Riemann^2: Learning Riemannian Submanifolds from Riemannian Data

Leonel Rozo* Bosch center for AI
Miguel González-Duque* DBtune
Noémie Jaquier KTH
Søren Hauberg Technical Uni. of Denmark

Abstract

Latent variable models are powerful tools for learning low-dimensional manifolds from high-dimensional data. However, when dealing with constrained data such as unit-norm vectors or symmetric positive-definite matrices, existing approaches ignore the underlying geometric constraints or fail to provide meaningful metrics in the latent space. To address these limitations, we propose to learn Riemannian latent representations of such geometric data. To do so, we estimate the pullback metric induced by a Wrapped Gaussian Process Latent Variable Model, which explicitly accounts for the data geometry. This enables us to define geometry-aware notions of distance and shortest paths in the latent space, while ensuring that our model only assigns probability mass to the data manifold. This generalizes previous work and allows us to handle complex tasks in various domains, including robot motion synthesis and analysis of brain connectomes.

1 Introduction

Geometrically-constrained data appears in many application domains such as biology (Macaulay et al., 2023), robotics (Urain et al., 2022; Rozo and Dave, 2022), motion modeling (Jaquier et al., 2024; He et al., 2024), and medical imaging (Pennec et al., 2020; Baust and Weinmann, 2020). For example, the orientation of rigid bodies is commonly represented by unit quaternions in the hypersphere $\mathbb{S}^3$, or diffusion tensor images are encoded in the space of symmetric positive-definite matrices $\mathcal{S}^d_{++}$. One of the main challenges when analyzing or predicting this type of data arises when finding a low-dimensional subspace in a high-dimensional setting, as classical techniques are not tailored to non-Euclidean data. However, recent works on Gaussian processes latent variable models (GPLVM) (Mallasto et al., 2019) and variational autoencoders (VAE) (Miolane and Holmes, 2020; Beik-Mohammadi et al., 2021), offer compelling solutions to this problem by generating data complying with the given geometric constraints.

A key aspect the aforementioned works still overlook is the data manifold structure. In other words, the assumption that the latent space is Euclidean often fails at capturing the underlying data manifold (Shao et al., 2018; Hauberg, 2019). This problem was recently tackled by endowing the latent space of generative models with a Riemannian metric induced by the non-linear mapping of the model decoder (Tosi et al., 2014; Arvanitidis et al., 2018; Park et al., 2023). Importantly, this metric encapsulates the data uncertainty into the latent space, and therefore enables downstream tasks that comply with the data manifold geometry. This proved useful in reinforcement learning (Tennenholtz and Mannor, 2022), protein sequencing (Detlefsen et al., 2022), and robotics (Chen et al., 2018; Scannell et al., 2021; Beik-Mohammadi et al., 2021), among other disciplines.

Inspired by the foregoing works on the geometry of generative models and the need of handling data with geometric constraints in downstream tasks, we here tackle the problem of learning low-dimensional representations of Riemannian data that pull back the data space metric into the latent space. In other words, **this paper** proposes a generative model that learns Riemannian submanifolds from Riemannian data, hereinafter referred to as Riemann^2. Formally, our model learns a stochastic latent variable model $f\colon \mathcal{Z} \to \mathcal{M}$, which maps latent embeddings to Riemannian data lying on $\mathcal{M}$. The latent space $\mathcal{Z}$ is equipped with the pullback metric induced by the decoder f, similarly to Tosi et al. (2014). This way, we immerse the latent space, via the nonlinear mapping f, into an ambient Riemannian manifold. Specifically, we employ the geometry-aware wrapped GPLVM (WGPLVM) (Mallasto et al., 2019) to represent f, so that the decoded latent embeddings are guaranteed to lie on $\mathcal{M}$. Unlike Tosi et al. (2014) and Mallasto et al. (2019), Riemann^2 builds on multitask Gaussian

Proceedings of the 28$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. * indicates equal contribution.

	UQ	Riemannian Data	Pullback metric	Output correlation
Arvanitidis et al. (2018)	✗	✗	✓	✗
Beik-Mohammadi et al. (2021)	✗	(✓)	✓	✗
Tosi et al. (2014)	✓	✗	✓	✗
Mallasto et al. (2019)	✓	✓	✗	✗
Miolane and Holmes (2020)	✓	✓	✗	✗
RIEMANN^2	✓	✓	✓	✓

Table 1: Comparison of RIEMANN^2 against previous works. Note that the manifold-aware components of (Beik-Mohammadi et al., 2021) are specific to the hypersphere $\mathbb{S}^3$.

Processes (GPs) (Bonilla et al., 2007), which are crucial to account for correlated outputs. This is relevant to account for correlations in products of manifolds, often found in, e.g., synchronized motion generation.

In summary, **our contributions are**: (1) We define latent space geometries for multitask WGPLVMs; (2) we derive an expression for the distribution of the Riemannian pullback metric for multitask wrapped GPs, thus generalizing the work of Tosi et al. (2014); (3) we define back-constraints for WGPLVMs by leveraging Riemannian kernels, accounting for the Riemannian geometry of the observation space; and (4) we formulate the wrapped GP likelihood to account for the change of volume of the distribution. Our experiments confirm the importance of considering both the data intrinsic geometry and its distribution.

2 Related Work

Latent generative models like GPLVMs (Lawrence, 2003; Titsias and Lawrence, 2010) and VAEs (Kingma and Welling, 2014) generally assume a Gaussian prior distribution over the mapping function or the latent variables. Interestingly, as shown by Tosi et al. (2014); Shao et al. (2018); Arvanitidis et al. (2018), the decoding function of these models induces a pullback Riemannian metric on their latent space, opening the door to a better understanding on the learned embeddings geometry. However, none of these works considered data lying on Riemannian manifolds. Several generative models such as GMMs (Jaquier et al., 2021), normalizing flows (Rezende et al., 2020; Chen and Lipman, 2024), or diffusion models (Huang et al., 2022) were formulated to account for the intrinsic geometry of the data. However, latent generative models with geometry-aware decoders are still scarce, and few recent works stand out: the Wrapped GPLVM (Mallasto et al., 2019) and the geometry-aware VAEs (Miolane and Holmes, 2020; Beik-Mohammadi et al., 2021). We take inspiration from them in this work.

Table 1 summarizes how our approach contributes to the state of the art. Tosi et al. (2014) and Arvanitidis et al. (2018) define latent space geometries using GPLVMs and VAEs, respectively. However, their decoders are not manifold-aware, and thus assign probability mass outside the specified manifold in the ambient space. Mallasto et al. (2019) defined GPLVMs on tangent bundles, thus restricting the mass to a given manifold, but did not pull the data geometry back onto the latent space. Similarly, Miolane and Holmes (2020) design a VAE using a Riemannian normal distribution for the decodings, but do not equip the latent space with a metric. Beik-Mohammadi et al. (2021) define latent space geometries of a hypersphere VAE, but rely on handcrafted uncertainty estimates. RIEMANN^2 is manifold-aware, achieves automatic uncertainty quantification (UQ) through the use of GPLVMs, and defines latent space geometries. Furthermore, it accounts for correlated outputs via multitask kernels.

3 Background

Gaussian Process Latent Variable Model: A GPLVM defines a generative mapping from latent variables $\boldsymbol{x}_i \in \mathbb{R}^Q$ to data $\boldsymbol{y}_i \in \mathbb{R}^D$ by modeling the corresponding non-linear transformation f with a multitask Gaussian process (GP) (Rasmussen and Williams, 2006; Lawrence, 2003). The data is assumed to be normally distributed, i.e., $y_{i,d} \sim \mathcal{N}(y_{i,d}; f_{i,d}, \sigma_d^2)$ with,

$$f_{i,d} \sim \mathrm{GP}(m_d(\boldsymbol{x}_i), k_d^{\boldsymbol{x}}(\boldsymbol{x}_i, \boldsymbol{x}_i)) \text{ and } \boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \quad (1)$$

where $y_{i,d}$ is the d-th dimension of $\boldsymbol{y}_i$, $m_d : \mathbb{R}^Q \to \mathbb{R}$, and $k_d^{\boldsymbol{x}} : \mathbb{R}^Q \times \mathbb{R}^Q \to \mathbb{R}$ are the GP mean and kernel function, and σ_d^2 is a hyperparameter. The main design choice in a GP is the kernel $k_d^{\boldsymbol{x}}$, which encodes the covariance between two predictions $f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)$ as a similarity measure between the corresponding inputs $\boldsymbol{x}_i, \boldsymbol{x}_j$. A common choice is the square exponential (SE) kernel parametrized by a variance σ^2 and a lengthscale θ.

A common approach to model vector-valued functions $f : \mathbb{R}^Q \to \mathbb{R}^D$ is to learn each dimension f_d independently with its own GP (Álvarez et al., 2012, Sec. 3.3). However, this disregards possible correlations between the output dimensions. We consider a more general formulation proposed by Bonilla et al. (2007), in which such correlations are modeled using a learned positive-definite matrix $k^{\boldsymbol{f}}$. The covariance thus becomes,

$$\mathrm{cov}(f_{i,r}, f_{j,s}) = k\big(f_r(\boldsymbol{x}_i), f_s(\boldsymbol{x}_j)\big) = k_{rs}^{\boldsymbol{f}} k^{\boldsymbol{x}}(\boldsymbol{x}_i, \boldsymbol{x}_j). \quad (2)$$

In other words, $k = k^{\boldsymbol{f}} \otimes k^{\boldsymbol{x}}$, where $\otimes$ is the Kronecker product. The GPLVM hyperparameters and latent variables are often optimized using *maximum likelihood* or *maximum a posteriori* (MAP) estimates. The task covariance can be parametrized by a low-rank square root $\boldsymbol{B}$ added to a diagonal variance vector $k^{\boldsymbol{f}} = (\boldsymbol{B}\boldsymbol{B}^\top + \mathrm{diag}(\boldsymbol{v}))$ (Daskalakis et al., 2022; Gardner et al., 2018a). In large-datasets settings, contemporary methods use inducing points and variational approximations of the evidence (Titsias and Lawrence, 2010). Note that, unlike neural-networks, GPLVMs are data efficient and provide automatic uncertainty quantification, a relevant aspect when estimating a stochastic Riemannian metric, as explained in Sec. 4.2.

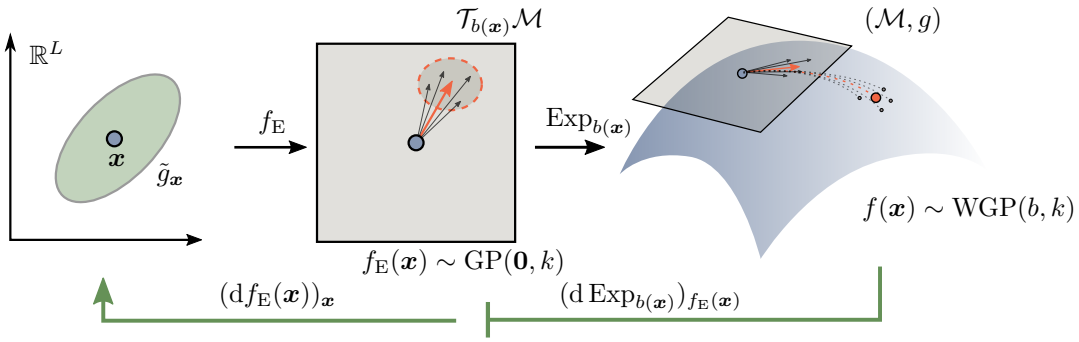


Figure 1: RIEMANN^2 : To learn a Riemannian submanifold from Riemannian data, our method pulls back a Riemannian metric $\tilde{g}_{\boldsymbol{x}}$ to a latent space via a Wrapped GPLVM. In this model, each latent code $\boldsymbol{x} \in \mathbb{R}^L$ defines a distribution of tangent vectors $f_{\mathrm{E}}(\boldsymbol{x}) \sim \mathrm{GP}(\mathbf{0}, k)$, which is then pushed forward onto the manifold $\mathcal{M}$ via the exponential map $\mathrm{Exp}_{b(\cdot)}$. Our framework enables geodesics that, when decoded, comply with the data manifold and are guaranteed to lie on $\mathcal{M}$.

Riemannian Geometry: A smooth manifold $\mathcal{M}$ is a topological space that is locally Euclidean, meaning that a neighborhood surrounding a point $\boldsymbol{p} \in \mathcal{M}$ is diffeomorphic to $\mathbb{R}^M$. We can locally approximate $\mathcal{M}$ at each point $\boldsymbol{p} \in \mathcal{M}$ with a tangent space $\mathcal{T}_{\boldsymbol{p}}\mathcal{M} \simeq \mathbb{R}^M$, defined as the set of derivatives of all smooth curves that pass through $\boldsymbol{p}$ (Lee, 2012). The collection of all tangent spaces is called the *tangent bundle*, and is formally defined as the disjoint union $\mathcal{TM} = \sqcup_{\boldsymbol{p}\in\mathcal{M}} \mathcal{T}_{\boldsymbol{p}}\mathcal{M}$.

Given a function $h\colon \mathcal{M} \to \mathcal{N}$ between two smooth manifolds, the *differential* at $\boldsymbol{p} \in \mathcal{M}$ is the linear function $\mathrm{d}h_{\boldsymbol{p}}\colon \mathcal{T}_{\boldsymbol{p}}\mathcal{M} \to \mathcal{T}_{h(\boldsymbol{p})}\mathcal{N}$ that maps tangent vectors $\boldsymbol{v}_{\boldsymbol{p}} \in \mathcal{T}_{\boldsymbol{p}}\mathcal{M}$ to tangent vectors $\mathrm{d}h_{\boldsymbol{p}}(\boldsymbol{v}_{\boldsymbol{p}}) \in \mathcal{N}$. When considering coordinates $\varphi = (x^1, \ldots, x^M)$ around $\boldsymbol{p}$ and $\psi = (y^1, \ldots, y^N)$ around $h(\boldsymbol{p})$, the differential $\mathrm{d}h_{\boldsymbol{p}}$ is represented as the Jacobian matrix,[1]

$$[\mathrm{d}h_{\boldsymbol{p}}] = \boldsymbol{J}_h(\varphi(\boldsymbol{p})) = \left[\frac{\partial (y^i \circ h)}{\partial x^j}\bigg|_{\varphi(\boldsymbol{p})} \right]_{i,j=1}^{N,M}. \quad (3)$$

Intuitively, the Jacobian (3) transforms tangent vectors on $\mathcal{M}$ to tangent vectors on $\mathcal{N}$, which allows us to define latent space geometries as discussed later.

