R}^3 \times \mathcal{S}^3 \times \mathbb{R}^3 \times \mathcal{S}^3$. We encode the demonstrations in a single Gaussian defined on this manifold, i.e. we assume the recorded data to be distributed as $\boldsymbol{x} \sim \mathcal{N}_{\mathcal{B}}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$, with $\boldsymbol{x} = (\boldsymbol{x}_L, \boldsymbol{q}_L, \boldsymbol{x}_R, \boldsymbol{q}_R)$ composed of the position and quaternion of the left and right end-effectors.

Figure 8a shows the mean pose and its corresponding covariance, which is defined in the tangent space $\mathcal{T}_{\boldsymbol{\mu}}\mathcal{B}$. The x_1, x_2 and x_3 axes are displayed in red, blue and green, respectively. The horizontal constraint of the left hand resulted in low rotational variance around the x_2 and x_3 axes, which is reflected in the small covariance at the tip of the x_1 axis. The low correlation of its x_2 and x_3 axes with other variables confirms that the constraint is properly learned (Fig. 8b).

The bi-manual coordination required for the pouring task is encoded in the correlation coefficients of the covariance matrix[3] visualized in Fig. 8b. The block diagonal elements of the correlation matrix relate to the correlations within the manifolds, and the off-diagonal elements indicate the correlations between manifolds. Strong positive and negative correlations appear in the last block column/row of the correlation matrix. The strong correlation in the rotation of the right hand (lower right corner of the matrix) confirms that the main action of rotation is around the x_3-axis (blue) which causes the x_1(red) and x_2(green) axes to move in synergy. The strong correlations of the position $\boldsymbol{x}_L$, $\boldsymbol{x}_R$, and rotation $\boldsymbol{\omega}_L$ with rotation $\boldsymbol{\omega}_R$, demonstrate their importance in the task.

These correlations can be exploited to create a controller with online adaptation to a new situation. To test the responsive behavior, we compute the causal relation $\mathcal{P}\left(\boldsymbol{q}_R|\boldsymbol{x}_R, \boldsymbol{q}_L, \boldsymbol{x}_L\right)$ through the Gaussian conditioning approach presented in Sec. III-B. A typical reproduction is shown in Fig. 6, and others can be found in the video accompanying this paper. In contrast to the original demonstrations showing noisy synchronization patterns from one recording to the next, the reproduction shows a smooth coordination behavior. The orientation of the right hand correctly adapts to the position of the right hand and the pose of the left hand. This behavior generalizes outside the demonstration area.

To assess the regression quality, we perform a cross-validation on 12 out of the 18 original demonstrations (2 demonstrations on 6 different locations). The demonstra-

[3]The covariance matrix combines correlation $-1 \leq \rho_{ij} \leq 1$ among random variables X_i and X_j with deviation σ_i of random variables X_i, i.e. it has elements $\Sigma_{ij} = \rho_{ij}\sigma_i\sigma_j$. We prefer to visualize the correlation matrix, which only contains the correlation coefficients, because it highlights the coordination among variables.

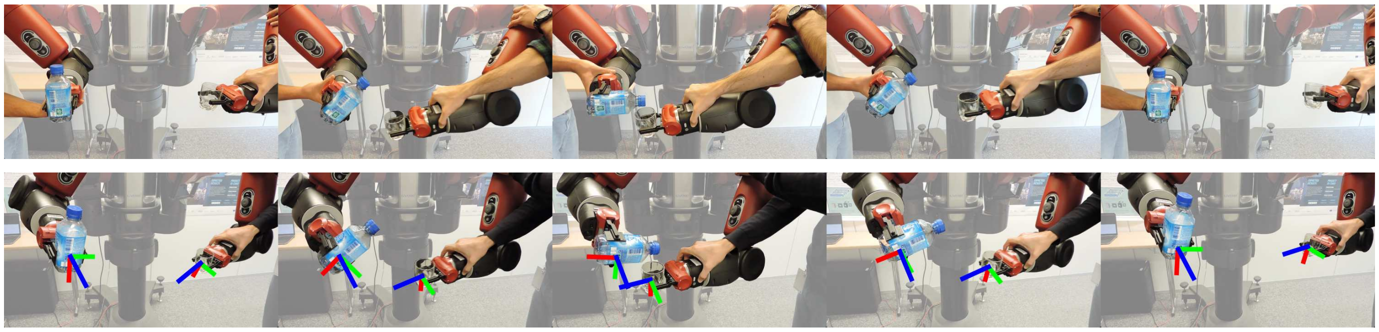


Fig. 6: Snapshot sequences of a typical demonstration (top) and reproduction (bottom) of the pouring task. The colored perpendicular lines indicate the end-effector orientations.

