Gaussians on Riemannian Manifolds: Applications for Robot Learning and Adaptive Control

Sylvain Calinon

***Abstract*— This article presents an overview of robot learning and adaptive control applications that can benefit from a joint use of Riemannian geometry and probabilistic representations. The roles of Riemannian manifolds, geodesics and parallel transport in robotics are first discussed. Several forms of manifolds already employed in robotics are then presented, by also listing manifolds that have been underexploited but that have potentials in future robot learning applications. A varied range of techniques employing Gaussian distributions on Riemannian manifolds is then introduced, including clustering, regression, information fusion, planning and control problems. Two examples of applications are presented, involving the control of a prosthetic hand from surface electromyography (sEMG) data, and the teleoperation of a bimanual underwater robot. Further perspectives are finally discussed, with suggestions of promising research directions.**

I. INTRODUCTION

Data encountered in robotics are characterized by simple but varied geometries, which are sometimes underexploited in robot learning and adaptive control algorithms. Such data range from joint angles in revolving articulations [1], rigid body motions [2], [3], unit quaternions to represent orientations [4], and symmetric positive definite matrices, which can represent sensory data processed as spatial covariances [5], inertia [6], [7], stiffness/manipulability ellipsoids [8], as well as metrics used in the context of similarity measures.

Moreover, many applications require these heterogeneous data to be handled altogether. Several robotics techniques employ components from the framework of Riemannian geometry. But unfortunately, this is often implemented without providing an explicit link to this framework, which can either weaken the links to other techniques or limit the potential extensions. This can for example be the case when computing orientation errors with a logarithmic map in the context of inverse kinematics, or when interpolating between two unit quaternions with spherical linear interpolation (SLERP). This article aims at highlighting the links between existing techniques and cataloging the missing links that could be explored in further research. These points are discussed in the context of varied robot learning and adaptive control challenges.

Sylvain Calinon is with the Idiap Research Institute, Martigny, Switzerland sylvain.calinon@idiap.ch

I would like to thank Noémie Jaquier, who provided relevant suggestions for the writing and organization of the article, and who carefully proofread the manuscript.

This work was supported by the Swiss National Science Foundation (SNSF/DFG project TACT-HAND), and by the European Commission's Horizon 2020 Programme (MEMMO project, http://www.memmo-project.eu/, grant 780684, and DexROV project, http://www.dexrov.eu/, grant 635491).

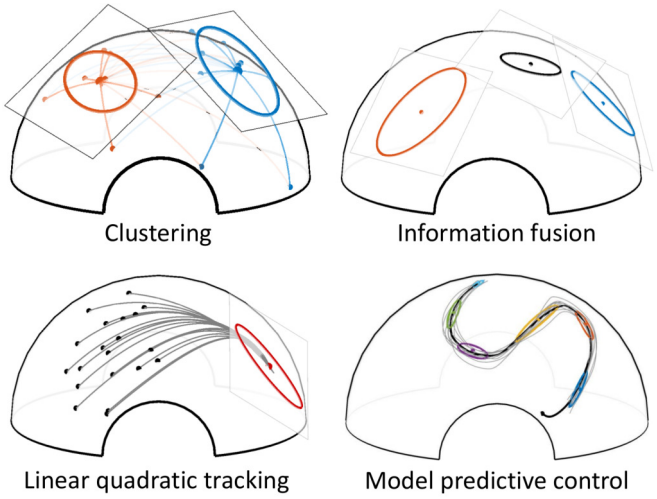

Fig. 1. Problems in robotics that can leverage the proposed Gaussian-based representation on Riemannian manifolds. Such an approach can be used to extend clustering, regression, fusion, control and planning problems to non-Euclidean data (see main text for details). In these examples, Gaussians are defined with centers on the manifolds and covariances in the tangent spaces of the centers.

Riemannian manifolds are related to a wide range of problems in machine learning and robotics. This article mainly focuses on robot learning applications based on Gaussian distributions. This includes techniques requiring uncertainty and statistical modeling to be computed on structured non-Euclidean data. The article presents an overview of existing work and further perspectives in jointly exploiting statistics and Riemannian geometry. One of the appealing use of Riemannian geometry in robotics is that it provides a principled and simple way to extend algorithms initially developed for Euclidean data to other manifolds, by efficiently taking into account prior geometric knowledge about these manifolds.

By using Riemannian manifolds, data of various forms can be treated in a unified manner, with the advantage that existing models and algorithms initially developed for Euclidean data can be extended to a wider range of data structures. It can for example be used to revisit constrained optimization problems formulated in Euclidean space, by treating them as unconstrained problems inherently taking into account the geometry of the data.

Figure 1 shows various common problems in robotics that can directly leverage Riemannian geometry. In the *clustering* example, a set of datapoints is clustered as two distributions

represented in red and blue, trained as a Gaussian mixture model, where each point is displayed in the color of the most likely cluster in which it belongs. In the *information fusion* example, the intersection of two distributions (in red and blue) results in a distribution (in black), which is the equivalent of a product of Gaussians. In the *tracking* example, a controller is computed by solving a linear quadratic tracking problem, with the goal of reaching a target point on the manifold within a desired precision matrix, represented here as a covariance matrix (inverse of precision matrix) in the tangent space of the target point. In the *model predictive control* example, a reference path is defined as a set of Gaussians that act as viapoints to pass through (i.e., within desired covariances). This Gaussian mixture model is first learned from a set of demonstrated reference paths (in gray lines). The resulting controller computes a series of control commands anticipating the next points to reach, resulting in a path on the manifold (in black line).

The problems depicted in Fig. 1 require data to be handled in a probabilistic manner. For Euclidean data, multivariate Gaussian distributions are typically considered to encode either the (co)variations of the data or the uncertainty of the estimates. This article discusses how such approaches can be extended to other manifolds by exploiting a Riemannian extension of Gaussian distributions. A practitioner perspective is adopted, with the goal of conveying the main intuitions behind the presented algorithms, sometimes at the expense of a more rigorous treatment of each topic. Didactic source codes accompany the paper, available as part of PbDlib [9], a collection of source codes for robot programming by demonstration (learning from demonstration), including various functionalities for statistical learning, dynamical systems, optimal control and Riemannian geometry. Two distinct versions are maintained, which can be used independently in Matlab (with full compatibility with GNU Octave) or in C++.

The paper is organized as follows. Section II presents an overview of Riemannian geometry in robotics. Section III presents a Gaussian-like distribution on Riemannian manifold, and shows how it can be used in mixture, regression and fusion problems. Section IV presents examples of applications, and Section V concludes the paper.

Scalars are denoted by lower case letters x, vectors by boldface lower case letters $\boldsymbol{x}$, matrices by boldface uppercase letters $\boldsymbol{X}$, where $\boldsymbol{X}^\top$ is the transpose of $\boldsymbol{X}$. Manifolds and tangent spaces are designated by calligraphic letters $\mathcal{M}$ and $\mathcal{TM}$, respectively.

II. RIEMANNIAN GEOMETRY IN ROBOTICS

A. Riemannian manifolds

A smooth d-dimensional manifold $\mathcal{M}$ is a topological space that locally behaves like the Euclidean space $\mathbb{R}^d$. A *Riemannian manifold* is a smooth and differentiable manifold equipped with a positive definite metric tensor. For each point $\boldsymbol{p}\in\mathcal{M}$, there exists a tangent space $\mathcal{T}_{\boldsymbol{p}}\mathcal{M}$ that locally linearizes the manifold. On a Riemannian manifold, the metric tensor induces a positive definite inner product on each tangent space $\mathcal{T}_{\boldsymbol{p}}\mathcal{M}$, which allows vector lengths and angles between vectors to be measured. The affine connection, computed from the metric, is a differential operator that provides, among other functionalities, a way to compute geodesics and to transport vectors on tangent spaces along any smooth curves on the manifold [10], [11]. It also fully characterizes the intrinsic curvature and torsion of the manifold. The Cartesian product of two Riemannian manifolds is also a Riemannian manifold (often called manifold bundles or manifold composites), which allows joint distributions to be constructed on any combination of Riemannian manifolds.

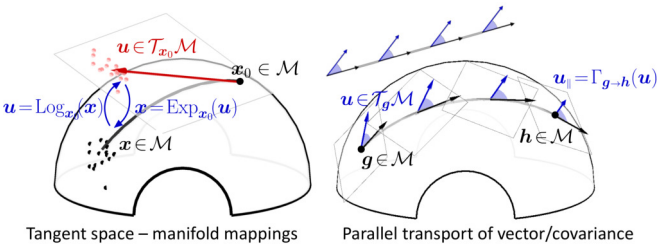

Fig. 2. Applications in robotics using Riemannian manifolds rely on two well-known principles of Riemannian geometry: exponential/logarithmic mapping (*left*) and parallel transport (*right*), which are depicted here on a $\mathcal{S}^2$ manifold embedded in $\mathbb{R}^3$. *Left:* Bidirectional mappings between tangent space and manifold. *Right:* Parallel transport of a vector along a geodesic (see main text for details).

Two basic notions of Riemannian geometry are crucial for robot learning and adaptive control applications, which are illustrated in Fig. 2 and described below.

Geodesics: The minimum length curves between two points on a Riemannian manifold are called geodesics. Similarly to straight lines in Euclidean space, the second derivative is zero everywhere along a geodesic. The exponential map $\mathrm{Exp}_{\boldsymbol{x}_0}:\mathcal{T}_{\boldsymbol{x}_0}\mathcal{M}\to\mathcal{M}$ maps a point $\boldsymbol{u}$ in the tangent space of $\boldsymbol{x}_0$ to a point $\boldsymbol{x}$ on the manifold, so that $\boldsymbol{x}$ lies on the geodesic starting at $\boldsymbol{x}_0$ in the direction $\boldsymbol{u}$. The norm of $\boldsymbol{u}$ is equal to the geodesic distance between $\boldsymbol{x}_0$ and $\boldsymbol{x}$. The inverse map is called the logarithmic map $\mathrm{Log}_{\boldsymbol{x}_0}:\mathcal{M}\to\mathcal{T}_{\boldsymbol{x}_0}\mathcal{M}$. Figure 2-*left* depicts these mapping functions.