A Riemannian manifold $(\mathcal{M}, g)$ is a smooth manifold equipped with a *Riemannian metric* $g_{\boldsymbol{p}}$, i.e., a smoothly-varying inner product over $\mathcal{T}_{\boldsymbol{p}}\mathcal{M}$ (Lee, 2018). Given a smooth curve $\gamma : [a, b] \to \mathcal{M}$, its Riemannian length is given by $\mathrm{L}[\gamma] = \int_a^b \sqrt{g_{\gamma(t)}(\dot{\gamma}(t), \dot{\gamma}(t))}\mathrm{d}t$. *Geodesics* are defined as curves that locally minimize this length. Our method deals with Riemannian submanifolds, which are locally-Euclidean topological subspaces $\mathcal{S} \subseteq \mathcal{M}$ that inherit the metric of $\mathcal{M}$ via their immersion. Specifically, the immersion $f\colon \mathcal{S} \to (\mathcal{M}, g)$ induces a *pullback* metric $\tilde{g}_{\boldsymbol{x}}$ on $\mathcal{S}$ which, for $\boldsymbol{x} \in \mathcal{S}$ and $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathcal{T}_{\boldsymbol{x}}\mathcal{S}$, is given by (Lee, 2018, Chap. 2),

$$\tilde{g}_{\boldsymbol{x}}(\boldsymbol{v}_1, \boldsymbol{v}_2) = g_{f(\boldsymbol{x})}\big(\mathrm{d}f_{\boldsymbol{x}}(\boldsymbol{v}_1), \mathrm{d}f_{\boldsymbol{x}}(\boldsymbol{v}_2)\big). \quad (4)$$

[1] We omit the choice of charts from our notation of $[\mathrm{d}h_{\boldsymbol{p}}]$, but we make this choice explicit when necessary.

Intuitively the pullback metric $\tilde{g}_{\boldsymbol{x}}$ evaluates on tangent vectors of $\mathcal{T}_{\boldsymbol{x}}\mathcal{S}$ by "moving" them to $\mathcal{T}_{f(\boldsymbol{x})}\mathcal{M}$ to compute their inner product. RIEMANN^2 defines pullback metrics in the latent spaces of manifold-aware latent variable models by explicitly computing an approximation of $\mathrm{d}f_{\boldsymbol{x}}$, where f is a GPLVM defined on manifolds.

To operate with data on Riemannian manifolds, we additionally leverage the Euclidean tangent spaces. To do so, we resort to mappings back and forth between $\mathcal{T}_{\boldsymbol{p}}\mathcal{M}$ and $\mathcal{M}$. The exponential map $\mathrm{Exp}_{\boldsymbol{p}}(\boldsymbol{u})\colon \mathcal{T}_{\boldsymbol{p}}\mathcal{M} \to \mathcal{M}$ maps a point $\boldsymbol{u} \in \mathcal{T}_{\boldsymbol{p}}\mathcal{M}$ to a point $\boldsymbol{y} \in \mathcal{M}$, so that it lies on the geodesic γ satisfying $\gamma(0) = \boldsymbol{p}$ and $\gamma'(0) = \boldsymbol{u} \in \mathcal{T}_{\boldsymbol{p}}\mathcal{M}$ at time 1. The exponential map $\mathrm{Exp}_{\boldsymbol{p}}(\boldsymbol{u})$ establishes a *local* diffeomorphism around $\boldsymbol{p}$, whose inverse function is the logarithmic map $\mathrm{Log}_{\boldsymbol{p}}(\boldsymbol{y}) : \mathcal{M} \to \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$. Intuitively, the logarithm map $\mathrm{Log}_{\boldsymbol{p}}(\boldsymbol{y})$ represents the direction and speed at which we need to "shoot" a geodesic based at $\boldsymbol{p}$ to land at $\boldsymbol{y}$.

Wrapped GPLVMs: Introduced by Mallasto et al. (2019), Wrapped GPLVMs (WGPLVM) extend GPLVMs to data lying on Riemannian manifolds. WGPLVMs build on a *Wrapped GP* (WGP) (Mallasto and Feragen, 2018)) defined on the tangent spaces of $\mathcal{M}$. To fit a WGP on a dataset $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\} \in \mathbb{R}^Q \times \mathcal{M}$, we first project the observations onto the tangent bundle $\mathcal{TM}$, constructing a new tangent space dataset $\{(\boldsymbol{x}_i, \mathrm{Log}_{b(\boldsymbol{x}_i)}(\boldsymbol{y}_i))\}$, where $b : \mathbb{R}^Q \to \mathcal{M}$ assigns a *basepoint* on the manifold $\mathcal{M}$ to each latent variable $\boldsymbol{x}$. By identifying $\mathcal{T}_{b(\boldsymbol{x})}\mathcal{M}$ with $\mathbb{R}^M$, this dataset is now Euclidean, and we can fit a GP $f_{\mathrm{E}} \sim \mathrm{GP}(\mathbf{0}, k)$ to it.

To predict over unseen points $\boldsymbol{x}_*$, we first compute the Euclidean posterior, and then push it forward onto $\mathcal{M}$ via $\mathrm{Exp}_{b(\boldsymbol{x}_*)}$. In summary, a WGP is $\mathrm{Exp}_{b(\cdot)}\big(f_{\mathrm{E}}(\cdot)\big)$, where $f_{\mathrm{E}} \sim \mathrm{GP}(\mathbf{0}, k)$ is a Euclidean GP learned on the tangent spaces $\mathcal{T}_{b(\cdot)}\mathcal{M}$. Analogously, WGPLVMs are defined as WGPs in which the latent variables $\boldsymbol{x}_i$ are unobserved and thus optimized w.r.t the likelihood of the data $\boldsymbol{y}_i$ alongside the GP hyperparameters.

4 Riemann^2

We here introduce Riemann^2 to learn Riemannian submanifolds via WGPLVMs, that is, to learn latent representations of Riemannian data alongside a Riemannian pullback metric in the latent space. From a bird's eye view, we first learn a mapping $f : \mathbb{R}^Q \to (\mathcal{M}, g)$, which we then consider as a stochastic immersion, allowing us to pull back the metric g onto the latent space $\mathbb{R}^Q$ using Eq. (4). We define the mapping f as a WGPLVM. Instead of a WGP that considers each output dimension independently, like Mallasto and Feragen (2018), we propose a multitask WGP that explicitly models the covariance across output dimensions. The Jacobian of this WGP is then used to approximate a distribution of the differential $\mathrm{d}f_{\boldsymbol{x}}$ and compute the metric $\tilde{g}_{\boldsymbol{x}}$.

Formally, let $f = \mathrm{Exp}_{b(\cdot)} \circ f_{\mathrm{E}} : \mathbb{R}^Q \to (\mathcal{M}, g)$ be a multitask WGP. The differential $\mathrm{d}f_{\boldsymbol{x}}$ is computed by applying the chain rule as,

$$\mathrm{d}f_{\boldsymbol{x}} = \mathrm{d}(\mathrm{Exp}_{b(\cdot)} \circ f_{\mathrm{E}})_{\boldsymbol{x}} = (\mathrm{d}\,\mathrm{Exp}_{b(\boldsymbol{x})})_{f_{\mathrm{E}}(\boldsymbol{x})}(\mathrm{d}f_{\mathrm{E}})_{\boldsymbol{x}}. \quad (5)$$

Therefore, we need to compute the differential of the exponential map w.r.t. the tangent GP f_{E}, and the derivative of f_{E} w.r.t. the latent variable $\boldsymbol{x}$. In coordinates, these correspond to Jacobians, as discussed in Sec. 3. Next, we compute the distribution of the Euclidean Jacobian $\boldsymbol{J}_{f_{\mathrm{E}}}(\boldsymbol{x}) = [(\mathrm{d}f_{\mathrm{E}})_{\boldsymbol{x}}]$ under the correspondence $\mathcal{T}_{b(\boldsymbol{x})}\mathcal{M} \simeq \mathbb{R}^M$. In Sec. 4.2 we then obtain a point estimate of the pullback metric $\tilde{g}$ by approximating $(\mathrm{d}\,\mathrm{Exp}_{b(\boldsymbol{x})})_{f_{\mathrm{E}}(\boldsymbol{x})}$ using the posterior mean of f_{E}. Section 4.3 discusses the problem of operating with tangent spaces and bundles in practice. Finally, training and additional priors on the latent space of Riemann^2 are introduced in Sec. 4.4 and 4.5, respectively. Riemann^2 is summarized in Algorithm 1.

4.1 Jacobian of a Multitask Euclidean GP

Following Rasmussen and Williams (2006, Chap. 9.4), the derivative of a GP is another GP as long as its covariance function is differentiable. This also holds for multitask Euclidean GPs f_{E} with covariance $k = k^{\boldsymbol{f}} \otimes k^{\boldsymbol{x}}$ (see App. A). Given the dataset $\{\boldsymbol{x}_i, f_{\mathrm{E}}(\boldsymbol{x}_i)\}_{i=1}^N \subseteq \mathbb{R}^Q \times \mathbb{R}^M$, this implies that the joint distribution of the data and the transpose Jacobian $\boldsymbol{J}^\top_{f_{\mathrm{E}}(\boldsymbol{x}_*)}$ is of the form,

$$\begin{bmatrix} \mathrm{vec}(\boldsymbol{F}) \\ \mathrm{vec}\left(\boldsymbol{J}^\top_{f_{\mathrm{E}}}(\boldsymbol{x}_*)\right) \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K} & \partial\boldsymbol{K} \\ \partial\boldsymbol{K}^\top & \partial^2\boldsymbol{K} \end{bmatrix}\right), \quad (6)$$

where we defined $\boldsymbol{F} \in \mathbb{R}^{N\times M}$ in vector form as $\mathrm{vec}(\boldsymbol{F}) = [f_1(\boldsymbol{x}_1), \ldots, f_1(\boldsymbol{x}_N), \ldots, f_m(\boldsymbol{x}_1), \ldots, f_m(\boldsymbol{x}_N)]^\top$, $\boldsymbol{K}$ is the $NM \times NM$ matrix representing the covariance between the elements in $\mathrm{vec}(\boldsymbol{F})$, and $\partial\boldsymbol{K}$, $\partial^2\boldsymbol{K}$ are the first and second derivatives of $\boldsymbol{K}$ with respect to the latent variable $\boldsymbol{x}$, which we derive next for $k = k^{\boldsymbol{f}} \otimes k^{\boldsymbol{x}}$.

Algorithm 1 Riemann^2

Input: Observations $\{\boldsymbol{y}_n\}_{n=1}^N$ with $\boldsymbol{y}_n \in \mathcal{M}$, prior on hyperparameters $p(\boldsymbol{\psi})$.
Output: Latent variables $\{\boldsymbol{x}_n\}_{n=1}^N$, hyperparameters $\boldsymbol{\psi}$ including the kernel hyperparameters, point estimates $\mathbb{E}[\tilde{\boldsymbol{G}}]$ of the Riemannian pullback metric.
Initialization:
  Set the prior distribution $p(\boldsymbol{x})$.
  Initialize the latent variables $\{\boldsymbol{x}_n\}_{n=1}^N$.
Training via MAP:
  repeat
    Compute the WGP marginal likelihood (11).
    $\boldsymbol{X}, \psi \leftarrow \mathsf{OptStep}(\log p(\boldsymbol{y}|\boldsymbol{x}))$.
  until convergence
Pullback metric:
  Compute the distribution (8) of $\boldsymbol{J}_{f_{\mathrm{E}}}$.
  Compute point estimates $\mathbb{E}[\tilde{\boldsymbol{G}}]$ using (10).

Intuitively, differentiating a kernel defined as a Kronecker product resembles the product rule. Namely, for $\boldsymbol{K} = k^{\boldsymbol{f}} \otimes \boldsymbol{K}^{\boldsymbol{x}}$ with $\boldsymbol{K}^{\boldsymbol{x}} = [k^{\boldsymbol{x}}(\boldsymbol{x}_n, \boldsymbol{x}_a)]_{n,a=1}^N$, we have,

$$\partial\boldsymbol{K} = k^{\boldsymbol{f}} \otimes \partial\boldsymbol{K}^{\boldsymbol{x}}, \partial\boldsymbol{K}^{\boldsymbol{x}} = \left[\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_n, \boldsymbol{x}_*)}{\partial x^{(r)}}\right]_{n,r=1}^{N,Q},$$

$$\partial^2\boldsymbol{K} = k^{\boldsymbol{f}} \otimes \partial^2\boldsymbol{K}^{\boldsymbol{x}}, \partial^2\boldsymbol{K}^{\boldsymbol{x}} = \left[\frac{\partial^2 k^{\boldsymbol{x}}(\boldsymbol{x}_*, \boldsymbol{x}_*)}{\partial x^{(r)}\partial x^{(s)}}\right]_{r,s=1}^{Q}.$$

Notice that the task kernel $k^{\boldsymbol{f}}$ is factored out as it does not depend on $\boldsymbol{x}$ (see App. B for details).

Using the joint distribution (6), the posterior over the Jacobian is computed as a usual GP posterior as,

$$\mathrm{vec}\left(\boldsymbol{J}^\top_{f_{\mathrm{E}}}(\boldsymbol{x}_*)\right) \sim \mathcal{N}\big(\mathrm{vec}(\partial\boldsymbol{K}^{\boldsymbol{x}\top}(\boldsymbol{K}^{\boldsymbol{x}})^{-1}\boldsymbol{F}), \quad (7)$$
$$k^{\boldsymbol{f}} \otimes (\partial^2\boldsymbol{K}^{\boldsymbol{x}} - \partial\boldsymbol{K}^{\boldsymbol{x}\top}(\boldsymbol{K}^{\boldsymbol{x}})^{-1}\partial\boldsymbol{K}^{\boldsymbol{x}})\big).$$

The posterior (7) is equivalent to the following matrix normal distribution (Gupta and Nagar, 1999, Chap. 2.),

$$\boldsymbol{J}^\top_{f_{\mathrm{E}}}(\boldsymbol{x}_*) \sim \mathcal{MN}_{Q\times M}(\partial\boldsymbol{K}^{\boldsymbol{x}\top}(\boldsymbol{K}^{\boldsymbol{x}})^{-1}\boldsymbol{F}, \quad (8a)$$
$$\partial^2\boldsymbol{K}^{\boldsymbol{x}} - \partial\boldsymbol{K}^{\boldsymbol{x}\top}(\boldsymbol{K}^{\boldsymbol{x}})^{-1}\partial\boldsymbol{K}^{\boldsymbol{x}}, \quad (8b)$$
$$k^{\boldsymbol{f}}). \quad (8c)$$

where Eqs. (8a), (8b), and (8c) are the posterior mean, covariance over rows, and covariance over columns, respectively. A more detailed derivation is provided in App. A.2. We argue that this result makes intuitive sense: The usual Jacobian posterior for the dimension-independent case appears as the posterior mean and covariance over rows (i.e., over the data), while the posterior covariance over columns is precisely the task covariance. In summary, the posterior distribution of the Jacobian of a multitask GP is the matrix normal distribution in Eq. (8). Next, we leverage this distribution to compute an estimate of the pullback metric.

4.2 Pullback Metric of a Wrapped GP

Given the distribution (8) of the differential of the Euclidean part of a WGP, the missing component to compute $\mathrm{d}f_{\boldsymbol{x}}$ in Eq. (5) is the differential of the exponential map. Since computing the posterior distribution of $(\mathrm{d}\operatorname{Exp}_{b(\boldsymbol{x})})_{f_{\mathrm{E}}(\boldsymbol{x})}$ is involved and manifold-specific, we propose a deterministic approximation given by the mean posterior of the Euclidean GP f_{E}. After evaluating the exponential map on the mean posterior $\widehat{f_{\mathrm{E}}}(\boldsymbol{x})$, the differential can be computed using autodifferentiation. We denote this approximation as $\boldsymbol{J}_{\operatorname{Exp}_{b(\boldsymbol{x})}}\big(\widehat{f_{\mathrm{E}}}(\boldsymbol{x})\big)$. We also approximate the metric evaluations $g_{f(\boldsymbol{x})}$ in Eq. (4) by using the posterior mean of the Wrapped GP f, whose matrix representation we denote by $\widehat{\boldsymbol{G}}(\boldsymbol{x})$.