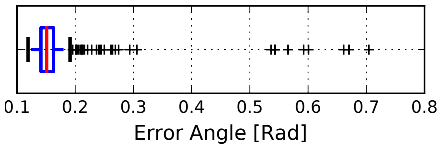


Fig. 7: Cross validation results described in Section IV.

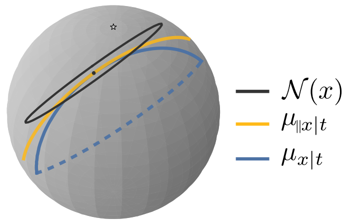
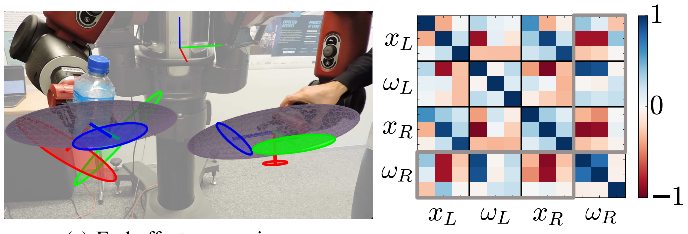


Fig. 9: Visualization of Gaussian conditioning with and without parallel transport, by computing $\mathcal{N}(\boldsymbol{x}|t)$ with $\boldsymbol{x} \in \mathcal{S}^2, t \in \mathbb{R}$. The figure shows the output manifold where the marginal distribution $\mathcal{N}(\boldsymbol{x})$ is displayed in gray. See Section V for further details.



(a) End-effector covariance
(b) End-effector correlation

Fig. 8: The encoded task projected on the left and right end-effectors. (a) Gray ellipsoids visualize one standard deviation of the spatial covariance (i.e. $\Sigma_{\boldsymbol{x}_L}$, $\Sigma_{\boldsymbol{x}_R}$), their centers are the mean end-effector positions ($\boldsymbol{\mu}_{x_L}$, $\boldsymbol{\mu}_{x_R}$). Each set of colored orthogonal lines visualize the end-effector mean orientation, its covariance is visualized by the ellipses at the end of each axis. (b) Correlation encoded within and between the different manifolds. The gray rectangles mark the correlations required for the reproduction presented in this experiment.

tions are split in training set of 8 and a validation set of 4 demonstrations, yielding $\binom{12}{8} = 495$ combinations. For each of the N data points $(\boldsymbol{x}_L, \boldsymbol{q}_L, \boldsymbol{x}_R, \boldsymbol{q}_R)$ in the validation set, we compute the average rotation error between the demonstrated right hand rotation $\boldsymbol{q}_R$, and the estimated rotation $\hat{\boldsymbol{q}}_R = \mathbb{E}[\mathcal{P}(\boldsymbol{q}_R|\boldsymbol{x}_R, \boldsymbol{q}_L, \boldsymbol{x}_L)]$, i.e. the error statistic is $\frac{1}{N}\sum_i \| \mathrm{Log}_{\hat{\boldsymbol{q}}_{R,i}}(\boldsymbol{q}_{R,i}) \|$. The results of the cross-validation are summarized in the boxplot of Figure 7. The median rotation error is about 0.15 radian, and the interquartile indicates a small variance among the results. The far outliers (errors in the range $0.5 - 0.8$ radian) correspond to combinations in which the workspace areas covered by the training set and validation set are disjoint. Such combinations make generalization harder, yielding the relatively large orientation error.

V. Related Work

The literature on statistical representation of orientation is large. We focus on the statistical encoding of orientation using quaternions, which is related in general to the field of directional statistics [19]. Here we review related work, and discuss differences with our approach.