Parallel transport: Parallel transport $\Gamma_{\boldsymbol{g}\to\boldsymbol{h}}:\mathcal{T}_{\boldsymbol{g}}\mathcal{M}\to\mathcal{T}_{\boldsymbol{h}}\mathcal{M}$ moves vectors between tangent spaces such that the inner product between two vectors in a tangent space is conserved. It employs the notion of connection, defining how to associate vectors between infinitesimally close tangent spaces. This connection allows the smooth transport of a vector from one tangent space to another by sliding it (with infinitesimal moves) along a curve.

Figure 2-*right* depicts this operation. The flat surfaces show the coordinate systems of several tangent spaces along the geodesic. The black vectors represent the directions of the geodesic in the tangent spaces. The blue vectors are transported from $\boldsymbol{g}$ to $\boldsymbol{h}$. Parallel transport allows a vector $\boldsymbol{u}$ in the tangent space of $\boldsymbol{g}$ to be transported to the tangent space of $\boldsymbol{h}$, by ensuring that the angle (i.e., inner product) between $\boldsymbol{u}$ and the direction of the geodesic (represented as black vectors) are conserved. At point $\boldsymbol{g}$, this direction is expressed as $\mathrm{Log}_{\boldsymbol{g}}(\boldsymbol{h})$. This operation is crucial to combine

information available at $\boldsymbol{g}$ with information available at $\boldsymbol{h}$, by taking into account the rotation of the coordinate systems along the geodesic (notice the rotation of the tangent spaces in Fig. 2-*right*). In Euclidean space (top-left inset), such parallel transport is simply the identity operator (a vector operation can be applied to any point without additional transformation).

By extension, a covariance matrix $\boldsymbol{\Sigma}$ can be transported with $\boldsymbol{\Sigma}_{\parallel} = \sum_{i=1}^{d} \Gamma_{\boldsymbol{g}\to\boldsymbol{h}}(\boldsymbol{v}_i)\ \Gamma_{\boldsymbol{g}\to\boldsymbol{h}}^{\top}(\boldsymbol{v}_i)$, using the eigendecomposition $\boldsymbol{\Sigma} = \sum_{i=1}^{d} \boldsymbol{v}_i\, \boldsymbol{v}_i^{\top}$. For many manifolds in robotics, this transport operation can equivalently be expressed as a linear mapping $\boldsymbol{\Sigma}_{\parallel} = \boldsymbol{A}_{\boldsymbol{g}\to\boldsymbol{h}} \boldsymbol{\Sigma}\, \boldsymbol{A}_{\boldsymbol{g}\to\boldsymbol{h}}^{\top}$ (see Supplementary Material for the computation of $\boldsymbol{A}_{\boldsymbol{g}\to\boldsymbol{h}}$).

B. Manifolds in robot applications

The most common manifolds in robotics are homogeneous, providing simple analytic expressions for exponential/logarithmic mapping and parallel transport. Some of the most important representations are listed below (see Supplementary Material for the corresponding mapping and transport operations).

Figure 3 shows four examples of Riemannian manifolds that can be employed in robot manipulation tasks. For these four manifolds, the bottom graphs depict $\mathcal{S}^2$, $\mathcal{S}^2_{++}$, $\mathcal{H}^2$ and $\mathcal{G}^{3,2}$, with a clustering problem in which the datapoints (black dots/planes) are segmented in two classes, each represented by a center (red and blue dots/planes).

The geodesics depicted in Fig. 3 show the specificities of each manifold. The sphere manifold $\mathcal{S}^d$ is characterized by constant positive curvature. The elements of $\mathcal{S}^d_{++}$ can be represented as the interior of a convex cone embedded in its tangent space Sym^d. Here, the three axes correspond to A_{11}, A_{12} and A_{22} in the SPD matrix $\left(\begin{smallmatrix} A_{11} & A_{12} \\ A_{12} & A_{22} \end{smallmatrix}\right)$. The hyperbolic manifold $\mathcal{H}^d$ is characterized by constant negative curvature. Several representations exist: $\mathcal{H}^2$ can for example be represented as the interior of a unit disk in Euclidean space, with the boundary of the disk representing infinitely remote point (Poincaré disk model, as depicted here). In this model, geodesic paths are arcs of circles intersecting the boundary perpendicularly. $\mathcal{G}^{d,p}$ is the Grassmann manifold of all p-dimensional subspaces of $\mathbb{R}^d$.

Two Lie groups widely used in robotics are also presented, namely the special orthogonal group $\mathrm{SO}(3)$ and the special Euclidean group $\mathrm{SE}(3)$. Similarities and differences between Riemannian manifolds and Lie groups are discussed in the next section. Examples of applications for the aforementioned Riemannian manifolds are presented below.

The sphere manifold $\mathcal{S}^d$ can be used in robotics to encode directions/orientations. Unit quaternions $\mathcal{S}^3$ can be used to represent endeffector (tool tip) orientations [4]. $\mathcal{S}^2$ can be used to represent unit directional vector perpendicular to surfaces (e.g., for contact planning). Articulatory joints can be represented on the torus $\mathcal{S}^1 \times \mathcal{S}^1 \times \ldots \times \mathcal{S}^1$ [12], [13]. The Kendall shape space used to encode 3D skeletal motion capture data also relies on unit spheres [14].

The special orthogonal group $\mathrm{SO}(d)$ is the group of rotations around the origin in a d-dimensional space. $\mathrm{SO}(2)$ and $\mathrm{SO}(3)$ are widely used in robotics. For example, in [15], the manifold structure of the rotation group is exploited for preintegration and uncertainty propagation in $\mathrm{SO}(3)$. This is exploited for state estimation in visual-inertial odometry with mobile robots. In [16], Kalman filtering adapted to data in $\mathrm{SO}(3)$ is used for estimating the attitude of robots that can rotate in space. The optimization problem in [17] uses sequential quadratic programming (SQP) working directly on the manifold $\mathrm{SO}(3) \times \mathbb{R}^3$.

The special Euclidean group $\mathrm{SE}(3)$ is the group of rigid body transformations. A rigid body transformation is composed of a rotation and a translation. The geometry of $\mathrm{SE}(3)$ can be used to describe the kinematics and the Jacobian of robots [2]. Therefore, it is widely used to describe robot motion and pose estimation [18]. For example, in [3], exponential maps are exploited to associate uncertainty with $\mathrm{SE}(3)$ datapoints in robot pose estimation problems.

The manifold of symmetric positive definite (SPD) matrices $\mathcal{S}^d_{++}$ can be employed in various ways in robotics. For example, human-robot collaboration applications require the use of various sensors. These sensory data can be preprocessed with sliding windows to analyze at each time step the evolution of the signals within a short time window (e.g., to analyze data flows). Often, such analysis takes the form of spatial covariances, which are SPD matrices [5]. In robot control, tracking gains can be defined in the form of SPD matrices. The use of tracking gains as full SPD matrices instead of scalars has the advantage of allowing the controller to take into account the coordination of different control variables (e.g., motor commands). For articulatory joints, these coordinations often relate to characteristic synergies in human movements. Manipulability ellipsoids are representations used to analyze and control the robot dexterity as a function of the articulatory joints configuration. This descriptor can be designed according to different task requirements, such as tracking a desired position or applying a specific force [8], [19]. Manipulator inertia matrices also belong to $\mathcal{S}^d_{++}$ and can for example be exploited in humanlike trajectory planning [13]. SPD matrices are also used in problems related to metric interpolation/extrapolation and metric learning [20]. In CHOMP (covariant Hamiltonian motion planning) [21], a precision matrix (metric tensor) is used to prefer perturbations resulting in small accelerations in the overall trajectory. In [22], Riemannian manifold policies are employed to generate natural obstacle-avoiding reaching motion through traveling along geodesics of curved spaces defined by the presence of obstacles.

Hyperbolic manifolds $\mathcal{H}^d$ are the analogues of spheres with constant negative curvature instead of constant positive curvature. They are currently underexploited in robotics, despite their interesting potential in a wide range of representations, including dynamical systems,

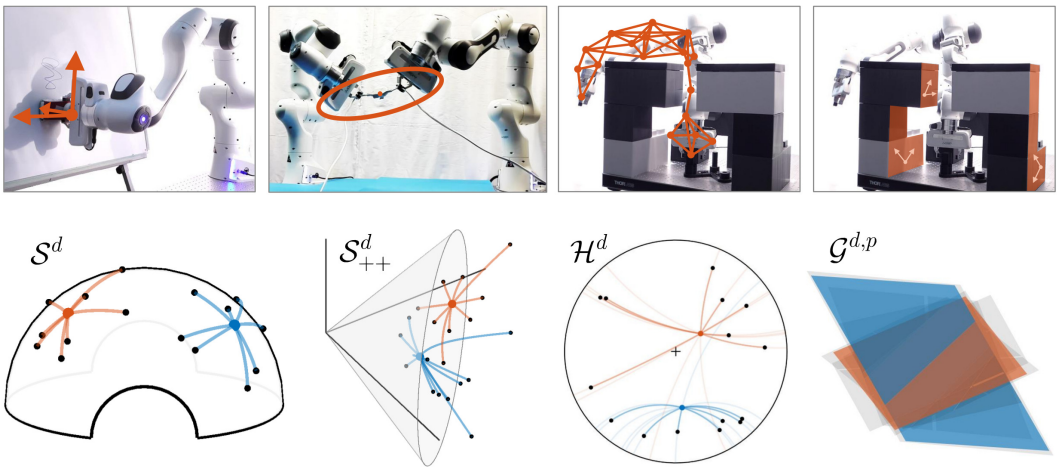

Fig. 3. Structured manifolds in robotics. $\mathcal{S}^3$ can be used to represent the orientation of robot endeffectors (unit quaternions). $\mathcal{S}^6_{++}$ can be used to represent manipulability ellipsoids (manipulability capability in translation and rotation), corresponding to a symmetric positive definite (SPD) matrix manifold. $\mathcal{H}^d$ can be used to represent trees, graphs and roadmaps. $\mathcal{G}^{d,p}$ can be used to represent subspaces (planes, nullspaces, projection operators), see main text for details.