By applying the foregoing approximations, we derive the pullback metric $\tilde{g}_{\boldsymbol{x}}$ of Eq. (4) in matrix form as,

$$\tilde{\boldsymbol{G}} = \boldsymbol{J}_{f_{\mathrm{E}}}^{\top}\boldsymbol{J}_{\operatorname{Exp}_{b(\boldsymbol{x})}}^{\top}\widehat{\boldsymbol{G}}\boldsymbol{J}_{\operatorname{Exp}_{b(\boldsymbol{x})}}\boldsymbol{J}_{f_{\mathrm{E}}} = \boldsymbol{J}_{f_{\mathrm{E}}}^{\top}\check{\boldsymbol{G}}\boldsymbol{J}_{f_{\mathrm{E}}}. \quad (9)$$

Finally, we leverage the distribution of the posterior Jacobian of the Euclidean GP (8) to compute a point estimate of the Riemannian pullback metric $\tilde{\boldsymbol{G}}$. By using properties of the matrix normal distribution (Gupta and Nagar, 1999, Theorem 2.3.5 (ii)), we obtain

$$\mathbb{E}[\tilde{\boldsymbol{G}}] = \mathbb{E}[\boldsymbol{J}_{f_{\mathrm{E}}}^{\top}]\,\check{\boldsymbol{G}}\,\mathbb{E}[\boldsymbol{J}_{f_{\mathrm{E}}}] + \operatorname{Tr}(\check{\boldsymbol{G}}^{\top}\boldsymbol{K}^{\boldsymbol{f}})\,\boldsymbol{\Sigma}_{\mathrm{r}}(\boldsymbol{J}_{f_{\mathrm{E}}}^{\top}), \quad (10)$$

where $\boldsymbol{\Sigma}_{\mathrm{r}}$ is the posterior covariance over rows given by Eq. (8b) (see also App. A). Importantly, the point estimate of $\tilde{\boldsymbol{G}}$ in Eq. (10) unlocks the computation of geodesics in the latent space. It is worth noting that Eq. (10) generalizes the work of Tosi et al. (2014), whose approach exclusively considers Euclidean data, i.e., $\check{\boldsymbol{G}} = \boldsymbol{I}$, and dimension-independent GPs with no correlation between output dimensions, i.e., $k^{\boldsymbol{f}} = \boldsymbol{I}$.

4.3 How to Represent Tangent Spaces

When considering WGPs, the specific representation of the tangent space $\mathcal{T}_{\boldsymbol{p}}\mathcal{M}$ is often left aside, although it is highly relevant when using WGPs in practice. This relates to the need of constructing smooth frames, i.e., collections of smoothly-varying bases over the tangent bundle $\mathcal{T}_b\mathcal{M} = \sqcup_{\boldsymbol{x}\in\mathbb{R}^Q}\mathcal{T}_{b(\boldsymbol{x})}\mathcal{M}$, where b is the basepoint function (Mallasto and Feragen, 2018; Mallasto et al., 2019). We here discuss how we implement the concepts of tangent space and bundle, and how it differs from alternative implementations.

Current software (Townsend et al., 2016; Miolane et al., 2020) implements logarithmic maps $\operatorname{Log}_{\boldsymbol{p}}(\boldsymbol{u})$ by considering their output vectors as elements of the ambient space, instead of relying on an intrinsic formulation w.r.t. the coordinates of a basis. For example, with the sphere $\mathbb{S}^2 \subseteq \mathbb{R}^3$, tangent vectors at $\boldsymbol{p} \in \mathbb{S}^2$ are considered as 3-dimensional vectors belonging to the plane orthogonal to $\boldsymbol{p} \in \mathbb{R}^3$. There exist two ways to think about $\mathcal{T}_{\boldsymbol{p}}\mathbb{S}^2$ when implementing a WGP on $\mathbb{S}^2$: *(1)* as an abstract vector space, where we place a GP prior on the coefficients of a chosen basis; *(2)* as a subspace of $\mathbb{R}^3$ whose vectors we regress directly in ambient space (Miolane et al., 2020). We find the latter interpretation potentially problematic, as in practice, current implementations learn a projected version of WGPs (Myers et al., 2022), where the learned tangent vectors do not necessarily belong to the tangent space. This problem compounds when considering manifolds such as the space of symmetric positive-definite matrices $\mathcal{S}_{++}^k$, where current implementations treat tangent vectors as $k \times k$ symmetric matrices instead of abstract vectors.

We here consider the former alternative: We define a vectorization for each $\mathcal{T}_{\boldsymbol{p}}\mathcal{M}$. Specifically, instead of considering $\operatorname{Log}_{\boldsymbol{p}}(\boldsymbol{q})$ as a vector in the ambient space where $\mathcal{M}$ is embedded in, we specify a basis of $\mathcal{T}_{\boldsymbol{p}}\mathcal{M}$ and define the output of $\operatorname{Log}_{\boldsymbol{p}}(\boldsymbol{q})$ as a linear combination of this basis. For example, in the case $\mathcal{M} = \mathbb{S}^2$, this implies learning a tangent GP on a 2-dimensional space instead of a 3-dimensional one. In the case of $\mathcal{S}_{++}^k$, this implies regressing $k(k-1)/2$ output dimensions instead of k^2. For most manifolds, such a basis (a.k.a. *frame*) cannot be built smoothly and globally. Nevertheless, we found that non-smooth choices (see App. C) work well in practice.

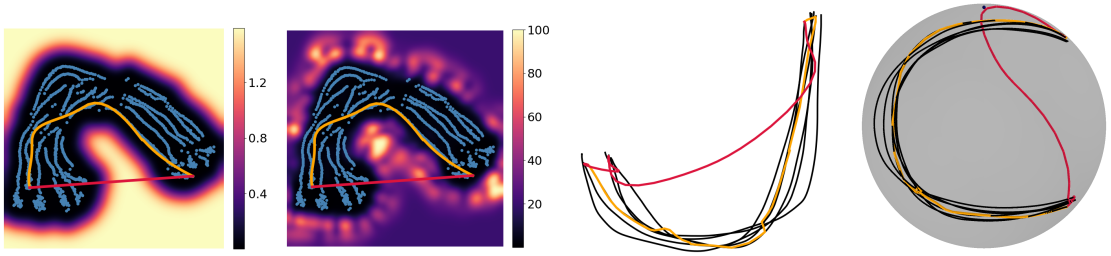
4.4 A Note on Training Wrapped Models

As mentioned in Sec. 3, the GPLVM latent variables and hyperparameters are often inferred by maximizing the marginal log-likelihood $\log p(\boldsymbol{y}|\boldsymbol{x})$ or the MAP estimate $\log\big(p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})\big)$. In the case of wrapped models, Mallasto and Feragen (2018); Mallasto et al. (2019) instead maximized the marginal likelihood of the tangent space vectors $\boldsymbol{v}_i = \operatorname{Log}_{b(\boldsymbol{x}_i)}(\boldsymbol{y}_i)$ alongside the GP parameters. However, as the exponential map pushes forward the Euclidean distribution onto $\mathcal{M}$, the marginal likelihood of a WGP must consider the change of volume induced by $\operatorname{Exp}_{b(\cdot)}$. Dismissing it may lead to ill-optimized densities. We instead account for this change in volume using the change of variable formula (see App. D) and optimize the WGP marginal log-likelihood,

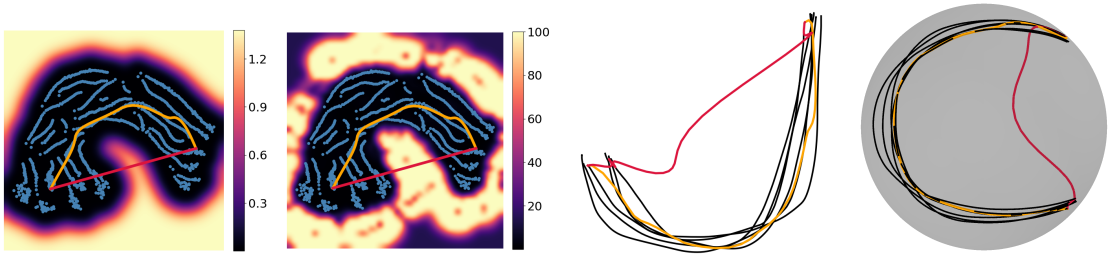
$$\log p(\boldsymbol{y}|\boldsymbol{x}) = \log\mathcal{N}\big(\boldsymbol{v};\boldsymbol{0},\boldsymbol{K}\big) - \log\det\left(\frac{\partial\operatorname{Exp}_{b(\boldsymbol{x})}}{\partial\boldsymbol{v}}\right). \quad (11)$$

4.5 Priors and Back Constraints

Priors on the latent space and back constraints (Lawrence and Quiñonero Candela, 2006; Urtasun et al., 2008) are often used to bias the structure of the latent space. Similarly, the latent space of RIEMANN^2 can be augmented with priors $p(\boldsymbol{x})$. For instance, in Sec. 5, we use the Gaussian process dynamical model (GPDM) prior (Wang et al.,



(a) WPGLVM and RIEMANN^2 latent space with one (decoded) geodesic each (—, —).



(b) GPLVM and pGPLVM latent space with one (decoded) geodesic each (—, —).

Figure 2: Illustrative example on $\mathbb{R}^2 \times \mathbb{S}^2$: From *left* to *right*: Latent variables (•) with model uncertainty and magnification factor of the pullback metrics, demonstrations (—) and decoded geodesics (—, —) on $\mathbb{R}^2$ and $\mathbb{S}^2$.

2008) to account for the temporal structure of the data. RIEMANN^2 can also be augmented with back constraints that define the latent variables as function of the data via the mapping,

$$x_{i,q} = \sum_{j=1}^{N} w_{q,j} k^{\mathcal{M}}(\boldsymbol{y}_i, \boldsymbol{y}_j). \tag{12}$$

Importantly, the mapping (12) must be manifold-aware, which we achieve by expressing data similarities via a Riemannian kernel $k^{\mathcal{M}}$, following the formulations in (Borovitskiy et al., 2020; Azangulov et al., 2023).

5 Experiments

We test RIEMANN^2 on the following experiments: *(1)* an illustrative example with synthetic data on $\mathbb{R}^2 \times \mathbb{S}^2$, *(2)* a robot end-effector motion synthesis experiment on $\mathbb{R}^3 \times \mathbb{S}^3$, *(3)* a manipulability learning scenario in $\mathbb{R}^2 \times \mathcal{S}^2_{++}$, and *(4)* brain connectomes in $\mathcal{S}^{15}_{++}$. We compare RIEMANN^2 against three GP-based baselines: *(1)* a vanilla GPLVM, which is manifold-unaware and does not equip the latent space with a metric; *(2)* the pullback GPLVM (pGPLVM) from Tosi et al. (2014), which assumes Euclidean data but endows the latent space with a pullback metric through a dimension-independent GP; and *(3)* a multitask variant of the WGPLVM proposed in (Mallasto et al., 2019), which is manifold-aware but does not pullback the data manifold geometry in the latent space. All models are trained and tested under the same settings. The latent spaces of WPGLVM and RIEMANN^2 differ from the latent spaces of GPLVM and pGPLVM due to the intrinsic differences between the Euclidean and wrapped GP models. Note that we do not compare against VAE-based approaches (Miolane and Holmes, 2020; Beik-Mohammadi et al., 2021) as these models are structurally different from GPLVMs. For example, as discussed in Sec. 1, they rely on handcrafted uncertainty quantification. Experimental settings, geodesics optimization, and training parameters for all models are detailed in App. E. A supplementary video and open source code can be found in `https://sites.google.com/view/riemann2`.

5.1 Illustrative Example on $\mathbb{R}^2 \times \mathbb{S}^2$

To illustrate the main differences between RIEMANN^2 and the baselines, we first learn 2-dimensional latent spaces out of trajectories on $\mathbb{R}^2 \times \mathbb{S}^2$. As in (Beik-Mohammadi et al., 2021), we consider a J shape in $\mathbb{R}^2$ and a C shape projected on $\mathbb{S}^2$. All approaches are augmented with a GPDM latent prior to account for the temporal structure of the trajectories. Figure 2 shows the learned latent spaces as well as a geodesic in each latent space and the resulting decoded trajectory. Note that the GPLVM and pGPLVM share the same latent space as the pullback metric of the pGPLVM is obtained after training the GPLVM. This also applies to WGPLVM and RIEMANN^2 . The background colors in the first and second columns represent the model uncertainty and the magnification factor $\sqrt{\det(\tilde{\boldsymbol{G}})}$ of RIEMANN^2 and pGPLVM pullback metrics.

We observe that the magnification factor is low in the data manifold and high on its boundary, for both pullback approaches. This induces the Riemannian geodesic to travel along the data manifold, while Euclidean

Table 2: Percentage of geodesics on manifold (%) and average dynamic time warping distance (DTWD) between the geodesic of each model and the demonstrations.

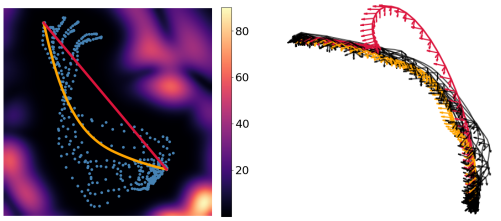
		GPLVM	pGPLVM	WGPLVM	RIEMANN^2
$\mathbb{R}^2\times\mathcal{S}^2$	%	0.01	0.0	**1.0**	**1.0**
	DTWD	13.2 ± 0.98	$\mathbf{3.4 \pm 0.77}$	11.9 ± 0.91	3.8 ± 0.91
$\mathbb{R}^3\times\mathcal{S}^3$	%	0.02	0.04	**1.0**	**1.0**
	DTWD	1.08 ± 0.06	0.63 ± 0.11	0.94 ± 0.09	$\mathbf{0.57 \pm 0.18}$
$\mathcal{S}^{15}_{++}$	%	0.76	**1.0**	**1.0**	**1.0**

geodesics, i.e., straight lines, cross empty regions in the latent space. Consequently, the decoded Riemannian geodesics of pGPLVM and RIEMANN^2 resemble the training trajectories unlike the decoded Euclidean geodesics of GPLVM and WGPLVM. However, only the decoded trajectories obtained from the manifold-aware WGPLVM and RIEMANN^2 intrinsically stay on $\mathbb{R}^2 \times \mathbb{S}^2$. These observations are quantitatively supported by the metrics reported in Table 2, which reports the percentage of each geodesic belonging to $\mathbb{R}^2 \times \mathbb{S}^2$, and their dynamic time warping distance (DTWD) indicating their resemblance with the training data. Overall, RIEMANN^2 is the only model that is manifold-aware and captures the underlying data manifold.