In [20], [21], the authors use the central projection mapping between $\mathcal{S}^3$ and a Euclidean space $\mathbb{R}^3$ to project distributions on the orientation manifold. A (mixture of) Gaussian(s) is used in [20], while Gaussian Processes are used in [21]. Since the central projection is not distance preserving (i.e. points equally spaced on the manifold will not be equally spaced in the projection plane), the projection plane cannot be directly used to compute statistics on manifold elements. From a topological point of view, central projection is a chart $\phi : \mathcal{M} \to \mathbb{R}^d$; a way to numerically identify elements in a subset of the manifold. It is not the mapping between the manifold and a vector space of velocities that is typically called the tangent space. The Riemannian approach differs in two ways: (i) It relies on distance preserving mappings between the manifold and its tangent spaces to compute statistics on the manifold; and (ii) It uses a Gaussian whose center is an element of the manifold, and covariance is defined in the tangent space at that center. As a result, each Gaussian in the GMM is defined in its own tangent space, instead of a single projection space.

Both [10] and [11] follow a Riemannian approach, although not explicitly mentioned in [11]. Similar to our work, they rely on a simplified version of the maximum entropy distribution on Riemannian manifolds [7]. Independently, they present a method for Gaussian conditioning. But the proposed methods do not properly generalize the linear behavior of Gaussian conditioning to Riemannian manifolds, i.e. the means of the conditioned distribution do not lie on a single geodesic—the generalization of a straight line on Riemannian manifolds. This is illustrated in Fig. 9. Here, we computed the update Δ (given in Table II row 3, column 4) for Gaussian conditioning $\mathcal{N}(\boldsymbol{x}|t)$

with and without parallel transport, i.e. using $\mathbf{\Lambda}$ and $\mathbf{\Lambda}_{\parallel}$, respectively. Without parallel transport the regression output (solid blue line) does not lie on a geodesic, since it does not coincide with the (unique) geodesic between the outer points (dotted blue line). Using the proposed method that relies on the parallel transported precision matrix $\mathbf{\Lambda}_{\parallel}$, the conditioned means $\boldsymbol{\mu}_{\parallel\boldsymbol{x}|t}$ (displayed in yellow) follow a geodesic path on the manifold, thus generalizing the 'linear' behavior of Gaussian conditioning to Riemannian manifolds.

Furthermore, the generalization of GMR proposed in [11] relies on the weighted combination of quaternions presented in [22]. Although this method seems to be widely accepted, it skews the weighting of the quaternions. By applying this method, one computes the chordal mean instead of the required weighted mean [23].

In general, the objective in probabilistic imitation learning is to model behavior τ as the distribution $\mathcal{P}(\tau)$. In this work we take a direct approach and model a (mixture of) Gaussian(s) over the state variables. Inverse Optimal Control (IOC) uses a more indirect approach: it uses the demonstration data to uncover an underlying cost function $c_\theta(\tau)$ parameterized by θ. In [24], [25], pose information is learned by demonstration by defining cost features for specific orientation and position relationships between objects of interest and the robot end-effector. An interesting parallel exists between our 'direct' approach and recent advances in IOC where behavior is modeled as the maximum entropy distribution $p(\tau) = \frac{1}{Z_\theta}\exp(-c_\theta(\tau))$ [26]–[28]. The Gaussian used in our work can be seen as an approximation of this distribution, where the cost function is the Mahalanobis distance defined on a Riemannian manifold. The benefits of such an explicit cost structure are the ease of finding the optimal reward (we learn its parameters through EM), and the derivation of the policy (which we achieve through conditioning).

VI. Conclusion

In this work we showed how a set of probabilistic imitation learning techniques can be extended to Riemannian manifolds. We described Gaussian conditioning, Gaussian Product and parallel transport, which are the elementary tools required to extend TP-GMM and GMR to Riemannian manifolds. The selection of the Riemannian manifold approach is motivated by the potential of extending the developed approach to other Riemannian manifolds, and by the potential of drawing bridges between other robotics challenges that involve a rich variety of manifolds (including perception, planning and control problems).