Toeplitz/Hankel matrices or autoregressive models [23]. Hyperbolic geometry could notably be used to encode and visualize heterogeneous topology data, including graphs and trees structures, such as rapidly exploring random trees (RRT) [24], designed to efficiently search nonconvex, high-dimensional spaces in motion planning by randomly building a space-filling tree. The interesting property of hyperbolic manifolds is that the circumference of a circle grows exponentially with its radius, which means that exponentially more space is available with increasing distance. It provides a convenient representation for hierarchies, which tend to expand exponentially with depth.

Grassmannian $\mathcal{G}^{d,p}$ is the manifold of all p-dimensional subspaces of $\mathbb{R}^d$. It can for example be used to extract and cluster planar surfaces in the robot's 3D environment. This manifold is largely underrepresented in robotics, despite such structure can be used in various approaches such as system identification [25], spatiotemporal modeling of human gestures [26], or the encoding of nullspaces and projection operators in a probabilistic way.

Manifolds with nonconstant curvature are also employed in robotics, such as spaces endowed with the Fisher information metric [27], [28] or kinetic energy metric [1], [18], [12], [13]. As described in Section II-A, the curvature of a Riemannian manifold depends on the selected metric tensor. Consequently, a varying metric will result in a varying curvature. Many problems in robotics can be formulated with a smoothly varying matrix $\boldsymbol{M}$ (Riemannian metric) that measures the distance between two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ as a quadratic error term $(\boldsymbol{x}_1-\boldsymbol{x}_2)^\top \boldsymbol{M}(\boldsymbol{x}_1-\boldsymbol{x}_2)$. In this context, the Riemannian formulation has the advantage of being coordinate independent (i.e., geodesic paths are invariant to the choice of local coordinates) [1], [12], [13]. In robot dynamics problems, this is typically useful to take into account the inertia in the robot motion [1]. In policy learning problems, if the conditional density of the action given the state is Gaussian, the natural policy gradient is given by the Fisher information matrix [28], which can for example be used in deep reinforcement learning [29].

It is also relevant for deep generative models such as variational autoencoders (VAEs) and generative adversarial networks (GANs), as it provides a geometric interpretation of these models. For example, VAEs learn nonlinear data distributions through a set of latent variables and a nonlinear generator function that maps latent points into the input space. The nonlinearity of the generator implies that the latent space gives a distorted view of the input space. The latent space not only provides a low-dimensional representation of the data manifold: it can also reveal its underlying geometrical structure [30].

In neural networks such as VAEs, by using activation functions that are C^2 differentiable, a symmetric positive definite matrix $\boldsymbol{M} = \boldsymbol{J}^\top \boldsymbol{J}$ can be used as a smoothly changing inner product structure, acting as a local Mahalanobis distance measure, where $\boldsymbol{J}$ is the Jacobian characterizing the decoder function. The method yields a distance measure that can for example be used to replace linear interpolations in the latent space by geodesics. In [30], Arvanitidis *et al.* show that in the latent space learned by a VAE, distances and interpolants can significantly be improved under this metric, which in turn improves probability distributions, sampling algorithms and clustering in the latent space.

A downside of manifolds with nonconstant curvature is that the geodesic optimization problem most often corresponds to a nonconvex problem (system of ordinary differential equations) that can be computationally heavy to solve. Several research directions can be explored to address this issue. In [31], the problem is circumvented by spanning the latent space with a discrete finite graph (k-d tree data structure with edge weights based on Riemannian distances), used in conjunction with a classic A^* search algorithm.

Recent approaches in discrete differential geometry also address similar problems by extending continuous Riemannian manifolds to discrete formulations with fast computation [32]. Currently, these developments principally target computer graphics applications, but they have great potentials in robotics. It could for example provide an approach to link discrete robot planning problems such as probabilistic roadmaps (PRMs) to their continuous counterparts in Riemannian geometry.

C. Riemannian geometry and Lie theory

A Lie group is a smooth and differentiable manifold that possesses a group structure, therefore satisfying the group axioms. There are strong links between Riemannian geometry and Lie theory. In particular, some Lie groups, such as $\mathrm{SO}(3)$, can be endowed with a bi-invariant Riemannian metric, which give them the structure of a Riemannian manifold. In robotics, Lie theory is mainly exploited for applications involving $\mathrm{SO}(3)$ and $\mathrm{SE}(3)$ groups.

In the literature, distinctive vocabulary and notation are often employed, which hinder some of the links between the applications exploiting Riemannian geometry and Lie theory. Among these differences, the *Lie algebra* is the tangent space at the origin of the manifold, acting as a global reference. $\boldsymbol{u}^\wedge$ (*hat*) and $\boldsymbol{u}^\vee$ (*vee*) are used to transform elements from the Lie algebra (which can have nontrivial structures such as complex numbers or skew-symmetric matrices) to vectors in $\mathbb{R}^d$, which are easier to manipulate. They are the operations corresponding to the exponential and logarithmic maps in Riemannian geometry. In Lie theory, $\oplus$ and $\ominus$ are operators used to facilitate compositions with exponential/logarithmic mapping operations.

For further reading, an excellent introduction to Lie theory for robot applications can be found in [33].

III. GAUSSIAN DISTRIBUTIONS ON RIEMANNIAN MANIFOLDS

Several approaches have been proposed to extend Gaussian distributions in Euclidean space to Riemannian manifolds [34]. Here, we focus on a simple approach that consists of estimating the mean of the Gaussian as a centroid on the manifold (also called Karcher/Fréchet mean), and representing the dispersion of the data as a covariance expressed in the tangent space of the mean [10], [35], [4]. Distortions arise when points are too far apart from the mean, but this distortion is negligible in most robotics applications. In particular, this effect is strongly attenuated when a mixture of Gaussians is considered, as each Gaussian will be employed to model a limited region of the manifold. In the general case of a manifold $\mathcal{M}$, such a model is a distribution maximizing the entropy in the tangent space. It is defined as

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

where $\boldsymbol{x} \in \mathcal{M}$ is a point of the manifold, $\boldsymbol{\mu} \in \mathcal{M}$ is the mean of the distribution (origin of the tangent space), and $\boldsymbol{\Sigma} \in \mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}$ is the covariance defined in this tangent space.

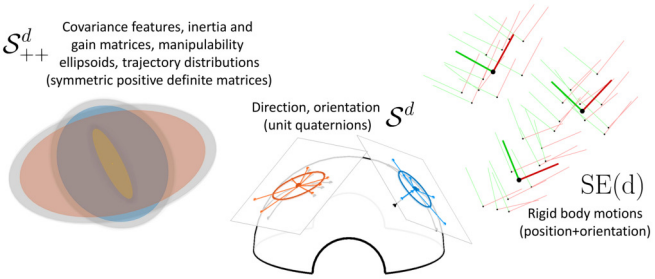

Fig. 4. Clustering on various manifolds with Gaussian mixture models.

For a set of N datapoints, this geometric mean corresponds to the minimization

$$\min_{\boldsymbol{\mu}} \sum_{n=1}^{N} \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n)^\top \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n),$$

which can be solved by a simple and fast Gauss-Newton iterative algorithm. The algorithm starts from an initial estimate on the manifold and an associated tangent space. The datapoints $\{\boldsymbol{x}_n\}_{n=1}^N$ are projected in this tangent space to compute a direction vector, which provides an updated estimate of the mean. This process is repeated by iterating

$$\boldsymbol{u} = \frac{1}{N}\sum_{n=1}^{N} \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n), \qquad \boldsymbol{\mu} \leftarrow \mathrm{Exp}_{\boldsymbol{\mu}}(\boldsymbol{u}),$$

until convergence. In practice, such an algorithm converges very fast in only a couple of iterations (typically less than 10 for the accuracy required by the applications presented here). After convergence, a covariance is computed in the tangent space as $\boldsymbol{\Sigma} = \frac{1}{N-1}\sum_{n=1}^{N} \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n)\,\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x}_n)^\top$, see Fig. 1. This distribution can for example be used in a control problem to represent a reference to track with an associated required precision (e.g., learned from a set of demonstrations). Such a learning and control problem results in the linear quadratic tracking (LQT) solution depicted in Fig. 1 and described in details in [36].

Importantly, this geometric mean can be directly extended to weighted distances, which will be exploited in the next sections for mixture modeling, fusion (product of Gaussians), regression (Gaussian conditioning) and planning problems.

A. Gaussian mixture model

Gaussian mixture model (GMM) is a ubiquitous representation in robotics, including clustering and modeling of distributions as a superposition of Gaussians, see Fig. 4. Similarly to a GMM in the Euclidean space, a GMM on a manifold $\mathcal{M}$ is defined by $p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}_{\mathcal{M}}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, with K the number of components and π_k the mixing coefficients (priors) such that $\sum_k \pi_k = 1$ and $\pi_k \geq 0$, $\forall\, k \in \{1,\dots,K\}$. The parameters of this GMM can be estimated by Expectation-Maximization (EM) [37], where the Gauss-Newton procedure presented above is performed in the M-step.