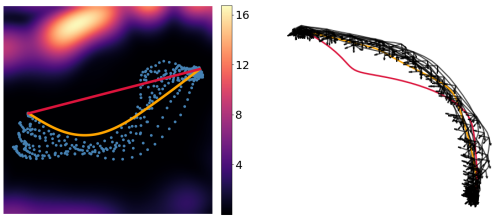
5.2 Robot Motion Synthesis on $\mathbb{R}^3 \times \mathbb{S}^3$

Learning from demonstrations (LfD) consists of modeling and generating robot motions from a set of human examples. In this context, Beik-Mohammadi et al. (2021) proposed to learn a latent Riemannian manifold from demonstrations and to leverage the corresponding geodesics to synthesize robot motions. They considered a common LfD downstream task, namely a reach-and-pick-up task performed by a 7-DoF robotic arm. The data was collected from real-world human demonstrations and consisted of end-effector positions and rotations encoded as points in $\mathbb{R}^3 \times \mathbb{S}^3$. We evaluate RIEMANN^2 in a subset of this dataset, consisting of one starting and one target point. We here employ back-constrained models, where $k^{\mathcal{M}}$ is defined as the product $k^{\mathbb{R}} \times k^{\mathbb{S}}$ of an Euclidean SE and sphere SE (Borovitskiy et al., 2020), for the wrapped models. For Euclidean models, we use only an Euclidean SE kernel.

Figure 3-*left* shows the latent spaces learned by the pullback models along with the corresponding magnification factors, as well as one geodesic for each model. The corresponding decoded trajectories are represented as frames on the *right* panels. The pullback metrics of pGPLVM and RIEMANN^2 encode the data manifold in the latent space and generate geodesics that follow the data distribution (see also the DTWD in Table 2). However, pGPLVM, as the GPLVM, fail to generate valid orientations in $\mathbb{S}^3$, as shown in Fig. 3b-*right* and Table 2. In contrast, RIEMANN^2 generates valid trajectories resembling the training data in $\mathbb{R}^3 \times \mathbb{S}^3$. We use the decoded trajectories of WGPLVM and RIEMANN^2 as reference positions and orientations to be tracked

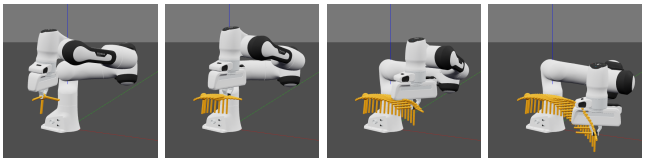


(a) WPGLVM (—) and RIEMANN^2 (—).

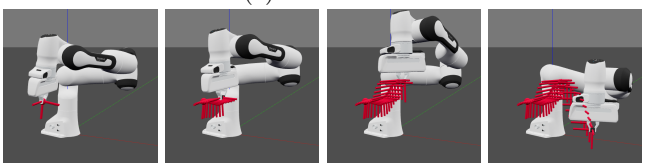


(b) GPLVM (—) and pGPLVM (—).

Figure 3: $\mathbb{R}^3 \times \mathbb{S}^3$: *Left*: Latent variables (•) with magnification factor of the pullback metrics. One geodesic is depicted per model in the corresponding latent space. *Right*: Demonstrations (—) and reconstructions represented as positions and rotation frames. Rotations are not depicted for GPLVM and pGPLVM as their reconstructions do not lie on $\mathbb{S}^3$.



(a) RIEMANN^2.



(b) WGPLVM.

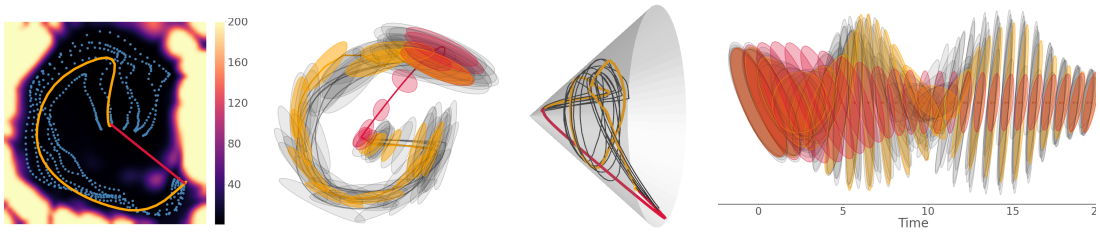
Figure 4: $\mathbb{R}^3 \times \mathbb{S}^3$: Robot motions generated from the decoded WGPLVM and RIEMANN^2 geodesics. End-effector trajectories are represented as position and rotation frames.

by the robot to execute the reach-and-pick task. The resulting robot motions are shown in Fig. 4. Note that GPLVM and pGPLVM cannot be used to generate robot motions as they do not lead to valid orientations.

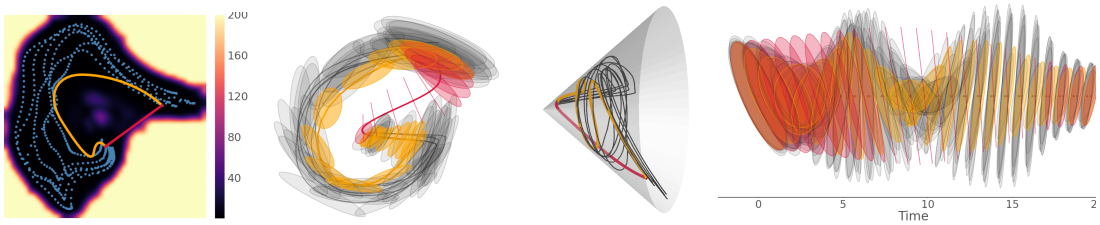
5.3 Manipulability Learning in $\mathbb{R}^2 \times \mathcal{S}^2_{++}$

Next, we test RIEMANN^2 on a challenging robot learning downstream task: Manipulability learning on a highly-redundant planar robot. Manipulability ellipsoids are defined as functions of the robot joint configuration $\boldsymbol{q}$ as $\boldsymbol{M}(\boldsymbol{q}) = \boldsymbol{J}(\boldsymbol{q})^\top \boldsymbol{J}(\boldsymbol{q})$, with $\boldsymbol{J}(\boldsymbol{q})$ being the robot Jacobian[2]. They indicate the preferred directions of velocity control. The considered task involves learning jointly a robot end-effector trajectory and a manipulability profile. Robot motions are then generated by following the

[2] $\boldsymbol{J}(\boldsymbol{q})$ is computed from the robot kinematics and should not be confused with the Jacobian introduced in Sec. 3.



(a) WPGLVM and RIEMANN^2 latent space with one (decoded) geodesic each (—, —).



(b) GPLVM and pGPLVM latent space with one (decoded) geodesic each (—, —).

Figure 5: $\mathbb{R}^2 \times \mathcal{S}^2_{++}$: From *left* to *right*: Latent variables (•) with magnification factor of the pullback metrics, demonstrations (—,⬭) and reconstructions depicted as curves and ellipsoids in $\mathbb{R}^2$, on the manifold $\mathcal{S}^2_{++}$, and as ellipsoids over time.

Table 3: Percentage of geodesics on manifold (%) and average dynamic time warping distance (DTWD) between the geodesic and the demonstrations for the manipulability learning experiment on $\mathbb{R}^2 \times \mathcal{S}^2_{++}$.

	Independent output dimensions				**Correlated output dimensions**			
	GPLVM	**pGPLVM**	**WGPLVM**	RIEMANN^2	**GPLVM**	**pGPLVM**	**WGPLVM**	RIEMANN^2
%	0.52	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
DTWD	18.3 ± 1.8	6.1 ± 1.9	18.3 ± 1.8	3.5 ± 0.8	17.0 ± 1.9	4.9 ± 1.7	16.3 ± 1.9	$\mathbf{3.5 \pm 0.3}$

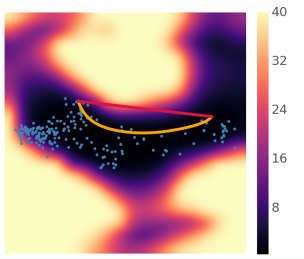
desired end-effector trajectory as the main control task, and the desired manipulability profile as a secondary objective, as proposed by (Jaquier et al., 2021). In this setting, we learn a Riemannian submanifold from data in $\mathbb{R}^2 \times \mathcal{S}^2_{++}$. Specifically, we consider training data consisting of the position trajectories and manipulability profiles recorded from a simulated highly-redundant planar robot following a G-shape trajectory on a 2D plane. In the demonstrations, the robot velocity manipulability is aligned with the direction of motion, thereby allowing it to follow a given trajectory at a higher speed. Note that this is a desired feature in robotic tasks, as this allows a robot to shape its posture according to the task requirements (Jaquier et al., 2021). All models are augmented with a GPDM prior and with back constraints. The kernel $k^{\mathcal{M}}$ is defined as the product $k^{\mathbb{R}} \times k^{\mathcal{S}_{++}}$ of a Euclidean SE and SPD SE (Azangulov et al., 2023) kernels, and as a Euclidean SE kernel, for the wrapped and Euclidean models, respectively.

Figure 5 shows the learned latent spaces, alongside magnification factors for the pullback metrics, geodesic trajectories, and decoded position and manipulability profiles. As for the previous experiments, the decoded geodesics generated by pGPLVM and RIEMANN^2 resemble the data distribution due to the pullback metric. Unlike the experiments with data on the hypersphere manifold, the trajectories generated by pGPLVM fulfill the SPD manifold constraints (see also Table 3). This is due to the geometry of the SPD manifold (see third panels of Fig. 5), for which linear interpolation between nearby SPD matrices remains in the SPD cone. However, the trajectories generated by RIEMANN^2 match the training data more closely than those obtained by pGPLVM, as shown by the DTWD values in Table 3. Moreover, crossing empty regions, e.g., with the GPLVM geodesic, still results in invalid trajectories.

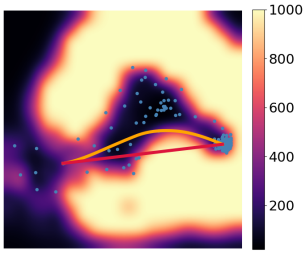
Next, we analyse the importance of considering correlated output dimensions within the multitask models. Table 3 shows that models incorporating correlated output dimensions consistently generate trajectories that more closely align with the training data patterns than those assuming independent dimensions. RIEMANN^2 with correlated outputs achieves the best results over all, showing the relevance of accounting for the data and latent space geometries, as well as the correlations between position and manipulability. Note that RIEMANN^2 with independent outputs achieves a similar DTWD mean, but higher variance. In contrast, the manipulability profiles are better reconstructed with correlated outputs, as illustrated in Fig. 8 of App. F.

5.4 Brain Connectomes in $\mathcal{S}^{15}_{++}$

Finally, we test the ability of RIEMANN^2 to learn a submanifold of high-dimensional data. Similarly as (Miolane and Holmes, 2020), we consider resting-state func-



(a) WPGLVM (—) and RIEMANN^2 (—).



(b) GPLVM (—) and pGPLVM (—).

Figure 6: $\mathcal{S}^{15}_{++}$: Latent variables (•) with magnification factor of the pullback metrics. One Euclidean (—) and one Riemannian (—) geodesic are depicted for each model.

tional brain connectome data from the "1200 Subjects release" of the Human Connectome Project (Van Essen et al., 2013). We learn a submanifold from brain connectomes from 200 subjects out of the 812 release. Each connectome is represented as a point in $\mathcal{S}^{15}_{++}$.

Figure 6 shows the latent spaces learned with GPLVM, pGPLVM, WGPLVM, and RIEMANN^2 , alongside magnification factors for the pullback metrics and geodesic trajectories. We observe that most of the latent variables are cluttered in the middle-right part of the GPLVM and pGPLVM latent space, while they are more evenly distributed in the case of RIEMANN^2 and WPGLVM. Moreover, the pGPLVM magnification factor does not draw a clear boundary around the data manifold in this case. As for previous experiments, the Euclidean models do not guarantee that the decoded geodesics belong to the given manifold (see Table 2).

5.5 Comparison with Metric Learning

We compare RIEMANN^2 with metric learning in the latent space of a trained WGPLVM. Following (Lebanon, 2002), we use a metric $(\det \boldsymbol{G}(\boldsymbol{x}))^{-1/2} \propto p(\boldsymbol{x})$, where $p(\boldsymbol{x})$ is a density estimated from data. As the density is a scalar measure, we chose the metric to be isotropic, i.e., $\boldsymbol{G}(\boldsymbol{x}) = \lambda(\boldsymbol{x})\boldsymbol{I}$ with $\lambda^{-Q/2}(\boldsymbol{x}) \propto p(\boldsymbol{x})$. We define $p(\boldsymbol{x})$ as a kernel density estimate of the latent variables, i.e., $p(\boldsymbol{x}) = \sum_{n=1}^{N} \frac{1}{(2\pi)^{Q/2}\sigma^D} \exp(-\frac{d(\boldsymbol{x},\boldsymbol{x}_n)}{2\sigma^2})$, where $\boldsymbol{x}_n$ are the training latent variables and σ is a hyperparameter defining the density variance. We compute Riemannian metrics $\boldsymbol{G}(\boldsymbol{x})$ derived from the kernel density estimate on the illustrative example of Sec. 5.1 for different values of σ, as well as corresponding geodesics for each metric with the same start and end points as in Fig. 2. The corresponding results are reported in Fig. 7 and Table 4. We observe that the WGPLVM ensures that all decoded geodesics stay on $\mathbb{R}^2 \times \mathbb{S}^2$. However, the aforementioned metric learning approach is sensitive to the hyperparameter σ: Its DTWD is similar to that of RIEMANN^2 for $\sigma = 0.25$, indicating the similarity of the generated geodesic with the training data. However, the

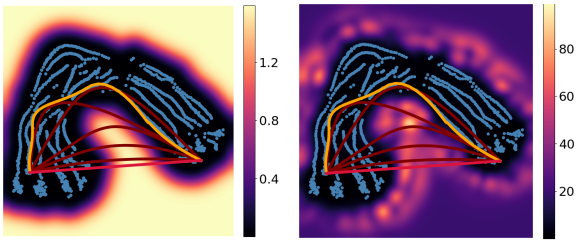


Figure 7: Metric learning in $\mathbb{R}^2 \times \mathbb{S}^2$: From *left* to *right*: Latent variables (•) with model uncertainty and magnification factor of the pullback metrics with one WGPLVM (—), one RIEMANN^2 (—), and several metric-learning (—) geodesics for different parameters σ.

Table 4: Percentage of geodesics on manifold (%) and average DTWD between the geodesic of each metric-learning model and the demonstrations.

σ	0.1	0.25	0.5	1.0	2.0
%	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
DTWD	7.3 ± 0.49	$\mathbf{3.6 \pm 0.80}$	11.6 ± 0.65	13.1 ± 0.71	12.3 ± 0.81

DTWD drastically increases for other hyperparameter values, leading to trajectories that strongly differ from the training data, similar to the WGPLVM geodesic.

Compared to metric learning on the latent space, pullback metrics offer several advantages: They avoid introducing additional parameters, thus simplifying the model,and they eliminate the need for a separate learning step after training the latent variables, streamlining the overall training process. Furthermore, metric learning is very sensitive to parameter choices.