References

[1] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[2] A. G. Billard, S. Calinon, and R. Dillmann, "Learning from humans," in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Secaucus, NJ, USA: Springer, 2016, ch. 74, pp. 1995–2014, 2nd Edition.

[3] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 2616–2624.

[4] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, no. 1, pp. 1–29, January 2016.

[5] J. Silvério, L. Rozo, S. Calinon, and D. G. Caldwell, "Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2015, pp. 464–470.

[6] R. Hosseini and S. Sra, "Matrix manifold optimization for Gaussian mixtures," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 910–918.

[7] X. Pennec, "Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements," *Journal of Mathematical Imaging and Vision*, vol. 25, no. 1, pp. 127–154, 2006.

[8] N. Ratliff, M. Toussaint, and S. Schaal, "Understanding the geometry of workspace obstacles in motion optimization," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2015, pp. 4202–4209.

[9] S. Hauberg, O. Freifeld, and M. J. Black, "A geometric take on metric learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 2024–2032.

[10] E. Simo-Serra, C. Torras, and F. Moreno-Noguer, "3D Human Pose Tracking Priors using Geodesic Mixture Models," *International Journal of Computer Vision (IJCV)*, pp. 1–21, August 2016.

[11] S. Kim, R. Haschke, and H. Ritter, "Gaussian mixture model for 3-dof orientations," *Robotics and Autonomous Systems*, vol. 87, pp. 28 – 37, October 2017.

[12] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ: Princeton University Press, 2008.

[13] G. Dubbelman, "Intrinsic statistical techniques for robust pose estimation," Ph.D. dissertation, University of Amsterdam, September 2011.

[14] S. M. Khansari-Zadeh and A. Billard, "Learning stable non-linear dynamical systems with Gaussian mixture models," *IEEE Trans. on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[15] S. Calinon, D. Bruno, and D. G. Caldwell, "A task-parameterized probabilistic model with minimal intervention control," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2014, pp. 3339–3344.

[16] O. Freifeld, S. Hauberg, and M. J. Black, "Model transport: Towards scalable transfer learning on manifolds," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1378 – 1385.

[17] B. B. Ready, "Filtering techniques for pose estimation with applications to unmanned air vehicles," Ph.D. dissertation, Brigham Young University-Provo, November 2012.

[18] S. Hauberg, F. Lauze, and K. S. Pedersen, "Unscented Kalman filtering on Riemannian manifolds," *Journal of mathematical imaging and vision*, vol. 46, no. 1, pp. 103–120, 2013.

[19] K. V. Mardia and P. E. Jupp, *Directional statistics*. John Wiley & Sons, 2009, vol. 494.

[20] W. Feiten, M. Lang, and S. Hirche, "Rigid Motion Estimation using Mixtures of Projected Gaussians," in *16th International Conference on Information Fusion (FUSION)*, 2013, pp. 1465–1472.

[21] M. Lang, M. Kleinsteuber, O. Dunkley, and S. Hirche, "Gaussian process dynamical models over dual quaternions," in *European Control Conference (ECC)*, 2015, pp. 2847–2852.

[22] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, "Averaging quaterions," *Journal of Guidance, Control, and Dynamics (AIAA)*, vol. 30, pp. 1193–1197, July 2007.

[23] R. I. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *International Journal of Computer Vision*, vol. 103, no. 3, pp. 267–305, July 2013.

[24] P. Englert and M. Toussaint, "Inverse KKT–learning cost functions of manipulation tasks from demonstrations," in *Proc. Intl Symp. on Robotics Research (ISRR)*, 2015.

[25] A. Doerr, N. Ratliff, J. Bohg, M. Toussaint, and S. Schaal, "Direct loss minimization inverse optimal control," in *Proceedings of Robotics: Science and Systems (RSS)*, 2015, pp. 1–9.

[26] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *AAAI Conference on Artificial Intelligence*, 2008, pp. 1433–1438.

[27] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples," in *Proceedings of the 29th International Conference on Machine Learning (ICML)*, July 2012, pp. 41–48.

[28] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *Proceedings of 33rd International Conference on Machine Learning (ICML)*, June 2016, pp. 49–58.