Figure 5-*top* shows that a GMM computed in a single tangent space (here, at the origin of the manifold) introduces distortions resulting in a poor modeling of the data. Figure

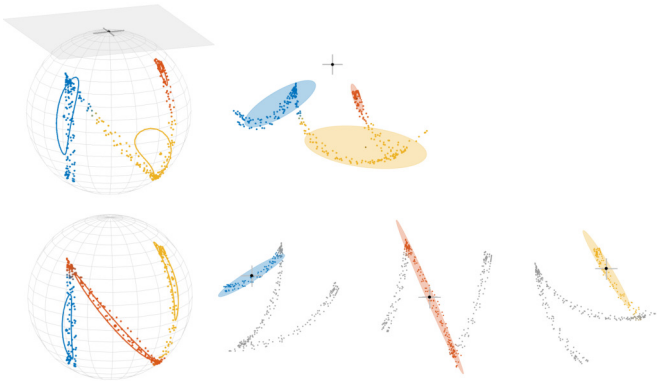

Fig. 5. Gaussian mixture model (GMM) trained with EM algorithm on $\mathcal{S}^2$ manifold. *Top row:* GMM encoding in a single tangent space (at the origin). *Bottom row:* Proposed GMM, where the covariances are computed in the tangent spaces of the means. On the left figures, the contours of the covariances are projected on the manifold. The right figures show the projections of the data into the tangent spaces considered in the computation of the GMM. We can see that representing the local dispersion of the data as covariances in the tangent spaces of the means (bottom row) results in a much better fit than representing the GMM in a single tangent space (top row).

5-*bottom* shows that the proposed representation limits the distortions by encoding the local spread of the data in covariance matrices expressed in different tangent spaces (i.e., at the centers of the Gaussians).

An example of application with links to robotics is [35], where human poses are modeled using a GMM on $\mathcal{S}^d$. Matlab examples `demo_Riemannian_Sd_GMM*.m` can be found in [9].

B. Gaussian conditioning

As detailed in [4], we consider input and output data jointly encoded as a multivariate Gaussian $\mathcal{N}_{\mathcal{M}}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ partitioned with symbols $^{\mathcal{I}}$ and $^{\mathcal{O}}$ (input and output). Given an input datapoint $\boldsymbol{x}^{\mathcal{I}}$, the conditional distribution $\boldsymbol{x}^{\mathcal{O}}|\boldsymbol{x}^{\mathcal{I}} \sim \mathcal{N}_{\mathcal{M}}(\hat{\boldsymbol{\mu}}^{\mathcal{O}}, \hat{\boldsymbol{\Sigma}}^{\mathcal{O}})$ can be locally evaluated by iterating

$$\boldsymbol{u} = \mathrm{Log}_{\hat{\boldsymbol{\mu}}^{\mathcal{O}}}(\boldsymbol{\mu}^{\mathcal{O}}) - \boldsymbol{\Sigma}_{\parallel}^{\mathcal{O}\mathcal{I}}\, \boldsymbol{\Sigma}_{\parallel}^{\mathcal{I}-1}\, \mathrm{Log}_{\boldsymbol{x}^{\mathcal{I}}}(\boldsymbol{\mu}^{\mathcal{I}}),\ \hat{\boldsymbol{\mu}}^{\mathcal{O}} \leftarrow \mathrm{Exp}_{\hat{\boldsymbol{\mu}}^{\mathcal{O}}}(\boldsymbol{u}),$$

with $\boldsymbol{\Sigma}_{\parallel}$ a covariance matrix transported from $[\boldsymbol{\mu}^{\mathcal{I}\top}, \boldsymbol{\mu}^{\mathcal{O}\top}]^\top$ to $[\boldsymbol{x}^{\mathcal{I}\top}, \hat{\boldsymbol{\mu}}^{\mathcal{O}\top}]^\top$ (see Section II for the description of parallel transport). After convergence, the covariance is computed in the tangent space as $\hat{\boldsymbol{\Sigma}}^{\mathcal{O}} = \boldsymbol{\Sigma}_{\parallel}^{\mathcal{O}} - \boldsymbol{\Sigma}_{\parallel}^{\mathcal{O}\mathcal{I}}\, \boldsymbol{\Sigma}_{\parallel}^{\mathcal{I}-1} \boldsymbol{\Sigma}_{\parallel}^{\mathcal{I}\mathcal{O}}$. Matlab examples `demo_Riemannian_Sd_GMR*.m` can be found in [9].

C. Fusion with products of Gaussians

As shown in [36], [38], the product of K Gaussians on a Riemannian manifold can be locally evaluated by iterating

$$\boldsymbol{u} = \left(\sum_{k=1}^{K} \boldsymbol{\Sigma}_{\parallel k}^{-1}\right)^{-1} \sum_{k=1}^{K} \boldsymbol{\Sigma}_{\parallel k}^{-1}\, \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{\mu}_k), \qquad \boldsymbol{\mu} \leftarrow \mathrm{Exp}_{\boldsymbol{\mu}}(\boldsymbol{u}),$$

with covariance matrix $\boldsymbol{\Sigma}_{\parallel k}$ transported from $\boldsymbol{\mu}_k$ to $\boldsymbol{\mu}$ (see Section II for the description of parallel transport). After convergence, the covariance is computed in the tangent space as $\boldsymbol{\Sigma} = \left(\sum_{k=1}^{K} \boldsymbol{\Sigma}_{\parallel k}^{-1}\right)^{-1}$.

An example of product of Gaussians on $\mathcal{S}^2$ is depicted in the top-right inset of Fig. 1. A Matlab example `demo_Riemannian_Sd_GaussProd01.m` can be found in [9].

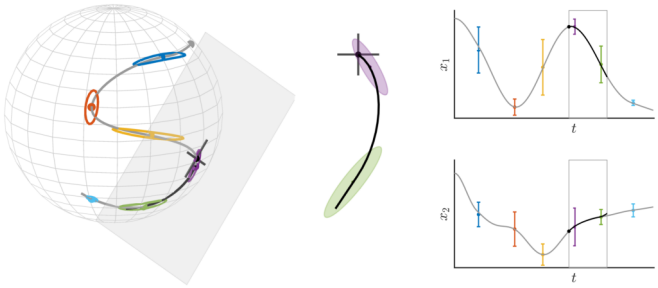

Fig. 6. Model predictive control (MPC) on a $\mathcal{S}^2$ manifold, with a set of viapoints defined by a Gaussian mixture model. *Left:* Final movement generated by MPC (in gray), superposed with the partial movement (in black) for the given time horizon (1/5 of total duration), predicted at 3/5 of the trajectory. *Center:* Visualization in the tangent space of $\boldsymbol{x}$, where only two Gaussians appear within the current time horizon. *Right:* Timeline plot showing the evolution of the first two variables of the state space, where the time horizon is depicted as a gray box. The reference to track is represented as a set of colored viapoints with desired error margins (represented as standard deviations).

D. Model predictive control

Model predictive control (MPC) is widely employed in robotics as an adaptive control strategy with anticipation capability. It consists of estimating a series of control commands $\boldsymbol{u}$ across a moving time window of size $T-1$. The problem is described here as a linear quadratic tracking (LQT) problem with velocity commands $\boldsymbol{u}_t \in \mathbb{R}^d$ and an evolution of the state $\boldsymbol{x}_t \in \mathbb{R}^d$ described by a linear system $\boldsymbol{x}_{t+1} = \boldsymbol{A}_t \boldsymbol{x}_t + \boldsymbol{B}_t \boldsymbol{u}_t$, but the approach can be generalized to other controllers. The resulting controller is

$$\begin{aligned}\hat{\boldsymbol{u}} &= \arg\min_{\boldsymbol{u}} \left\|\boldsymbol{x} - \boldsymbol{\mu}\right\|_{\boldsymbol{Q}}^2 + \left\|\boldsymbol{u}\right\|_{\boldsymbol{R}}^2 \\ &= \left(\boldsymbol{S}_{\boldsymbol{u}}{}^\top \boldsymbol{Q} \boldsymbol{S}_{\boldsymbol{u}} + \boldsymbol{R}\right)^{-1} \boldsymbol{S}_{\boldsymbol{u}}{}^\top \boldsymbol{Q}\left(\boldsymbol{\mu} - \boldsymbol{S}_{\boldsymbol{x}} \boldsymbol{x}_1\right), \end{aligned} \qquad (1)$$

with $\boldsymbol{x} = \left[\boldsymbol{x}_1^\top, \boldsymbol{x}_2^\top, \dots, \boldsymbol{x}_T^\top\right]^\top \in \mathbb{R}^{dT}$ the evolution of the state variable, $\boldsymbol{u} = \left[\boldsymbol{u}_1^\top, \boldsymbol{u}_2^\top, \dots, \boldsymbol{u}_{T-1}^\top\right]^\top \in \mathbb{R}^{d(T-1)}$ the evolution of the control variable, and d the dimension of the state space. $\boldsymbol{\mu} = \left[\boldsymbol{\mu}_1^\top, \boldsymbol{\mu}_2^\top, \dots, \boldsymbol{\mu}_T^\top\right]^\top \in \mathbb{R}^{dT}$ represents the evolution of the reference to track, $\boldsymbol{Q} = \mathrm{blockdiag}(\boldsymbol{Q}_1, \boldsymbol{Q}_2, \dots, \boldsymbol{Q}_T) \in \mathbb{R}^{dT \times dT}$ represents the evolution of the required tracking precision, and $\boldsymbol{R} = \mathrm{blockdiag}(\boldsymbol{R}_1, \boldsymbol{R}_2, \dots, \boldsymbol{R}_{T-1}) \in \mathbb{R}^{d(T-1) \times d(T-1)}$ represents the evolution of the cost on the control inputs. In (1), $\boldsymbol{S}_{\boldsymbol{u}}$ and $\boldsymbol{S}_{\boldsymbol{x}}$ are transfer matrices, see Supplementary Material for details of computation. This formulation corresponds to a basic form of MPC in Euclidean space, by considering quadratic objective functions, and linear systems with velocity commands and position states. We showed in [39] that the reference signal to be tracked can be represented by a GMM to form a stepwise trajectory.