6 Conclusions

Our goal of learning a Riemannian submanifold from Riemannian data was twofold: (1) When learning latent representations, any operation on the latent space such distances or geodesics between two embeddings, must comply with the underlying data manifold; (2) We must guarantee that the decoded latent variables lie on the Riemannian manifold of interest. Both objectives proved to be notably important when geodesics were employed as a motion generation mechanism, which may unlock applications on avatars animation or humanoid robots control. However, as RIEMANN^2 generalizes previous works, its features can also be separately employed to learn geometry-aware GP generative models, or to analyze the geometry of GPLVMs, for single or multitask GPs. For instance, if adhering to the ambient geometry is not critical, then learning a Riemannian pullback metric may suffice for data manifold estimation and sample generation. Similarly as GPLVMs, it is often relevant to incorporate additional smoothness constraints on the latent spaces of RIEMANN^2 through priors and back constraints. The latter increases the computational cost of the model due to the use of Riemannian kernels. However, they are necessary to account for the data geometry.

Acknowledgements

N. J. was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. S. H. was supported by a research grant (42062) from VILLUM FONDEN, and also partly funded by the Novo Nordisk Foundation through the Center for Basic Research in Life Science (NNF20OC0062606). S. H. received funding from the European Research Council (ERC) under the European Union's Horizon Programme (grant agreement 101125003).

References

P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2007. URL https://press.princeton.edu/absil.

M. A. Álvarez, L. Rosasco, and N. D. Lawrence. Kernels for vector-valued functions: A review. *Found. Trends Mach. Learn.*, 4(3):195–266, 2012. doi: 10.1561/2200000036.

G. Arvanitidis, L. K. Hansen, and S. Hauberg. Latent space oddity: on the curvature of deep generative models. In *Intl. Conf. on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1710.11379.

I. Azangulov, A. Smolensky, A. Terenin, and V. Borovitskiy. Stationary kernels and Gaussian processes on lie groups and their homogeneous spaces ii: non-compact symmetric spaces. *arXiv preprint arXiv:2301.13088*, 2023. URL https://arxiv.org/abs/2301.13088.

M. Baust and A. Weinmann. *Manifold-Valued Data in Medical Imaging Applications*, pages 613–647. Springer International Publishing, 2020. ISBN 978-3-030-31351-7. URL https://doi.org/10.1007/978-3-030-31351-7_22.

H. Beik-Mohammadi, S. Hauberg, G. Arvanitidis, G. Neumann, and L. Rozo. Learning Riemannian manifolds for geodesic motion skills. In *Robotics: Science and Systems (R:SS)*, 2021. URL https://www.roboticsproceedings.org/rss17/p082.pdf.

E. V. Bonilla, K. Chai, and C. Williams. Multi-task Gaussian process prediction. In *Neural Information Processing Systems (NeurIPS)*, volume 20, 2007. URL https://proceedings.neurips.cc/paper/2007/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf.

V. Borovitskiy, A. Terenin, P. Mostowsky, and M. P. Deisenroth. Matérn Gaussian processes on Riemannian manifolds. In *Neural Information Processing Systems (NeurIPS)*, pages 12426–12437, 2020. URL https://proceedings.neurips.cc/paper/2020/file/92bf5e6240737e0326ea59846a83e076-Paper.pdf.

N. Boumal. *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023. doi: 10.1017/9781009166164.

N. Chen, A. Klushyn, A. Paraschos, D. Benbouzid, and P. Van der Smagt. Active learning based on data uncertainty and model sensitivity. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1547–1554, 2018. doi: 10.1109/IROS.2018.8593552.

R. T. Q. Chen and Y. Lipman. Flow matching on general geometries. In *Intl. Conf. on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=g7ohDlTITL.

P. Corke and J. Haviland. Not your grandmother's toolbox–the robotics toolbox reinvented for Python. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 11357–11363, 2021. doi: 10.1109/ICRA48506.2021.9561366.

C. Daskalakis, P. Dellaportas, and A. Panos. How good are low-rank approximations in Gaussian process regression? *AAAI Conf. on Artificial Intelligence*, 36 (6):6463–6470, 2022. doi: 10.1609/aaai.v36i6.20598.

N. S. Detlefsen, A. Pouplin, C. W. Feldager, C. Geng, D. Kalatzis, H. Hauschultz, M. González-Duque, F. Warburg, M. Miani, and S. Hauberg. Stochman. *GitHub.*, 2021. URL Note:https://github.com/MachineLearningLifeScience/stochman/.

N. S. Detlefsen, S. Hauberg, and W. Boomsma. Learning meaningful representations of protein sequences. *Nature Communications*, 13(1):1–12, 2022. doi: 10.1038/s41467-022-29443-w.

J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Neural Information Processing Systems (NeurIPS)*, volume 31, 2018a. URL https://arxiv.org/abs/1912.01703.

J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with gpu acceleration. In *Neural Information Processing Systems (NeurIPS)*, 2018b. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/27e8e17134dd7083b050476733207ea1-Paper.pdf.

A. Gupta and D. Nagar. *Matrix variate distributions*. Monographs and surveys in pure and applied mathematics. Taylor & Francis, 1999. ISBN 978-1-58488-046-2. URL https://doi.org/10.1201/9780203749289.

S. Hauberg. Only Bayes should learn a manifold. *arXiv preprint arXiv:1806.04994*, 2019. URL https://arxiv.org/abs/1806.04994.

Y. He, G. Tiwari, T. Birdal, J. E. Lenssen, and G. Pons-Moll. NRDF: Neural Riemannian distance fields for learning articulated pose priors. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1661–1671, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/papers/He_NRDF_Neural_Riemannian_Distance_Fields_for_Learning_Articulated_Pose_Priors_CVPR_2024_paper.pdf.

C.-W. Huang, M. Aghajohari, J. Bose, P. Panangaden, and A. Courville. Riemannian diffusion models. In *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=ecevn9kPm4.

N. Jaquier, L. Rozo, D. G. Caldwell, and S. Calinon. Geometry-aware manipulability learning, tracking, and transfer. *Intl. Journal of Robotics Research*, 40 (2-3):624–650, 2021. doi: 10.1177/0278364920946815.

N. Jaquier, L. Rozo, M. González-Duque, V. Borovitskiy, and T. Asfour. Bringing motion taxonomies to continuous domains via GPLVM on hyperbolic manifolds. In *Intl. Conf. on Machine Learning (ICML)*, 2024. URL https://openreview.net/pdf?id=ndVXXmxSC5.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Intl. Conf. on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *Intl. Conf. on Learning Representations (ICLR)*, 2014. URL http://arxiv.org/abs/1312.6114.

N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Neural Information Processing Systems (NeurIPS)*, 2003. URL https://proceedings.neurips.cc/paper/2003/file/9657c1fffd38824e5ab0472e022e577e-Paper.pdf.

N. D. Lawrence and J. Quiñonero Candela. Local distance preservation in the GP-LVM through back constraints. In *Intl. Conf. on Machine Learning (ICML)*, page 513–520, 2006. doi: 10.1145/1143844.1143909.

G. Lebanon. Learning Riemannian metrics. In *The Conf. on Uncertainty in Artificial Intelligence (UAI)*, page 362–369, 2002.

J. Lee. *Introduction to Smooth Manifolds*. Springer New York, 2nd edition, 2012. doi: 10.1007/978-1-4419-9982-5.

J. M. Lee. *Introduction to Riemannian Manifolds*. Springer, 2 edition, 2018. doi: 10.1007/978-3-319-91755-9.

M. Macaulay, A. Darling, and M. Fourment. Fidelity of hyperbolic space for bayesian phylogenetic inference. *PLOS Computational Biology*, 19(4):1–20, 04 2023. URL https://doi.org/10.1371/journal.pcbi.1011084.

A. Mallasto and A. Feragen. Wrapped Gaussian process regression on Riemannian manifolds. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2018. URL https://doi.org/10.1109/CVPR.2018.00585.

A. Mallasto, S. Hauberg, and A. Feragen. Probabilistic Riemannian submanifold learning with wrapped Gaussian process latent variable models. In *Intl. Conf. on Artificial Intelligence and Statistic (AISTATS)*, pages 2368–2377, 2019. URL https://proceedings.mlr.press/v89/mallasto19a.html.

N. Miolane and S. Holmes. Learning weighted submanifolds with variational autoencoders and Riemannian variational autoencoders. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 14491–14499, 2020. URL 10.1109/CVPR42600.2020.01451.

N. Miolane, N. Guigui, A. L. Brigant, J. Mathe, B. Hou, Y. Thanwerdas, S. Heyder, O. Peltre, N. Koep, H. Zaatiti, H. Hajri, Y. Cabanes, T. Gerald, P. Chauchat, C. Shewmake, D. Brooks, B. Kainz, C. Donnat, S. Holmes, and X. Pennec. Geomstats: A python package for Riemannian geometry in machine learning. *Journal of Machine Learning Research*, 21(223): 1–9, 2020. URL http://jmlr.org/papers/v21/19-027.html.

P. Mostowsky, V. Dutordoir, I. Azangulov, N. Jaquier, M. J. Hutchinson, A. Ravuri, L. Rozo, A. Terenin, and V. Borovitskiy. The GeometricKernels package: Heat and Matérn kernels for geometric learning on manifolds, meshes, and graphs. *arXiv preprint arXiv:2407.08086*, 2024. URL https://arxiv.org/abs/2407.08086.

A. Myers, S. Utpala, S. Talbar, S. Sanborn, C. Shewmake, C. Donnat, J. Mathe, U. Lupo, R. Sonthalia, X. Cui, T. Szwagier, A. Pignet, A. Bergsson, S. Hauberg, D. Nielsen, S. Sommer, D. Klindt, E. Hermansen, M. Vaupel, B. Dunn, J. Xiong, N. Aharony, I. Pe'er, F. Ambellan, M. Hanik, E. Nava-Yazdani, C. von Tycowicz, and N. Miolane. Iclr 2022 challenge for computational geometry and topology: Design and results, 2022. URL https://zenodo.org/record/6554616.

Y.-H. Park, M. Kwon, J. Choi, J. Jo, and Y. Uh. Understanding the latent space of diffusion models through the lens of Riemannian geometry. In *Intl. Conf.*

on Learning Representations (ICLR), 2023. URL https://openreview.net/forum?id=VUlYp3jiEI.

X. Pennec, P. Fillard, and N. Ayache. A Riemannian framework for tensor computing. *International Journal of Computer Vision*, 66(1):41–66, 2006. doi: 10.1007/s11263-005-3222-z.

X. Pennec, S. Sommer, and T. Fletcher, editors. *Riemannian Geometric Statistics in Medical Image Analysis*. Academic Press, United States, 1. edition, 2020. doi: 10.1016/C2017-0-01561-6.

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. doi: 10.7551/mitpress/3206.001.0001.

D. J. Rezende, G. Papamakarios, S. Racaniere, M. Albergo, G. Kanwar, P. Shanahan, and K. Cranmer. Normalizing flows on tori and spheres. In *Intl. Conf. on Machine Learning (ICML)*, pages 8083–8092, 2020. URL https://proceedings.mlr.press/v119/rezende20a.html.

L. Rozo and V. Dave. Orientation probabilistic movement primitives on Riemannian manifolds. In *Conference on Robot Learning (CoRL)*, volume 164 of *Proceedings of Machine Learning Research*, pages 373–383, 2022. URL https://openreview.net/forum?id=csMg2h_LR37.

A. Scannell, C. H. Ek, and A. Richards. Trajectory optimisation in learned multimodal dynamical systems via latent-ODE collocation. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 12745–12751, 2021. doi: 10.1109/ICRA48506.2021.9561362.

H. Shao, A. Kumar, and P. T. Fletcher. The riemannian geometry of deep generative models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, page 428–4288, 2018. URL https://ieeexplore.ieee.org/document/8575533/.

G. Tennenholtz and S. Mannor. Uncertainty estimation using Riemannian model dynamics for offline reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2022. URL https://openreview.net/forum?id=pGLFkjgVvVe.

M. K. Titsias and N. D. Lawrence. Bayesian Gaussian process latent variable model. In *Intl. Conf. on Artificial Intelligence and Statistic (AISTATS)*, pages 844–851, 2010. URL https://proceedings.mlr.press/v9/titsias10a.html.

A. Tosi, S. Hauberg, A. Vellido, and N. D. Lawrence. Metrics for probabilistic geometries. In *The Conf. on Uncertainty in Artificial Intelligence (UAI)*, Quebec, Canada, July 2014. URL https://arxiv.org/abs/1411.7432.

J. Townsend, N. Koep, and S. Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137):1–5, 2016. URL http://jmlr.org/papers/v17/16-177.html.

J. Urain, D. Tateo, and J. Peters. Learning stable vector fields on Lie groups. *IEEE Robotics and Automation Letters*, 7(4):12569–12576, 2022. URL https://doi.org/10.1109/LRA.2022.3219019.

R. Urtasun, D. J. Fleet, A. Geiger, J. Popović, T. J. Darrell, and N. D. Lawrence. Topologically-constrained latent variable models. In *Intl. Conf. on Machine Learning (ICML)*, page 1080–1087, 2008. doi: 10.1145/1390156.1390292.

D. C. Van Essen, S. M. Smith, D. Barch, T. E. J. Behrends, E. Yacoub, and K. Ugurbil. The wu-minn human connectome project: An overview. *Neuroimage*, 80:62–79, 2013. URL https://doi.org/10.1016/j.neuroimage.2013.05.041.

J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):283–298, 2008. doi: 10.1109/TPAMI.2007.1167.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Yes.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. No, but the code will be released upon acceptance of the paper.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes.
   (b) Complete proofs of all theoretical results. Yes.
   (c) Clear explanations of any assumptions. Yes.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). No, the code will be released upon acceptance of the paper.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). Not Applicable.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. Yes.
   (b) The license information of the assets, if applicable. Yes.
   (c) New assets either in the supplemental material or as a URL, if applicable. Not Applicable.
   (d) Information about consent from data providers/curators. Not Applicable.
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not Applicable.
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable.
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable.

A Distribution of the pullback metric of Riemann^2

This section details the derivation of the pullback Riemannian metric of the latent space of RIEMANN^2 , presented in Section 4 of the main paper.