Equation (1) is typically used to compute a series of control commands across a time window, which are reevaluated at each iteration. Thus, only the first (few) commands are used in practice. In the above formulation, the first time step of this moving time window corresponds to the current time step in which the problem is solved (see Fig. 6-*right* for an illustration of this moving time window and the computed control commands within this time window).

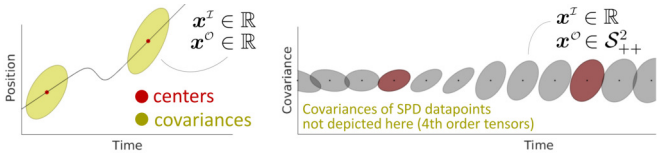

Fig. 7. Gaussian mixture regression (GMR) on SPD manifold. *Left:* Classical use of GMR to encode trajectories with time as input and position as output (both in the Euclidean space). *Right:* Extension to Riemannian manifolds with outputs on the SPD manifold. This nonlinear regression approach provides a conditional estimate of the output expressed in the form of matrix-variate Gaussians.

Such an MPC/LQT problem can be extended to Riemannian manifolds by exploiting the tangent space of the state $\boldsymbol{x}_1$ (the point that will introduce the least distortions). By extension of (1), we can solve at each iteration

$$\begin{aligned}\hat{\boldsymbol{u}} &= \arg\min_{\boldsymbol{u}} \left\|\mathrm{Log}_{\boldsymbol{x}_1}(\boldsymbol{x}) - \mathrm{Log}_{\boldsymbol{x}_1}(\boldsymbol{\mu})\right\|^2_{\boldsymbol{Q}_\parallel} + \left\|\boldsymbol{u}\right\|^2_{\boldsymbol{R}} \\ &= \left(\boldsymbol{S_u}^\top \boldsymbol{Q}_\parallel \boldsymbol{S_u} + \boldsymbol{R}\right)^{-1} \boldsymbol{S_u}^\top \boldsymbol{Q}_\parallel \,\mathrm{Log}_{\boldsymbol{x}_1}(\boldsymbol{\mu}), \end{aligned} \tag{2}$$

where the vector $\hat{\boldsymbol{u}}$ is composed of T–1 commands expressed in the tangent space of $\boldsymbol{x}_1$. $\mathrm{Log}_{\boldsymbol{x}_1}(\boldsymbol{x})$ and $\mathrm{Log}_{\boldsymbol{x}_1}(\boldsymbol{\mu})$ are vectors respectively composed of T elements $\mathrm{Log}_{\boldsymbol{x}_1}(\boldsymbol{x}_t)$ and $\mathrm{Log}_{\boldsymbol{x}_1}(\boldsymbol{\mu}_{s_t})$, with $\{s_t\}_{t=1}^T$ the sequence of Gaussian identifiers used to build the stepwise reference trajectory from the GMM. $\boldsymbol{Q}_\parallel$ is a matrix composed of the block-diagonal elements $\boldsymbol{Q}_{\parallel t} = \sum_{i=1}^d \Gamma_{\boldsymbol{\mu}_{s_t}\to\boldsymbol{x}_1}(\boldsymbol{v}_i)\ \Gamma^\top_{\boldsymbol{\mu}_{s_t}\to\boldsymbol{x}_1}(\boldsymbol{v}_i)$, using the eigendecomposition $\boldsymbol{Q}_{s_t} = \sum_{i=1}^d \boldsymbol{v}_i\,\boldsymbol{v}_i^\top$. This transport operation can equivalently be expressed as a linear mapping $\boldsymbol{Q}_{\parallel t} = \boldsymbol{M}_\parallel\,\boldsymbol{Q}_{s_t}\,\boldsymbol{M}_\parallel^\top$. In the above formulation, $\boldsymbol{R}$ is assumed to be isotropic and, thus, does not need to be transported.

The first velocity command in (2) (denoted by $\hat{\boldsymbol{u}}_1$) is then used to update the initial state for the next time window with

$$\boldsymbol{x}_1 \leftarrow \mathrm{Exp}_{\boldsymbol{x}_1}(\boldsymbol{B}_1\hat{\boldsymbol{u}}_1), \tag{3}$$

where $\boldsymbol{B}_1$ belongs to the linear system $\boldsymbol{A}_1\boldsymbol{0} + \boldsymbol{B}_1\hat{\boldsymbol{u}}_1$ at the first time step of the time window, described in the tangent space at $\boldsymbol{x}_1$, meaning that in this tangent space, the first point is at the origin $\boldsymbol{0}$.

Figure 6 shows an example on $\mathcal{S}^2$, where the computations in (2) and (3) are repeated at each time step to reproduce a movement (with the reference to track encoded as a GMM). Extensions to more elaborated forms of MPC follow a similar principle. A Matlab example `demo_Riemannian_Sd_MPC01.m` can be found in [9].

IV. EXAMPLES OF APPLICATIONS

The operations presented in the previous sections (mixture modeling, conditioning and fusion) can be combined in different ways, which is showcased here by two examples of applications.

A. Control of prosthetic hands with Gaussian mixture regression

The Gaussian conditioning approach presented in Section III-B can be extended to the Gaussian mixture model

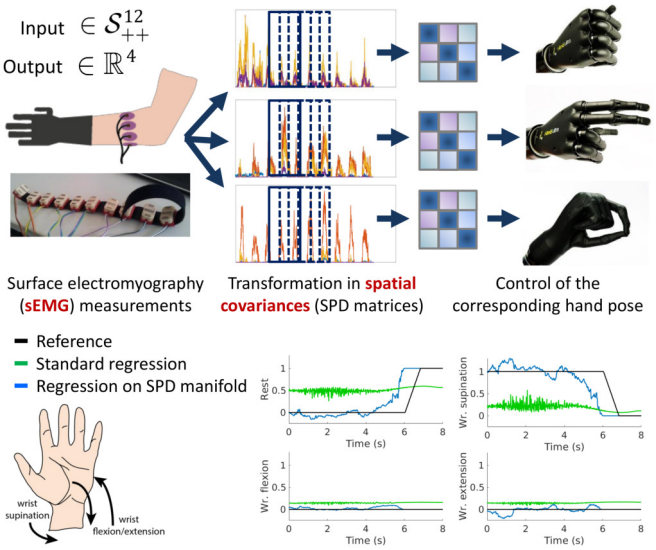

Fig. 8. GMR for the control of prosthetic hands within the TACT-HAND project. SPD signals are used as input, in the form of spatial covariances computed from sEMG sensors on the forearm of the participants. Activation signals corresponding to different hand poses are used as outputs. In this experiment (see [5] for details), taking the geometry of the data into account in GMR (bottom graphs, in blue) results in better discrimination than treating the data as if they were in a Euclidean space (bottom graphs, in green).

approach presented in Section III-A. The resulting approach is called Gaussian mixture regression (GMR), a simple nonlinear regression technique that does not model the regression function directly, but instead first models the joint probability density of input-output data in the form of a GMM [37], [39]. GMR provides a fast regression approach in which multivariate output distributions can be computed in an online manner, with a computation time independent of the number of datapoints used to train the model, by exploiting the learned joint density model. In GMR, both inputs and outputs can be multivariate, and after learning, any subset of input-output dimensions can be selected for regression. This is exploited in robotics to handle different sources of missing data, where expectations on the remaining dimensions can be computed as a multivariate distribution. These properties make GMR an attractive tool for robotics, which can be used in a wide range of problems and that can be combined fluently with other techniques [39].

Both [35] and [40] present methods for regression from a mixture of Gaussians on Riemannian manifolds, but they only partially exploit the manifold structure in Gaussian conditioning. In [35], each distribution is located on its own tangent space, with the covariances encoded separately, resulting in a block-diagonal structure in the joint distribution. In [40], a GMM is reformulated to handle the space of rotation in $\mathbb{R}^3$ by using logarithmic and exponential transformations on unit quaternions, with these operations formulated in a single tangent space (at the origin) instead of applying the transformations locally (see Fig. 5). The link to Riemannian manifolds is also not discussed.

Here, it is proposed to extend GMR to input and/or output data on SPD manifolds, see Fig. 7. As the covariance of

TABLE I

COMPARISON OF THE ROOT MEAN SQUARE ERROR (RMSE) OBTAINED BY GMR ON THE SPD MANIFOLD AND THE STANDARD EUCLIDEAN GMR FOR WRIST MOTION ESTIMATION FROM SEMG (SEE [5] FOR DETAILS). THE RESULTS ARE PRESENTED FOR THREE PARTICIPANTS.

	Rest	Wr. supination	Wr. extension	Wr. flexion
$\mathcal{S}^D_{++}$	0.29 ± 0.00	0.18 ± 0.00	0.25 ± 0.00	0.27 ± 0.00
$\mathbb{R}$	0.47 ± 0.00	0.31 ± 0.00	0.33 ± 0.00	0.33 ± 0.00
$\mathcal{S}^D_{++}$	0.32 ± 0.02	0.29 ± 0.14	0.36 ± 0.07	0.43 ± 0.13
$\mathbb{R}$	0.46 ± 0.00	0.34 ± 0.00	0.35 ± 0.00	0.35 ± 0.00
$\mathcal{S}^D_{++}$	0.36 ± 0.02	0.22 ± 0.00	0.31 ± 0.00	0.29 ± 0.00
$\mathbb{R}$	0.42 ± 0.00	0.42 ± 0.00	0.43 ± 0.00	0.43 ± 0.00

SPD datapoints is a 4th-order tensor, a method is proposed in [5] for parallel transport of high-order covariances on SPD manifolds, by exploiting the supersymmetry properties of these 4th-order tensors. As an example of application, GMR on SPD manifold is applied to predict wrist movement from spatial covariances computed from surface electromyography (sEMG) data. In this application, the input data of GMR are spatial covariances that belong to the SPD manifold. Compared to the Euclidean GMR, the GMR on SPD manifold improved the detection of wrist movement for most of the participants and proved to be efficient to detect transitions between movements, see Fig. 8 and Table I for a summary of the results. This shows the importance and benefits of considering the underlying manifold structure of the data in this application. The details of this experiment can be found in [5].

B. Underwater robot teleoperation with task-parameterized Gaussian mixture model

The fusion approach presented in Section III-C can be extended to the Gaussian mixture model approach presented in Section III-A. This is particularly useful when mixtures of Gaussians are encoded in different coordinate systems, which need to be fused at reproduction time to satisfy constraints in multiple frames of reference.

Within the DexROV project [41], this task-parameterized Gaussian mixture model (TP-GMM) approach [39], [4] is used together with the MPC approach presented in Section III-D to teleoperate an underwater robot from distance, with a teleoperator wearing an exoskeleton and visualizing a copy of the robot workspace in a virtual environment.

Figure 9 presents an overview of this application (see also [41] for a description of this teleoperation approach, and [39] for a general description of TP-GMM). Because of the long communication delays between the teleoperator and the robot, the locations of the objects or tools of interest are not the same on the teleoperator side and on the robot side. With a parameterization associated with the locations of objects and tools, we can cope with this discrepancy by adapting locally the movement representation to the position and orientation of the objects/tools, represented as coordinate systems. Figure 9 depicts an example with two coordinate systems (with models represented in orange and purple), corresponding respectively to the robot and to a valve that needs to be turned. A motion relative to the valve and to the robot is encoded as Gaussian mixture models (GMM) in the two respective coordinate systems. During teleoperation, each pair of GMMs are rotated and translated according to the current situations on the teleoperator side and on the robot side. Products of Gaussians are then computed at each side to fuse these representations.

Movement are encoded in this way with both position $\mathbb{R}^3$ and orientation $\mathcal{S}^3$ data (in Fig. 9, a representation with $\mathbb{R}^2$ and $\mathcal{S}^2$ is shown as an illustration). For position, the retrieved path in black (and the associated covariances) corresponds to a movement going from the robot to the valve (represented as red U shapes), by taking into account how these different coordinate systems are oriented. We can see that the approaching phase is perpendicular to the coordinate system to properly reach the valve. For orientation, the retrieved path shows how the orientation of the endeffector change with time. At the beginning of the motion, this orientation relates to the orientation of the robot, while at the end, the orientation of the endeffector matches the orientation of the valve. In these graphs, the purple and orange ellipsoids depict two GMMs, representing uncertain trajectories with respect to two different frames of reference (red U shapes for position data, and red points on the spheres for orientation data). The black ellipsoids represent the final trajectory and its uncertainty, obtained by fusing the trajectories of the two different frames of reference through products of Gaussians. Although the red U shapes and red points are not the same on the teleoperator side and on the robot side, the retrieved paths on the two sides can quickly adapt to these different situations. By using a Riemannian manifold framework, orientations are encoded uniquely in a representation that does not contain singularities. Such an approach is employed in this application on $\mathcal{S}^3$ to learn and retrieve the evolution of robot endeffector orientations, by adapting them to the orientation of objects or tools in the robot workspace.

This approach was successfully tested in field trials in the Mediterranean Sea offshore of Marseille, where 7 extended dives in 4 different sites (8m, 30m, 48m and 100m water depths) were performed with the underwater robot while being connected via satellite to the teleoperation center in Brussels, see [41] for a general description of the experiment.

V. FURTHER PERSPECTIVES AND CONCLUSION

This article showed that a wide range of challenges in robot learning and adaptive control can be recast as statistical modeling and information fusion on Riemannian manifolds. Such an interpretation can avoid potential misuses of algorithms in robotics that might originate from Riemannian geometry but that are treated with a limited view. One such example is to perform all computations in a single tangent space (typically, at the origin of the manifold), instead of considering the closest tangent spaces to avoid distortions. Another example concerns domain adaptation and transfer learning, which require the realignment of data to cope with nonstationarities. For example, sensory data collected by different subjects or throughout several days, which should use the Riemannian notion of *parallel transport* instead of only recentering the data [42].

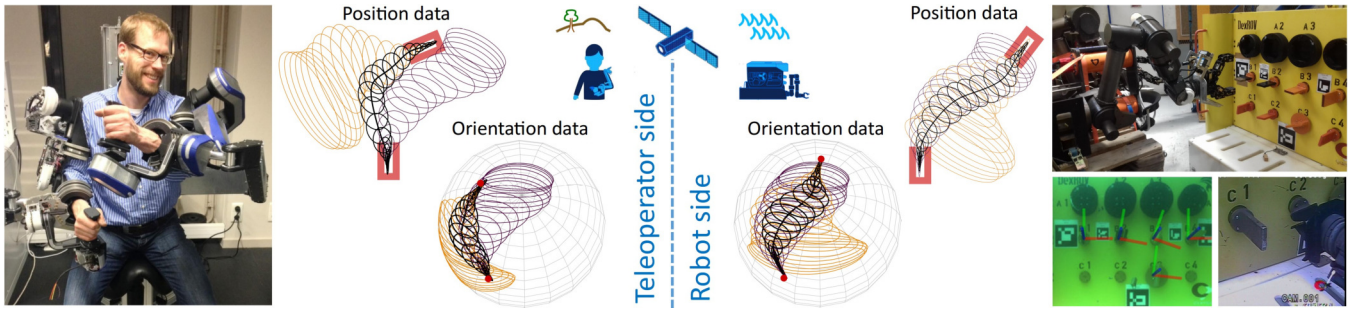

Fig. 9. Task-parameterized Gaussian mixture model (TP-GMM) extended to $\mathcal{S}^d$ manifolds within the DexROV project, see main text for details.

This article also showed that the combination of statistics and differential geometry offers many research opportunities, and can contribute to recent challenges in robotics. Further work can be organized in two categories. Firstly, the field of robotics is abundant of new techniques proposed by researchers, due to the interdisciplinary aspect and to the richness of problems it involves. The common factor in many of these developments is that they rely on some form of statistics and/or propagation of uncertainty. These models and algorithms are typically developed for standard Euclidean spaces, where an extension to Riemannian manifolds has several benefits to offer.

Secondly, some Riemannian manifolds remain largely underexploited in robotics, despite the fact that some of them are mathematically well understood and characterized by simple closed-form expressions. Grassmann manifolds seem particularly promising to handle problems in robotics with high dimensional datapoints and only few training data, where subspaces are required in the computation to keep the most essential characteristics of the data. It is also promising in problems in which hierarchies are considered (such as inverse kinematics with kinematically redundant robots), because it provides a geometric interpretation of nullspace structures. Other Riemannian manifolds such as hyperbolic manifolds also seem propitious to bring a probabilistic treatment to dynamical systems, tree-based structures, graphs, Toeplitz/Hankel matrices or autoregressive models. Finally, a wide range of metric learning problems in robotics could benefit from a Riemannian geometry treatment.

REFERENCES

[1] F. C. Park, J. E. Bobrow, and S. R. Ploen, "A Lie group formulation of robot dynamics," *The International Journal of Robotics Research*, vol. 14, no. 6, pp. 609–618, 1995.

[2] J. M. Selig, *Geometric fundamentals of robotics*. Springer, 2005.

[3] T. D. Barfoot and P. T. Furgale, "Associating uncertainty with three-dimensional poses for use in estimation problems," *IEEE Trans. on Robotics*, vol. 30, no. 3, pp. 679–693, June 2014.

[4] M. J. A. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, and D. G. Caldwell, "An approach for imitation learning on Riemannian manifolds," *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, no. 3, pp. 1240–1247, June 2017.

[5] N. Jaquier and S. Calinon, "Gaussian mixture regression on symmetric positive definite matrices manifolds: Application to wrist motion estimation with sEMG," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, September 2017, pp. 59–64.

[6] T. Lee and F. C. Park, "A geometric algorithm for robust multibody inertial parameter identification," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2455–2462, 2018.

[7] S. Traversaro, S. Brossette, A. Escande, and F. Nori, "Identification of fully physical consistent inertial parameters using optimization on manifolds," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 5446–5451.

[8] T. Yoshikawa, "Manipulability of robotic mechanisms," *Intl Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[9] "PbDlib robot programming by demonstration software library," http://www.idiap.ch/software/pbdlib/, 2020, accessed: 2020/03/10.

[10] 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.

[11] P. A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2007.

[12] S. Arimoto, M. Yoshida, M. Sekimoto, and K. Tahara, "A Riemannian-geometry approach for modeling and control of dynamics of object manipulation under constraints," *Journal of Robotics*, vol. 2009, pp. 1–16, 2009.