A.1 The pullback metric

In this paper, we model manifold-valued functions $f = \mathrm{Exp}_{b(\cdot)} \circ f_{\mathrm{E}} : \mathbb{R}^Q \to (\mathcal{M}, g)$ using WGPs. We then use this mapping to pull back the metric g, defining a custom geometry in the latent space $\mathbb{R}^d$. Given two tangent vectors $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathcal{T}_{\boldsymbol{x}}\mathbb{R}^d$, the pullback metric $\tilde{g} = f^* g$, with * denoting the pullback operator, is defined as in Eq. (4) (Lee, 2018, Chap. 2) by,

$$\tilde{g}_{\boldsymbol{x}}(\boldsymbol{v}_1, \boldsymbol{v}_2) = g_{f(\boldsymbol{x})}\Big(\mathrm{d}f_{\boldsymbol{x}}(\boldsymbol{v}_1), \mathrm{d}f_{\boldsymbol{x}}(\boldsymbol{v}_2)\Big).$$

In other words, we carry the tangent vectors from $\mathcal{T}_{\boldsymbol{x}}\mathbb{R}^d$ to $\mathcal{T}_{f(\boldsymbol{x})}\mathcal{M}$ using the differential of f, and compute their inner product there. Given a choice of coordinates, the pullback metric $\tilde{g}$ can be represented in matrix form. Specifically, considering the matrix representation of $(f^*g)_{\boldsymbol{x}}$, $g_{f(\boldsymbol{x})}$ and $(\mathrm{d}f)_{\boldsymbol{x}}$, we get,

$$\boldsymbol{v}_1^\top [\tilde{g}_{\boldsymbol{x}}] \boldsymbol{v}_2 = \boldsymbol{v}_1^\top \tilde{\boldsymbol{G}}(\boldsymbol{x}) \boldsymbol{v}_2 = \boldsymbol{v}_1^\top \boldsymbol{J}_{f(\boldsymbol{x})}^\top \boldsymbol{G}(f(\boldsymbol{x})) \boldsymbol{J}_{f(\boldsymbol{x})} \boldsymbol{v}_2 = \boldsymbol{v}_1^\top \boldsymbol{J}_{f(\boldsymbol{x})}^\top [g_{f(\boldsymbol{x})}] \boldsymbol{J}_{f(\boldsymbol{x})} \boldsymbol{v}_2.$$

This implies that, in coordinates, $\tilde{\boldsymbol{G}} = \boldsymbol{J}_f^\top \boldsymbol{G} \boldsymbol{J}_f$. When the mapping is a WGP, all the components of this product are stochastic and induce a distribution over the metric $\tilde{\boldsymbol{G}}$. As Tosi et al. (2014), we consider the expected pullback metric $\mathbb{E}[\tilde{\boldsymbol{G}}]$ as a point estimate of the pullback metric distribution, as explained in Section 4.2. To obtain the expected pullback metric, we need to compute the Jacobian $\boldsymbol{J}_f = \boldsymbol{J}_{\mathrm{Exp}_{b(\cdot)}} \boldsymbol{J}_{f_{\mathrm{E}}}$. The computation of the first Jacobian $\boldsymbol{J}_{\mathrm{Exp}_{b(\cdot)}}$ is explained in Section 4.2. The computation of the second Jacobian is briefly presented in Section 4.1 and further elaborated next.

A.2 The distribution of the Jacobian of a multitask Gaussian Process

Here, we focus on computing the distribution of the Jacobian $\boldsymbol{J}_{f_{\mathrm{E}}}$ of a multitask GP f_{E} at a new test point $\boldsymbol{x}_*$ and elaborate the derivation presented in Section 4.1. First, we derive the joint distribution of the data and the transpose Jacobian given by Eq. (6). Due to the linearity of the differentiation operator, the derivative of a GP $f_{\mathrm{E}} \sim \mathrm{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}))$ is given by another GP (Rasmussen and Williams, 2006, Ch. 9),

$$\boldsymbol{J}_{f_{\mathrm{E}}(\boldsymbol{x})} = \frac{\partial f_{\mathrm{E}}(\boldsymbol{x})}{\partial \boldsymbol{x}} \sim \mathrm{GP}\left(\frac{\partial \mu(\boldsymbol{x})}{\partial \boldsymbol{x}}, \frac{\partial^2 k(\boldsymbol{x}, \boldsymbol{x})}{\partial \boldsymbol{x}^2}\right). \tag{13}$$

Therefore, the distribution of the Jacobian of f_{E} is governed by the partial derivatives of the kernel,

$$\mathrm{cov}\left(f_j(\boldsymbol{x}_i), \frac{\partial f_m(\boldsymbol{x}_*)}{\partial x_r}\right) = \frac{\partial k_{jm}(\boldsymbol{x}_i, \boldsymbol{x}_*)}{\partial x^{(r)}}, \tag{14}$$

$$\mathrm{cov}\left(\frac{\partial f_j(\boldsymbol{x}_*)}{\partial x^{(r)}}, \frac{\partial f_m(\boldsymbol{x}_*)}{\partial x^{(s)}}\right) = \frac{\partial^2 k_{jm}(\boldsymbol{x}_i, \boldsymbol{x}_*)}{\partial x^{(r)} \partial x^{(s)}}, \tag{15}$$

where $f_j(\boldsymbol{x}_i)$ is the j-th component of $f(\boldsymbol{x}_i)$, and $x^{(r)}$ is the r-th component of a latent variable $\boldsymbol{x}$.

To compute the posterior distribution of the Jacobian $\boldsymbol{J}_{f_{\mathrm{E}}}$ at a test point $\boldsymbol{x}_*$, we consider the matrix $\boldsymbol{Y}$ containing our training outputs and the transpose Jacobian of f_{E} denoted $\boldsymbol{J}_{f_{\mathrm{E}}}^\top$,

$$\boldsymbol{Y} = \begin{bmatrix} y_1(\boldsymbol{x}_1) & y_2(\boldsymbol{x}_1) & \cdots & y_M(\boldsymbol{x}_1) \\ y_1(\boldsymbol{x}_2) & y_2(\boldsymbol{x}_2) & \cdots & y_M(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ y_1(\boldsymbol{x}_N) & y_2(\boldsymbol{x}_N) & \cdots & y_M(\boldsymbol{x}_N) \end{bmatrix}_{N \times M}, \quad \text{and} \quad \boldsymbol{J}_{f_{\mathrm{E}}(\boldsymbol{x}_*)}^\top = \begin{bmatrix} \frac{\partial y_1(\boldsymbol{x}_*)}{\partial x^{(1)}} & \frac{\partial y_2(\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & \frac{\partial y_M(\boldsymbol{x}_*)}{\partial x^{(1)}} \\ \frac{\partial y_1(\boldsymbol{x}_*)}{\partial x^{(2)}} & \frac{\partial y_2(\boldsymbol{x}_*)}{\partial x^{(2)}} & \cdots & \frac{\partial y_M(\boldsymbol{x}_*)}{\partial x^{(2)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1(\boldsymbol{x}_*)}{\partial x^{(Q)}} & \frac{\partial y_2(\boldsymbol{x}_*)}{\partial x^{(Q)}} & \cdots & \frac{\partial y_M(\boldsymbol{x}_*)}{\partial x^{(Q)}} \end{bmatrix}_{Q \times M}.$$

Using Eqs. (14) and (15)), we first obtain the joint distribution of the data $\boldsymbol{Y}$ and transpose Jacobian $\boldsymbol{J}_{f_{\mathrm{E}}(\boldsymbol{x}_*)}^\top$ given by,

$$\begin{bmatrix} \mathrm{vec}(\boldsymbol{Y}) \\ \mathrm{vec}(\boldsymbol{J}_{f_{\mathrm{E}}(\boldsymbol{x}_*)}^\top) \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K} & \partial \boldsymbol{K} \\ \partial \boldsymbol{K}^\top & \partial^2 \boldsymbol{K} \end{bmatrix}\right), \tag{16}$$

where $\boldsymbol{K}$ is a block of size $NM \times MN$ which contains the covariances between the elements in $\text{vec}(\boldsymbol{Y})$, $\partial\boldsymbol{K}$ is a block of size $NM \times QM$ of partial derivatives w.r.t one latent dimension (14), and $\partial^2\boldsymbol{K}$ is a block of size $QM \times QM$ of second partial derivatives (15).

Previous literature on latent geometries in GPLVMs (Tosi et al., 2014) modelled multivariate outputs using an independent GP for each dimension, or in other words, considered the output dimensions to be uncorrelated. In contrast, we model multivariate outputs using multitask GPs (Bonilla et al., 2007) via the multitask covariance of Eq. (2). This covariance is written succinctly as $k = k^{\boldsymbol{f}} \otimes k^{\boldsymbol{x}}$, where $k^{\boldsymbol{f}}$ is a symmetric positive-definite task covariance parametrized as $k^{\boldsymbol{f}} = \boldsymbol{B}\boldsymbol{B}^\top + \text{diag}(\boldsymbol{v})$ with a (low-rank) matrix $\boldsymbol{B}$ and a diagonal vector of variances $\boldsymbol{v}$, and $\otimes$ denotes the Kronecker product. For a multitask GP with covariance k, the kernel and partial derivatives of the joint distributions (16) are given by,

$$\boldsymbol{K} = k^{\boldsymbol{f}} \otimes \boldsymbol{K}^{\boldsymbol{x}}, \quad \text{with } \boldsymbol{K}^{\boldsymbol{x}} = [k^{\boldsymbol{x}}(\boldsymbol{x}_n, \boldsymbol{x}_a)]_{n,a=1}^{N}, \tag{17}$$

$$\partial\boldsymbol{K} = k^{\boldsymbol{f}} \otimes \partial\boldsymbol{K}^{\boldsymbol{x}}, \quad \text{with } \partial\boldsymbol{K}^{\boldsymbol{x}} = \left[\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_n, \boldsymbol{x}_*)}{\partial x^r}\right]_{n,r=1}^{N,d}, \tag{18}$$

$$\partial^2\boldsymbol{K} = k^{\boldsymbol{f}} \otimes \partial^2\boldsymbol{K}^{\boldsymbol{x}}, \quad \text{with } \partial^2\boldsymbol{K}^{\boldsymbol{x}} = \left[\frac{\partial^2 k^{\boldsymbol{x}}(\boldsymbol{x}_*, \boldsymbol{x}_*)}{\partial x^r \partial x^s}\right]_{r,s=1}^{d}. \tag{19}$$

The explicit computation of Eqs. (17)-(19) is provided in App. B. We argue that these equations are intuitive as the derivative of a Kronecker product follows the product rule, and the tasks kernel does not depend on $\boldsymbol{x}$.

Second, we compute the posterior over $\text{vec}(\boldsymbol{J}_{f_{\text{E}}(\boldsymbol{x}_*)}^\top)$ by conditioning the joint distribution (16) on $\text{vec}(\boldsymbol{Y})$. Using the multitask kernel expressions (17)-(19), we have,

$$\begin{aligned}
\text{vec}(\boldsymbol{J}_{f_{\text{E}}(\boldsymbol{x}_*)}^\top) &\sim \mathcal{N}(\partial\boldsymbol{K}^\top\boldsymbol{K}^{-1}\,\text{vec}(\boldsymbol{Y}), \partial^2\boldsymbol{K} - \partial\boldsymbol{K}^\top\boldsymbol{K}^{-1}\partial\boldsymbol{K}) && (20)\\
&= \mathcal{N}\big((k^{\boldsymbol{f}\top} \otimes \partial\boldsymbol{K}^{\boldsymbol{x}\top})((k^{\boldsymbol{f}})^{-1} \otimes (\boldsymbol{K}^{\boldsymbol{x}})^{-1})\,\text{vec}(\boldsymbol{Y}), && (21)\\
&\qquad (k^{\boldsymbol{f}} \otimes \partial^2\boldsymbol{K}^{\boldsymbol{x}}) - (k^{\boldsymbol{f}\top} \otimes \partial\boldsymbol{K}^{\boldsymbol{x}\top})((k^{\boldsymbol{f}})^{-1} \otimes (\boldsymbol{K}^{\boldsymbol{x}})^{-1})(k^{\boldsymbol{f}} \otimes \partial\boldsymbol{K}^{\boldsymbol{x}})\big)\\
&= \mathcal{N}\big((\boldsymbol{I}_M \otimes \partial\boldsymbol{K}^{\boldsymbol{x}\top}(\boldsymbol{K}^{\boldsymbol{x}})^{-1})\,\text{vec}(\boldsymbol{Y}), k^{\boldsymbol{f}} \otimes (\partial^2\boldsymbol{K}^{\boldsymbol{x}} - \partial\boldsymbol{K}^{\boldsymbol{x}\top}(\boldsymbol{K}^{\boldsymbol{x}})^{-1}\partial\boldsymbol{K}^{\boldsymbol{x}})\big) && (22)\\
&= \mathcal{N}\big(\text{vec}(\partial\boldsymbol{K}^{\boldsymbol{x}\top}(\boldsymbol{K}^{\boldsymbol{x}})^{-1}\boldsymbol{Y}), k^{\boldsymbol{f}} \otimes (\partial^2\boldsymbol{K}^{\boldsymbol{x}} - \partial\boldsymbol{K}^{\boldsymbol{x}\top}(\boldsymbol{K}^{\boldsymbol{x}})^{-1}\partial\boldsymbol{K}^{\boldsymbol{x}})\big), && (23)
\end{aligned}$$

with (23) corresponding to the posterior (7) of the main text. Note that we used properties of matrix vectorization, and the fact that the Kronecker product behaves well with respect to matrix multiplication and inversion. Equivalently, we can formulate this distribution as a matrix-valued normal (Gupta and Nagar, 1999, Chap. 2.) as in Eqs. (8a)-(8c).

B Explicit computations for multitask kernels' Jacobians

In this section, we explicitly compute the expressions (17)-(19). To do so, we consider the joint distribution of the vectors $\text{vec}(\boldsymbol{Y})$ and $\text{vec}(\boldsymbol{J}_{f_{\text{E}}(\boldsymbol{x}_*)}^\top)$ in Eq. (16). As previously mentioned, the block $\boldsymbol{K}$ is given by the covariances among the data, where we denote $y_{mn} = y_m(\boldsymbol{x}_n)$,

$$\boldsymbol{K} = \begin{bmatrix}
\text{cov}(y_{11}, y_{11}) & \cdots & \text{cov}(y_{11}, y_{N1}) & \cdots & \text{cov}(y_{11}, y_{1M}) & \cdots & \text{cov}(y_{11}, y_{NM}) \\
\vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\
\text{cov}(y_{N1}, y_{11}) & \cdots & \text{cov}(y_{N1}, y_{N1}) & \cdots & \text{cov}(y_{N1}, y_{1M}) & \cdots & \text{cov}(y_{N1}, y_{NM}) \\
\vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\
\text{cov}(y_{1M}, y_{11}) & \cdots & \text{cov}(y_{1M}, y_{N1}) & \cdots & \text{cov}(y_{1M}, y_{1M}) & \cdots & \text{cov}(y_{1M}, y_{NM}) \\
\vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\
\text{cov}(y_{NM}, y_{11}) & \cdots & \text{cov}(y_{NM}, y_{N1}) & \cdots & \text{cov}(y_{NM}, y_{1M}) & \cdots & \text{cov}(y_{NM}, y_{NM})
\end{bmatrix}.$$