[13] A. Biess, T. Flash, and D. G. Liebermann, "Riemannian geometric approach to human arm dynamics, movement optimization, and invariance," *Phys. Rev. E*, vol. 83, p. 031927, Mar 2011.

[14] N. Hosni, H. Drira, F. Chaieb, and B. Ben Amor, "3D gait recognition based on functional PCA on Kendall's shape space," in *Intl Conf. on Pattern Recognition (ICPR)*, Aug 2018, pp. 2130–2135.

[15] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Trans. on Robotics*, vol. 33, no. 1, pp. 1–21, Feb 2017.

[16] J. R. Forbes and D. E. Zlotnik, "Sigma point Kalman filtering on matrix Lie groups applied to the SLAM problem," in *Geometric Science of Information (GSI)*, 2017, pp. 318–328.

[17] S. Brossette, A. Escande, and A. Kheddar, "Multicontact postures computation on manifolds," *IEEE Trans. on Robotics*, vol. 34, no. 5, pp. 1252–1265, Oct 2018.

[18] M. Zefran and V. Kumar, "Planning of smooth motions on SE(3)," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, April 1996, pp. 121–126.

[19] N. Jaquier, L. Rozo, D. G. Caldwell, and S. Calinon, "Geometry-aware manipulability learning, tracking and transfer," *arXiv:1811.11050*, pp. 1–20, 2018.

[20] 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.

[21] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant Hamiltonian optimization for motion planning," *The Intl Journal of Robotics Research*, vol. 32, no. 9–10, pp. 1164–1193, 2013.

[22] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff, "RMPflow: A computational graph for automatic motion policy generation," *arXiv:1811.07049*, pp. 1–45, 2018.

[23] E. Chevallier, F. Barbaresco, and J. Angulo, "Probability density estimation on the hyperbolic space applied to radar processing," in *Geometric Science of Information (GSI)*. Springer Intl Publishing, 2015, pp. 753–761.

[24] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[25] K. Usevich and I. Markovsky, "Optimization on a Grassmann manifold with application to system identification," *Automatica*, vol. 50, no. 6, pp. 1656–1662, 2014.

[26] R. Slama, H. Wannous, M. Daoudi, and A. Srivastava, "Accurate 3D action recognition using learning on the Grassmann manifold," *Pattern Recognition*, vol. 48, no. 2, pp. 556–567, 2015.
[27] A. D. Wilson, J. A. Schultz, and T. D. Murphey, "Trajectory synthesis for Fisher information maximization," *IEEE Trans. on Robotics*, vol. 30, no. 6, pp. 1358–1370, 2014.
[28] S. Amari, *Information Geometry and Its Applications*. Springer Japan, 2016.
[29] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in *Proc. Robotics: Science and Systems (RSS)*, Pittsburgh, PA, June 2018, pp. 1–9.
[30] G. Arvanitidis, L. K. Hansen, and S. Hauberg, "Latent space oddity: on the curvature of deep generative models," in *Proc. of the Intl Conf. on Learning Representations (ICLR)*, 2018, pp. 1–15.
[31] N. Chen, F. Ferroni, A. Klushyn, A. Paraschos, J. Bayer, and P. van der Smagt, "Fast approximate geodesics for deep generative models," in *Intl Conf. on Artificial Neural Networks (ICANN)*, 2019, pp. 554–566.
[32] N. Sharp, Y. Soliman, and K. Crane, "The vector heat method," *ACM Trans. Graph.*, vol. 38, no. 3, pp. 24:1–24:19, 2019.
[33] J. Solà, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," *arXiv:1812.01537*, 2019.
[34] S. Said, L. Bombrun, Y. Berthoumieu, and J. H. Manton, "Riemannian Gaussian distributions on the space of symmetric positive definite matrices," *IEEE Trans. on Information Theory*, vol. 63, no. 4, pp. 2153–2170, 2017.
[35] E. Simo-Serra, C. Torras, and F. Moreno-Noguer, "3D human pose tracking priors using geodesic mixture models," *International Journal of Computer Vision*, vol. 122, no. 2, pp. 388–408, 2017.
[36] M. J. A. Zeestraten, I. Havoutis, S. Calinon, and D. G. Caldwell, "Learning task-space synergies using Riemannian geometry," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, September 2017, pp. 73–78.
[37] Z. Ghahramani and M. I. Jordan, "Supervised learning from incomplete data via an EM approach," in *Advances in Neural Information Processing Systems (NIPS)*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds., vol. 6. San Francisco, CA, USA: Morgan Kaufmann Publishers, Inc., 1994, pp. 120–127.
[38] M. J. A. Zeestraten, I. Havoutis, and S. Calinon, "Programming by demonstration for shared control with an application in teleoperation," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 3, pp. 1848–1855, July 2018.
[39] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, no. 1, pp. 1–29, January 2016.
[40] S. Kim, R. Haschke, and H. Ritter, "Gaussian mixture model for 3-DoF orientations," *Robotics and Autonomous Systems*, vol. 87, pp. 28–37, 2017.
[41] A. Birk, T. Doernbach, C. A. Mueller, T. Luczynski, A. Gomez Chavez, D. Koehntopp, A. Kupcsik, S. Calinon, A. K. Tanwani, G. Antonelli, P. di Lillo, E. Simetti, G. Casalino, G. Indiveri, L. Ostuni, A. Turetta, A. Caffaz, P. Weiss, T. Gobert, B. Chemisky, J. Gancet, T. Siedel, S. Govindaraj, X. Martinez, and P. Letier, "Dexterous underwater manipulation from onshore locations: Streamlining efficiencies for remotely operated underwater vehicles," *IEEE Robotics and Automation Magazine (RAM)*, vol. 25, no. 4, pp. 24–33, 2018.
[42] O. Yair, M. Ben-Chen, and R. Talmon, "Parallel transport on the cone manifold of SPD matrices for domain adaptation," *IEEE Trans. on Signal Processing*, vol. 67, no. 7, pp. 1797–1811, Apr 2019.
[43] A. Edelman, T. A. Arias, and S. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM Journal of Matrix Anal. & Appl.*, vol. 20, no. 2, pp. 303–351, 1998.

SUPPLEMENTARY MATERIAL

$\mathcal{S}^d$ *manifold*

The exponential and logarithmic maps corresponding to the distance

$$d(\boldsymbol{x}, \boldsymbol{y}) = \arccos(\boldsymbol{x}^\top \boldsymbol{y}), \tag{4}$$

with $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}^d$ can be computed as (see also [11])

$$\boldsymbol{y} = \mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{u}) = \boldsymbol{x}\cos(\|\boldsymbol{u}\|) + \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}\sin(\|\boldsymbol{u}\|), \tag{5}$$

$$\boldsymbol{u} = \mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y}) = d(\boldsymbol{x}, \boldsymbol{y})\,\frac{\boldsymbol{y} - \boldsymbol{x}^\top \boldsymbol{y}\,\boldsymbol{x}}{\|\boldsymbol{y} - \boldsymbol{x}^\top \boldsymbol{y}\,\boldsymbol{x}\|}. \tag{6}$$

The parallel transport of $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{S}^d$ to $\mathcal{T}_{\boldsymbol{y}}\mathcal{S}^d$ is given by

$$\Gamma_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{v}) = \boldsymbol{v} - \frac{\mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y})^\top \boldsymbol{v}}{d(\boldsymbol{x}, \boldsymbol{y})^2}\Big(\mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y}) + \mathrm{Log}_{\boldsymbol{y}}(\boldsymbol{x})\Big). \tag{7}$$

In some applications, it can be convenient to define the parallel transport with the alternative equivalent form

$$\begin{aligned} &\Gamma_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{v}) = \boldsymbol{A}_{\boldsymbol{x}\to\boldsymbol{y}}\,\boldsymbol{v}, \quad \text{with} \\ &\boldsymbol{A}_{\boldsymbol{x}\to\boldsymbol{y}} = -\boldsymbol{x}\sin(\|\boldsymbol{u}\|)\overline{\boldsymbol{u}}^\top + \overline{\boldsymbol{u}}\cos(\|\boldsymbol{u}\|)\overline{\boldsymbol{u}}^\top + (\boldsymbol{I} - \overline{\boldsymbol{u}}\,\overline{\boldsymbol{u}}^\top), \\ &\boldsymbol{u} = \mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y}), \quad \text{and} \quad \overline{\boldsymbol{u}} = \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}, \end{aligned} \tag{8}$$

highlighting the linear structure of the operation.

Corresponding examples in Matlab and C++ can be found in [9], named `demo_Riemannian_Sd_*.m` and `demo_Riemannian_S2_*.cpp`, respectively.

Note that in the above representation, $\boldsymbol{u}$ and $\boldsymbol{v}$ are described as vectors with $d+1$ elements contained in $\mathcal{T}_{\boldsymbol{x}}\mathcal{S}^d$. An alternative representation consists of expressing $\boldsymbol{u}$ and $\boldsymbol{v}$ as vectors of d elements in the coordinate system attached to $\mathcal{T}_{\boldsymbol{x}}\mathcal{S}^d$, see [4] for details.

$\mathcal{H}^d$ *manifold*

The exponential and logarithmic maps corresponding to the distance

$$d(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{arccosh}\Big(-\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathrm{M}}\Big), \tag{9}$$

with $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{H}^d$ can be computed as

$$\boldsymbol{y} = \mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{u}) = \boldsymbol{x}\cosh(\|\boldsymbol{u}\|_{\mathrm{M}}) + \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|_{\mathrm{M}}}\sinh(\|\boldsymbol{u}\|_{\mathrm{M}}), \tag{10}$$

$$\boldsymbol{u} = \mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y}) = d(\boldsymbol{x}, \boldsymbol{y})\,\frac{\boldsymbol{y} + \langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathrm{M}}\,\boldsymbol{x}}{\|\boldsymbol{y} + \langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathrm{M}}\,\boldsymbol{x}\|_{\mathrm{M}}}, \tag{11}$$

by using the Minkowski inner product $\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathrm{M}} = \boldsymbol{x}^\top \left(\begin{smallmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & -1 \end{smallmatrix}\right) \boldsymbol{y}$ and norm $\|\boldsymbol{x}\|_{\mathrm{M}} = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x}\rangle_{\mathrm{M}}}$. The parallel transport of $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{H}^d$ to $\mathcal{T}_{\boldsymbol{y}}\mathcal{H}^d$ is given by

$$\Gamma_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{v}) = \boldsymbol{v} - \frac{\langle \mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y}), \boldsymbol{v}\rangle_{\mathrm{M}}}{d(\boldsymbol{x}, \boldsymbol{y})^2}\Big(\mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y}) + \mathrm{Log}_{\boldsymbol{y}}(\boldsymbol{x})\Big). \tag{12}$$

Corresponding examples in Matlab can be found in [9], named `demo_Riemannian_Hd_*.m`.

$\mathcal{S}_{++}^d$ manifold

For an affine-invariant distance between $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{S}_{++}^d$

$$d(\boldsymbol{X}, \boldsymbol{Y}) = \left\| \log(\boldsymbol{X}^{-\frac{1}{2}} \boldsymbol{Y} \boldsymbol{X}^{-\frac{1}{2}}) \right\|_{\mathrm{F}}, \tag{13}$$

the exponential and logarithmic maps on the SPD manifold can be computed as (see also [10])

$$\boldsymbol{Y} = \mathrm{Exp}_{\boldsymbol{X}}(\boldsymbol{U}) = \boldsymbol{X}^{\frac{1}{2}} \exp\left(\boldsymbol{X}^{-\frac{1}{2}} \boldsymbol{U} \boldsymbol{X}^{-\frac{1}{2}}\right) \boldsymbol{X}^{\frac{1}{2}}, \tag{14}$$

$$\boldsymbol{U} = \mathrm{Log}_{\boldsymbol{X}}(\boldsymbol{Y}) = \boldsymbol{X}^{\frac{1}{2}} \log\left(\boldsymbol{X}^{-\frac{1}{2}} \boldsymbol{Y} \boldsymbol{X}^{-\frac{1}{2}}\right) \boldsymbol{X}^{\frac{1}{2}}. \tag{15}$$

The parallel transport of $\boldsymbol{V} \in \mathcal{T}_{\boldsymbol{X}} \mathcal{S}_{++}^d$ to $\mathcal{T}_{\boldsymbol{Y}} \mathcal{S}_{++}^d$ is given by

$$\Gamma_{\boldsymbol{X} \to \boldsymbol{Y}}(\boldsymbol{V}) = \boldsymbol{A}_{\boldsymbol{X} \to \boldsymbol{Y}} \boldsymbol{V} \boldsymbol{A}_{\boldsymbol{X} \to \boldsymbol{Y}}^{\top}, \text{ with } \boldsymbol{A}_{\boldsymbol{X} \to \boldsymbol{Y}} = \left(\boldsymbol{Y} \boldsymbol{X}^{-1}\right)^{\frac{1}{2}}. \tag{16}$$

Corresponding examples in Matlab and C++ can be found in [9], named `demo_Riemannian_SPD_*.m` and `demo_Riemannian_SPD_*.cpp`, respectively.

$\mathcal{G}^{d,p}$ manifold

For the arc length distance between two points $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{G}^{d,p}$

$$d(\boldsymbol{X}, \boldsymbol{Y}) = \| \arccos(\boldsymbol{\sigma}) \|_2, \quad \text{with} \quad \boldsymbol{\sigma} = \mathrm{diag}(\boldsymbol{\Sigma}), \tag{17}$$

computed with the singular value decomposition (SVD) $\boldsymbol{X}^{\top} \boldsymbol{Y} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^{\top}$, the exponential and logarithmic map of the Grassmann manifold are given by [43]

$$\boldsymbol{Y} = \mathrm{Exp}_{\boldsymbol{X}}(\boldsymbol{H}) = \begin{pmatrix} \boldsymbol{X}\boldsymbol{V} & \boldsymbol{U} \end{pmatrix} \begin{pmatrix} \cos \boldsymbol{\Sigma} \\ \sin \boldsymbol{\Sigma} \end{pmatrix} \boldsymbol{V}^{\top}, \tag{18}$$

computed with the SVD $\boldsymbol{H} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^{\top}$, where

$$\boldsymbol{H} = \mathrm{Log}_{\boldsymbol{X}}(\boldsymbol{Y}) = \boldsymbol{U} \arctan(\boldsymbol{\Sigma}) \boldsymbol{V}^{\top} \tag{19}$$

is computed with the SVD $(\boldsymbol{I} - \boldsymbol{X}\boldsymbol{X}^{\top}) \boldsymbol{Y} (\boldsymbol{X}^{\top} \boldsymbol{Y})^{-1} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^{\top}$. The parallel transport of $\boldsymbol{G} \in \mathcal{T}_{\boldsymbol{X}} \mathcal{G}^{d,p}$ to $\mathcal{T}_{\boldsymbol{Y}} \mathcal{G}^{d,p}$ corresponds to

$$\Gamma_{\boldsymbol{X} \to \boldsymbol{Y}}(\boldsymbol{G}) = \left(\begin{pmatrix} \boldsymbol{X}\boldsymbol{V} & \boldsymbol{U} \end{pmatrix} \begin{pmatrix} -\sin \boldsymbol{\Sigma} \\ \cos \boldsymbol{\Sigma} \end{pmatrix} \boldsymbol{U}^{\top} + (\boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^{\top}) \right) \boldsymbol{G}, \tag{20}$$

computed with the SVD $\mathrm{Log}_{\boldsymbol{X}}(\boldsymbol{Y}) = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^{\top}$.

Corresponding examples in Matlab can be found in [9], named `demo_Riemannian_Gdp_*.m`.

Computation of transfer matrices $\boldsymbol{S_u}$ and $\boldsymbol{S_x}$ in MPC

The MPC problem of estimating velocity commands $\boldsymbol{u}_t \in \mathbb{R}^d$ with a discrete linear dynamical system $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)$ can be solved by linearization with

$$\boldsymbol{x}_{t+1} = \boldsymbol{A}_t(\boldsymbol{x}_t, \boldsymbol{u}_t)\, \boldsymbol{x}_t + \boldsymbol{B}_t(\boldsymbol{x}_t, \boldsymbol{u}_t)\, \boldsymbol{u}_t, \tag{21}$$

and expressing all future states $\boldsymbol{x}_t$ as an explicit function of the state $\boldsymbol{x}_1$. By writing

$$\begin{aligned}
&\boldsymbol{x}_2 = \boldsymbol{A}_1 \boldsymbol{x}_1 + \boldsymbol{B}_1 \boldsymbol{u}_1, \\
&\boldsymbol{x}_3 = \boldsymbol{A}_2 \boldsymbol{x}_2 + \boldsymbol{B}_2 \boldsymbol{u}_2 = \boldsymbol{A}_2(\boldsymbol{A}_1 \boldsymbol{x}_1 + \boldsymbol{B}_1 \boldsymbol{u}_1) + \boldsymbol{B}_2 \boldsymbol{u}_2, \\
&\quad \vdots \\
&\boldsymbol{x}_T = \left(\prod_{t=1}^{T-1} \boldsymbol{A}_{T-t} \right) \boldsymbol{x}_1 + \left(\prod_{t=1}^{T-2} \boldsymbol{A}_{T-t} \right) \boldsymbol{B}_1 \boldsymbol{u}_1 + \\
&\qquad\qquad \left(\prod_{t=1}^{T-3} \boldsymbol{A}_{T-t} \right) \boldsymbol{B}_2 \boldsymbol{u}_2 + \cdots + \boldsymbol{B}_{T-1} \boldsymbol{u}_{T-1},
\end{aligned}$$

in a matrix form, we get an expression of the form $\boldsymbol{x} = \boldsymbol{S_x} \boldsymbol{x}_1 + \boldsymbol{S_u} \boldsymbol{u}$, with

$$\underbrace{\begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}}_{\boldsymbol{x}} = \underbrace{\begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \boldsymbol{A}_1 \\ \vdots \\ \prod_{t=1}^{T-1} \boldsymbol{A}_{T-t} \end{bmatrix}}_{\boldsymbol{S_x}} \boldsymbol{x}_1 \quad +$$

$$\underbrace{\begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{B}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{A}_2 \boldsymbol{B}_1 & \boldsymbol{B}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \left(\prod_{t=1}^{T-2} \boldsymbol{A}_{T-t}\right) \boldsymbol{B}_1 & \left(\prod_{t=1}^{T-3} \boldsymbol{A}_{T-t}\right) \boldsymbol{B}_2 & \cdots & \boldsymbol{B}_{T-1} \end{bmatrix}}_{\boldsymbol{S_u}} \underbrace{\begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_{T-1} \end{bmatrix}}_{\boldsymbol{u}},$$

where $\boldsymbol{S_x} \in \mathbb{R}^{dT \times d}$, $\boldsymbol{x}_1 \in \mathbb{R}^d$, $\boldsymbol{S_u} \in \mathbb{R}^{dT \times d(T-1)}$ and $\boldsymbol{u} \in \mathbb{R}^{d(T-1)}$.

Corresponding examples in Matlab and C++ can be found in [9], named `demo_MPC_*.m` and `demo_MPC_*.cpp`, respectively.