For our multitask GP with multitask kernel $k = k^{\boldsymbol{f}} \otimes k^{\boldsymbol{x}}$, this covariance is explicitly given by,

$$\boldsymbol{K} = \begin{bmatrix} k^{\boldsymbol{f}}_{11}k^{\boldsymbol{x}}(\boldsymbol{x}_1,\boldsymbol{x}_1) & \cdots & k^{\boldsymbol{f}}_{11}k^{\boldsymbol{x}}(\boldsymbol{x}_1,\boldsymbol{x}_N) & \cdots & k^{\boldsymbol{f}}_{1M}k^{\boldsymbol{x}}(\boldsymbol{x}_1,\boldsymbol{x}_1) & \cdots & k^{\boldsymbol{f}}_{1M}k^{\boldsymbol{x}}(\boldsymbol{x}_1,\boldsymbol{x}_N) \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ k^{\boldsymbol{f}}_{11}k^{\boldsymbol{x}}(\boldsymbol{x}_N,\boldsymbol{x}_1) & \cdots & k^{\boldsymbol{f}}_{11}k^{\boldsymbol{x}}(\boldsymbol{x}_N,\boldsymbol{x}_N) & \cdots & k^{\boldsymbol{f}}_{1M}k^{\boldsymbol{x}}(\boldsymbol{x}_N,\boldsymbol{x}_1) & \cdots & k^{\boldsymbol{f}}_{1M}k^{\boldsymbol{x}}(\boldsymbol{x}_N,\boldsymbol{x}_N) \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ k^{\boldsymbol{f}}_{M1}k^{\boldsymbol{x}}(\boldsymbol{x}_1,\boldsymbol{x}_1) & \cdots & k^{\boldsymbol{f}}_{M1}k^{\boldsymbol{x}}(\boldsymbol{x}_1,\boldsymbol{x}_N) & \cdots & k^{\boldsymbol{f}}_{MM}k^{\boldsymbol{x}}(\boldsymbol{x}_1,\boldsymbol{x}_1) & \cdots & k^{\boldsymbol{f}}_{MM}k^{\boldsymbol{x}}(\boldsymbol{x}_1,\boldsymbol{x}_N) \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ k^{\boldsymbol{f}}_{M1}k^{\boldsymbol{x}}(\boldsymbol{x}_N,\boldsymbol{x}_1) & \cdots & k^{\boldsymbol{f}}_{M1}k^{\boldsymbol{x}}(\boldsymbol{x}_N,\boldsymbol{x}_N) & \cdots & k^{\boldsymbol{f}}_{MM}k^{\boldsymbol{x}}(\boldsymbol{x}_N,\boldsymbol{x}_1) & \cdots & k^{\boldsymbol{f}}_{MM}k^{\boldsymbol{x}}(\boldsymbol{x}_N,\boldsymbol{x}_N) \end{bmatrix}.$$

Notice that this matrix is composed of blocks $k^{\boldsymbol{f}}_{ij}[k^{\boldsymbol{x}}(\boldsymbol{x}_a,\boldsymbol{x}_b)]^N_{a,b=1}$. In other words, $\boldsymbol{K}$ can equivalently be written as $\boldsymbol{K} = k^{\boldsymbol{f}} \otimes \boldsymbol{K}^{\boldsymbol{x}}$.

Next, we focus on $\partial\boldsymbol{K}$. Denoting the partial derivatives $\partial y_m(\boldsymbol{x}_*)/\partial x^{(r)}$ as $\partial_r y_{m*}$, we obtain,

$$\partial\boldsymbol{K} = \begin{bmatrix} \operatorname{cov}(y_{11},\partial_1 y_{1*}) & \cdots & \operatorname{cov}(y_{11},\partial_d y_{1*}) & \cdots & \operatorname{cov}(y_{11},\partial_1 y_{M*}) & \cdots & \operatorname{cov}(y_{11},\partial_d y_{M*}) \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \operatorname{cov}(y_{N1},\partial_1 y_{1*}) & \cdots & \operatorname{cov}(y_{N1},\partial_d y_{1*}) & \cdots & \operatorname{cov}(y_{N1},\partial_1 y_{M*}) & \cdots & \operatorname{cov}(y_{N1},\partial_d y_{M*}) \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ \operatorname{cov}(y_{1M},\partial_1 y_{1*}) & \cdots & \operatorname{cov}(y_{1M},\partial_d y_{1*}) & \cdots & \operatorname{cov}(y_{1M},\partial_1 y_{M*}) & \cdots & \operatorname{cov}(y_{1M},\partial_d y_{M*}) \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \operatorname{cov}(y_{NM},\partial_1 y_{1*}) & \cdots & \operatorname{cov}(y_{NM},\partial_d y_{1*}) & \cdots & \operatorname{cov}(y_{NM},\partial_1 y_{M*}) & \cdots & \operatorname{cov}(y_{NM},\partial_d y_{M*}) \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial k_{11}(\boldsymbol{x}_1,\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & \frac{\partial k_{11}(\boldsymbol{x}_1,\boldsymbol{x}_*)}{\partial x^{(d)}} & \cdots & \frac{\partial k_{1M}(\boldsymbol{x}_1,\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & \frac{\partial k_{1M}(\boldsymbol{x}_1,\boldsymbol{x}_*)}{\partial x^{(d)}} \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \frac{\partial k_{11}(\boldsymbol{x}_N,\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & \frac{\partial k_{11}(\boldsymbol{x}_N,\boldsymbol{x}_*)}{\partial x^{(d)}} & \cdots & \frac{\partial k_{1M}(\boldsymbol{x}_N,\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & \frac{\partial k_{1M}(\boldsymbol{x}_N,\boldsymbol{x}_*)}{\partial x^{(d)}} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ \frac{\partial k_{M1}(\boldsymbol{x}_1,\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & \frac{\partial k_{M1}(\boldsymbol{x}_1,\boldsymbol{x}_*)}{\partial x^{(d)}} & \cdots & \frac{\partial k_{MM}(\boldsymbol{x}_1,\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & \frac{\partial k_{MM}(\boldsymbol{x}_1,\boldsymbol{x}_*)}{\partial x^{(d)}} \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ \frac{\partial k_{M1}(\boldsymbol{x}_N,\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & \frac{\partial k_{M1}(\boldsymbol{x}_N,\boldsymbol{x}_*)}{\partial x^{(d)}} & \cdots & \frac{\partial k_{MM}(\boldsymbol{x}_N,\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & \frac{\partial k_{MM}(\boldsymbol{x}_N,\boldsymbol{x}_*)}{\partial x^{(d)}} \end{bmatrix}$$

$$= \begin{bmatrix} k^{\boldsymbol{f}}_{11}\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_1,\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & k^{\boldsymbol{f}}_{11}\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_1,\boldsymbol{x}_*)}{\partial x^{(d)}} & \cdots & k^{\boldsymbol{f}}_{1M}\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_1,\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & k^{\boldsymbol{f}}_{1M}\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_1,\boldsymbol{x}_*)}{\partial x^{(d)}} \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ k^{\boldsymbol{f}}_{11}\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_N,\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & k^{\boldsymbol{f}}_{11}\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_N,\boldsymbol{x}_*)}{\partial x^{(d)}} & \cdots & k^{\boldsymbol{f}}_{1M}\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_N,\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & k^{\boldsymbol{f}}_{1M}\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_N,\boldsymbol{x}_*)}{\partial x^{(d)}} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ k^{\boldsymbol{f}}_{M1}\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_1,\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & k^{\boldsymbol{f}}_{M1}\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_1,\boldsymbol{x}_*)}{\partial x^{(d)}} & \cdots & k^{\boldsymbol{f}}_{MM}\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_1,\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & k^{\boldsymbol{f}}_{MM}\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_1,\boldsymbol{x}_*)}{\partial x^{(d)}} \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ k^{\boldsymbol{f}}_{M1}\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_N,\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & k^{\boldsymbol{f}}_{M1}\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_N,\boldsymbol{x}_*)}{\partial x^{(d)}} & \cdots & k^{\boldsymbol{f}}_{MM}\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_N,\boldsymbol{x}_*)}{\partial x^{(1)}} & \cdots & k^{\boldsymbol{f}}_{MM}\frac{\partial k^{\boldsymbol{x}}(\boldsymbol{x}_N,\boldsymbol{x}_*)}{\partial x^{(d)}} \end{bmatrix}.$$

As previously shown, we can equivalently write $\partial\boldsymbol{K} = k^{\boldsymbol{f}} \otimes \partial\boldsymbol{K}^{\boldsymbol{x}}$, with $\partial\boldsymbol{K}^{\boldsymbol{x}} = [\partial k^{\boldsymbol{x}}(\boldsymbol{x}_n,\boldsymbol{x}_*)/\partial x^{(r)}]^{N,d}_{n,r=1}$. The computations for $\partial^2\boldsymbol{K}$ are analogous, and are left to the reader.

C Tangent space representations

This section elaborates on the choice of tangent space representations for the manifolds considered in our experiments. As discussed in Section 4.3, instead of considering tangent vectors in the ambient space in which $\mathcal{M}$ is embedded, we specify them with respect to a basis of the tangent space itself. This process can be viewed as a change of basis, as explained next.

C.1 Sphere

The sphere $\mathbb{S}^M$ is embedded in the Euclidean space $\mathbb{R}^{M+1}$. Each point $\boldsymbol{p} \in \mathbb{S}^M$ is described, in coordinates, as a $(M+1)$-dimensional unit-norm vector. Each tangent vector $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{p}}\mathbb{S}^M$ is expressed as a $(M+1)$-dimensional vector orthogonal to $\boldsymbol{p}$, i.e., $\langle \boldsymbol{p}, \frac{\boldsymbol{v}}{||\boldsymbol{v}||} \rangle = 0$. Principal operations on the sphere are given by Table 5-*left*. When considering WGPs, we express tangent vectors in a M-dimensional local basis of the tangent space $\mathcal{T}_{\boldsymbol{x}}\mathbb{S}^M$. To do so, we define a basis $\boldsymbol{B}_{\boldsymbol{p}} = [\boldsymbol{b}_{\boldsymbol{p}}^1 \dots \boldsymbol{b}_{\boldsymbol{p}}^M]$ composed of M orthonormal tangent vectors $\boldsymbol{b}_{\boldsymbol{p}}^i \in \mathbb{R}^{M+1}$. Tangent vectors $\boldsymbol{v}_{\boldsymbol{p}}$ in the local basis and their counterparts $\boldsymbol{v}$ in the ambient basis are consequently related as,

$$\boldsymbol{v}_{\boldsymbol{p}} = \boldsymbol{B}_{\boldsymbol{p}}^\top \boldsymbol{v} \qquad \text{and} \qquad \boldsymbol{v} = \boldsymbol{B}_{\boldsymbol{p}} \boldsymbol{v}_{\boldsymbol{p}}. \tag{24}$$

In the WGPs, we use the former change of basis to define the output of $\text{Log}_{b(\boldsymbol{x})}(\boldsymbol{q})$ locally in the tangent space $\mathcal{T}_{b(\boldsymbol{x})}\mathbb{S}^M$. The latter is then used to express the posterior mean of the Euclidean GP, computed in the local basis, in the ambient basis before projecting it onto the manifold with the exponential map $\text{Exp}_{b(\boldsymbol{x})}(\boldsymbol{q})$. Next, we discuss the choice of basis $\boldsymbol{B}_{\boldsymbol{p}}$ for $\mathbb{S}^2$ and $\mathbb{S}^3$.

Tangent basis for $\mathbb{S}^2$: Denoting each point $\boldsymbol{p} \in \mathbb{S}^2$ in coordinates as $\boldsymbol{p} = (x\ y\ z)^\top$, we define the orthonormal basis $\boldsymbol{B}_{\boldsymbol{x}}$ via the reduced QR decomposition,

$$\boldsymbol{A}_{\boldsymbol{x}} = \boldsymbol{B}_{\boldsymbol{x}} \boldsymbol{R}_{\boldsymbol{x}} \qquad \text{with} \qquad \boldsymbol{A}_{\boldsymbol{x}} = \begin{pmatrix} -z & 0 \\ 0 & z \\ x & y \end{pmatrix}. \tag{25}$$

Tangent basis for $\mathbb{S}^3$: Denoting each point $\boldsymbol{p} \in \mathbb{S}^3$ in coordinates as $\boldsymbol{p} = (w\ x\ y\ z)^\top$, we define the orthonormal basis for each tangent space $\mathcal{T}_{b(\boldsymbol{x})}\mathbb{S}^3$ as,

$$\boldsymbol{B}_{\boldsymbol{x}} = \begin{pmatrix} -x & -y & -z \\ w & z & -y \\ -z & w & x \\ y & -x & w \end{pmatrix}. \tag{26}$$

C.2 SPD manifold

The SPD manifold $\mathcal{S}_{++}^M$ is embedded in the space of symmetric matrices Sym^M, which is itself embedded in the Euclidean space. Each point $\boldsymbol{P} \in \mathcal{S}_{++}^M$ is a $M \times M$ symmetric positive-define matrix and each tangent vector $\boldsymbol{V} \in \mathcal{T}_{\boldsymbol{P}}\mathcal{S}_{++}^M$ is expressed as a $M \times M$ symmetric matrix. Principal operations on the SPD manifold are given by Table 5-*right*. When considering WGPs, we express tangent vectors in a $M(M+1)/2$-dimensional local basis of the tangent space $\mathcal{T}_{\boldsymbol{P}}\mathcal{S}_{++}^M$. This is simply achieved by defining tangent vectors $\boldsymbol{v}_{\boldsymbol{P}}$ in the local basis as vectors composed of the diagonal and upper triangular elements of $\boldsymbol{V}$.

D Likelihood of Wrapped Gaussian Processes

A WGP is $\text{Exp}_{b(\cdot)}(f_{\text{E}}(\cdot))$, where $f_{\text{E}} \sim \text{GP}(\boldsymbol{0}, k)$ is a Euclidean GP learned on the tangent spaces $\mathcal{T}_{b(\cdot)}\mathcal{M}$. In other words, the exponential map pushes forward the Euclidean distribution onto the manifold. Therefore, the WGP marginal likelihood must account for the change of volume induced by the exponential map. This is achieved by leveraging the change of variable formula. Specifically, the change of variable formula states that, given a random variable $\boldsymbol{V}$ endowed with the probability function $p(\boldsymbol{v})$, the log-likelihood of $\boldsymbol{Y} = g(\boldsymbol{V})$ at $\boldsymbol{y}$ is expressed as,

$$\log p(\boldsymbol{y}) = \log p(\boldsymbol{v}) - \log \det \frac{\partial g}{\partial \boldsymbol{v}}. \tag{27}$$

Operation	Formula on $\mathbb{S}^M$
$\langle \boldsymbol{u}, \boldsymbol{v} \rangle_{\boldsymbol{x}}$	$\langle \boldsymbol{u}, \boldsymbol{v} \rangle$
$d_{\mathbb{S}^M}(\boldsymbol{x}, \boldsymbol{y})$	$\arccos(\boldsymbol{x}^\intercal \boldsymbol{y})$
$\mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{u})$	$\boldsymbol{x}\cos(\|\boldsymbol{u}\|) + \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}\sin(\|\boldsymbol{u}\|)$
$\mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{u})$	$d_{\mathbb{S}^M}(\boldsymbol{x}, \boldsymbol{y})\, \frac{\boldsymbol{y} - \boldsymbol{x}^\intercal \boldsymbol{y}\, \boldsymbol{x}}{\|\boldsymbol{y} - \boldsymbol{x}^\intercal \boldsymbol{y}\, \boldsymbol{x}\|}$

Operation	Formula on $\mathcal{S}^M_{++}$
$\langle \boldsymbol{U}, \boldsymbol{V} \rangle_{\boldsymbol{X}}$	$\mathrm{tr}\left(\boldsymbol{X}^{-\frac{1}{2}} \boldsymbol{U} \boldsymbol{X}^{-1} \boldsymbol{V} \boldsymbol{X}^{-\frac{1}{2}}\right)$
$d_{\mathcal{S}^M_{++}}(\boldsymbol{X}, \boldsymbol{Y})$	$\|\log(\boldsymbol{X}^{-\frac{1}{2}} \boldsymbol{Y} \boldsymbol{X}^{-\frac{1}{2}})\|_{\mathrm{F}}$
$\mathrm{Exp}_{\boldsymbol{X}}(\boldsymbol{U})$	$\boldsymbol{X}^{\frac{1}{2}} \exp(\boldsymbol{X}^{-\frac{1}{2}} \boldsymbol{U} \boldsymbol{X}^{-\frac{1}{2}}) \boldsymbol{X}^{\frac{1}{2}}$
$\mathrm{Log}_{\boldsymbol{X}}(\boldsymbol{Y})$	$\boldsymbol{X}^{\frac{1}{2}} \log(\boldsymbol{X}^{-\frac{1}{2}} \boldsymbol{Y} \boldsymbol{X}^{-\frac{1}{2}}) \boldsymbol{X}^{\frac{1}{2}}$

Table 5: *Left:* Principal operations on $\mathbb{S}d$, see (Absil et al., 2007) or (Boumal, 2023) for details. *Right:* Principal operations on $\mathcal{S}^M_{++}$ when endowed with the affine-invariant metric, see (Pennec et al., 2006) for details.

In the case of a WGP, $p(\boldsymbol{x})$ is equal to the probability function of the Euclidean GP f_{E} and the function g is the exponential map $\mathrm{Exp}_{b(\cdot)}$. Therefore, as discussed in Section 4.4, the WGP marginal likelihood is,

$$\log p(\boldsymbol{y}|\boldsymbol{x}) = \log \mathcal{N}\big(\boldsymbol{v}; \boldsymbol{0}, \boldsymbol{K}\big) - \log\det\left(\frac{\partial \,\mathrm{Exp}_{b(\boldsymbol{x})}}{\partial \boldsymbol{v}}\right),$$

with tangent vectors $\boldsymbol{v}$. Next, we provide the expression of the term $\log\det\left(\frac{\partial \,\mathrm{Exp}_{b(\boldsymbol{x})}}{\partial \boldsymbol{v}}\right)$ induced by the change of variable for the manifolds used in our experiments.

D.1 Sphere

The exponential map for the sphere manifold $\mathbb{S}^M$ is given by,

$$\mathrm{Exp}_{\boldsymbol{p}}(\boldsymbol{v}) = \boldsymbol{p}\cos(\|\boldsymbol{v}\|) + \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}\sin(\|\boldsymbol{v}\|), \tag{28}$$

leading to the change of variable correction

$$\log\det\left(\frac{\partial \,\mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{v})}{\partial \boldsymbol{v}}\right) = (M-1)\log\left(\frac{\sin^2(||\boldsymbol{v}||)}{||\boldsymbol{v}||^2}\right). \tag{29}$$

D.2 SPD manifold

The exponential map for the SPD manifold $\mathcal{S}^M_{++}$ endowed with the affine-invariant Riemannian metric (Pennec et al., 2006) is given by,

$$\mathrm{Exp}_{\boldsymbol{P}}(\boldsymbol{V}) = \boldsymbol{X}^{\frac{1}{2}} \exp(\boldsymbol{X}^{-\frac{1}{2}} \boldsymbol{V} \boldsymbol{X}^{-\frac{1}{2}}) \boldsymbol{X}^{\frac{1}{2}}, \tag{30}$$

where exp denotes the matrix exponential. In our experiments, we consider the basepoints of the WGP to be multiples of the identity matrix, i.e., $\boldsymbol{P} = b(\boldsymbol{X}) = a\boldsymbol{I}$. In this case, the exponential map simplifies as,

$$\mathrm{Exp}_{a\boldsymbol{I}}(\boldsymbol{V}) = a\exp(a^{-1}\boldsymbol{V}), \tag{31}$$

leading to the change of variable correction

$$\log\det\left(\frac{\partial \,\mathrm{Exp}_{a\boldsymbol{I}}(\boldsymbol{V})}{\partial \boldsymbol{V}}\right) = \exp(a^{-1}\boldsymbol{V}). \tag{32}$$

E Experimental Details

We implemented all approaches using `GPyTorch` (Gardner et al., 2018b). We used the `Geomstats` implementation of the exponential and logarithmic maps (Miolane et al., 2020), with minor modifications for the vectorization process discussed in Sec. 4.3. To compute geodesics in latent spaces we used `StochMan` (Detlefsen et al., 2021), and the robotic simulations rely on the `Python Robotics Toolbox` (Corke and Haviland, 2021). When back-constrained models are necessary, the Riemannian kernels are implemented using the `GeometricKernels` Python package (Mostowsky et al., 2024).

We used SE kernels $k^{\boldsymbol{x}}$ for all models and constant basepoint functions $b(\boldsymbol{x}) = \boldsymbol{p}$ for WGPLVM and RIEMANN^2 , with $\boldsymbol{p} = (1, 0, \ldots, 0)^\mathsf{T}$ for $\mathbb{S}^M$ and $\boldsymbol{p} = \boldsymbol{I}$ for $\mathcal{S}^M_{++}$. The tangent space vectors $\boldsymbol{v}_i = \mathrm{Log}_{b(\boldsymbol{x}_i)}(\boldsymbol{y}_i)$ were centered in order to build a zero-mean Euclidean GP f_E. The latent variables were initialized using PCA. The latent variables and GP parameters were optimized by minimizing the marginal likelihood for 1000 iterations using Adam (Kingma and Ba, 2015). Data and experiment-specific parameters are detailed next.

E.1 Illustrative example on $\mathbb{R}^2 \times \mathbb{S}^2$

The dataset of this experiment is composed of 6 trajectories of 200 datapoints. Each trajectory traces a J shape in $\mathbb{R}^2$ and a C shape on the $\mathbb{S}^2$. All models are augmented with a GPDM latent prior with dynamic kernel defined as a SE kernel with learnable lengthscale and variance. The learning rate of all models is fixed at 0.025. The geodesics are computed by discretizing the 2D latent manifold into a 50×50 graph and computing the shortest path on the obtained graph via classical algorithms using `StochMan` (Detlefsen et al., 2021).

E.2 Robot motion synthesis on $\mathbb{R}^3 \times \mathbb{S}^3$

The dataset of this experiment is composed of 6 trajectories of varying length (60 to 85 datapoints) for a total of 607 datapoints. We use a Gamma prior with concentration $\alpha = 2$ and rate $\beta = 2$ for the GP kernel lengthscale of all models. We augment the models with back constraints. For GPLVM and pGPLVM, we use a Euclidean SE back-constraints kernel $k^{\mathcal{M}} = k^{\mathbb{R}^7}$ with lengthscale $\theta = 0.2$ and variance $\sigma^2 = 1$. For WGPLVM and RIEMANN^2 , we use a back-constraints kernel defined as the product $k^{\mathcal{M}} = k^{\mathbb{R}^3} k^{\mathbb{S}^3}$. The Euclidean kernel $k^{\mathbb{R}^3}$ is a SE kernel with lengthscale $\theta = 0.2$ and variance $\sigma^2 = 1$. The sphere kernel $k^{\mathbb{S}^3}$ is a sphere SE kernel formulated as in (Borovitskiy et al., 2020) with lengthscale $\theta = 0.2$ and variance $\sigma^2 = 1$. The learning rate of all models is fixed at 0.05. For all models, geodesics are parametrized by cubic splines, whose parameters are optimized to minimize the curve energy using `StochMan` (Detlefsen et al., 2021). Specifically, we approximate the geodesics by cubic splines $c \approx \omega_\lambda(\boldsymbol{z}_c)$, with $\boldsymbol{z}_c = \{\boldsymbol{z}_{c_0}, \ldots, \boldsymbol{z}_{c_N}\}$, where $\boldsymbol{z}_{c_n} \in \mathbb{R}^Q$ is a vector defining a control point of the spline over the latent space. Given N control points, $N - 1$ cubic polynomials ω_{λ_i} with coefficients $\lambda_{i,0}$, $\lambda_{i,1}$, $\lambda_{i,2}$, $\lambda_{i,3}$ have to be estimated to minimize the curve energy.

E.3 Manipulability learning in $\mathbb{R}^2 \times \mathcal{S}^2_{++}$

The dataset of this experiment is composed of 6 trajectories of 100 datapoints. Each trajectory traces a C shape in $\mathbb{R}^2$. The manipulability profiles correspond to the manipulability of a planar robot whose end-effector follows this C shape. We use a Gamma prior with concentration $\alpha = 2$ and rate $\beta = 2$ for the WGP kernel lengthscale of WGPLVM and RIEMANN^2 . All models are augmented with a GPDM latent prior and with back constraints. The GPDM dynamic kernel is defined as a SE kernel with learnable lengthscale and variance for all models. For GPLVM and pGPLVM, we use a Euclidean SE back-constraints kernel $k^{\mathcal{M}} = k^{\mathbb{R}^7}$ with lengthscale $\theta = 0.2$ and variance $\sigma^2 = 1$. For WGPLVM and RIEMANN^2 , we use a back-constraints kernel defined as the product $k^{\mathcal{M}} = k^{\mathbb{R}^2} k^{\mathcal{S}^2_{++}}$. The Euclidean kernel $k^{\mathbb{R}^2}$ is a SE kernel with lengthscale $\theta = 0.5$ and variance $\sigma^2 = 1$. The SPD kernel $k^{\mathcal{S}^2_{++}}$ is a SPD SE kernel formulated as in (Azangulov et al., 2023) with lengthscale $\theta = 0.5$ and variance $\sigma^2 = 1$. The learning rate of all models is fixed at 0.025. For all models, geodesics are parametrized by cubic splines, whose parameters are optimized to minimize the curve energy using `StochMan` (Detlefsen et al., 2021), similar to the previous experiment.

E.4 Brain connectomes in $\mathcal{S}^{15}_{++}$

We consider resting-state functional brain connectome data from the "1200 Subjects release" of the Human Connectome Project (Van Essen et al., 2013). Specifically, we consider the 200 first subjects out of the sub-dataset "R 812", which includes rs-fMRI data. We choose the parcellation of the brain with $N = 15$ regions and use the nodetime series provided in the dataset. Specifically, for each subject, we use 15 time series corresponding to the activation of each of the 15 brain regions over time. We build the parcellated connectome for each subject as the 15×15 covariance matrix corresponding to the correlations between nodes over the time serie. Each connectome is represented as a point in $\mathcal{S}^{15}_{++}$. The learning rate of all models is fixed at 0.03. For all models, geodesics are parametrized by cubic splines, whose parameters are optimized to minimize the curve energy using `StochMan` (Detlefsen et al., 2021).

E.5 Dynamic time warping distance

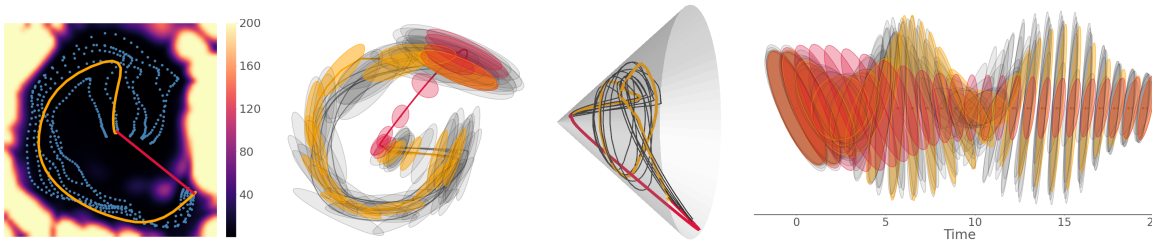
We use dynamic time warping distance (DTWD) as the established quantitative measure of reconstruction accuracy of a decoded geodesic w.r.t. a demonstrated trajectory, assuming equal initial conditions. The DTWD is defined as,

$$\mathrm{DTWD}(\tau_x, \tau_{x'}) = \sum_{j \in l(\tau_{x'})} \min_{i \in l(\tau_x)} \left(d(\tau_{x_i}, \tau_{x'_j}) \right) + \sum_{i \in l(\tau_x)} \min_{j \in l(\tau_{x'})} \left(d(\tau_{x_i}, \tau_{x'_j}) \right),$$

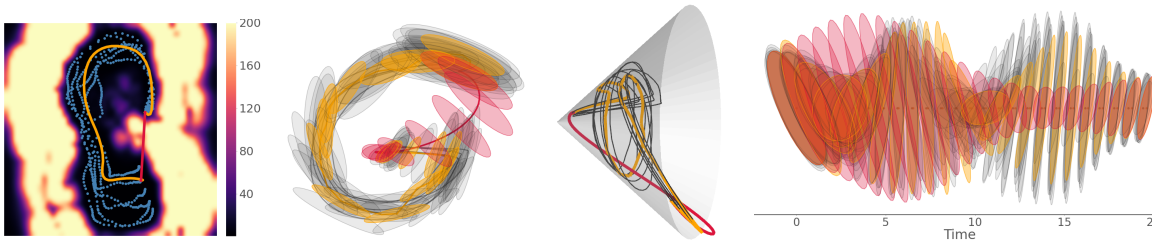
where τ_x and $\tau_{x'}$ are two trajectories (e.g. the decoded geodesic and a demonstration trajectory), d is a distance function (e.g. Euclidean distance), and $l(\tau)$ is the length of trajectory τ.

F Additional Results

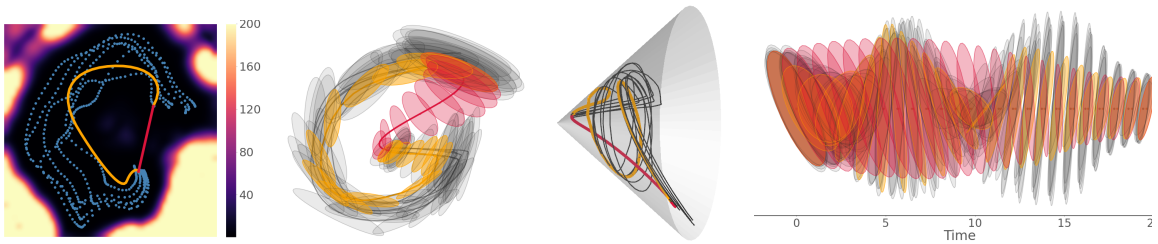
Figure 8 complements the results of Section 5.3 by showing the latent spaces learned with GPLVM, pGPLVM, WGPLVM, and RIEMANN^2 with both independent and correlated output dimensions. As discussed in Section 5.3 and shown in Table 3, RIEMANN^2 with correlated output dimensions achieve the best results overall as it accounts for the correlations between positions and manipulability along the trajectories. This is particularly visible in the *right-most* panel of Figures 8a and 8b, where the decoded manipulability profile of RIEMANN^2 with correlated output matches the demonstrations more than the manipulability profile obtained via independent output dimensions. We observe similar patterns for the pGPLVM of Figures 8c and 8d, even if these models do not account for the geometry of SPD data.



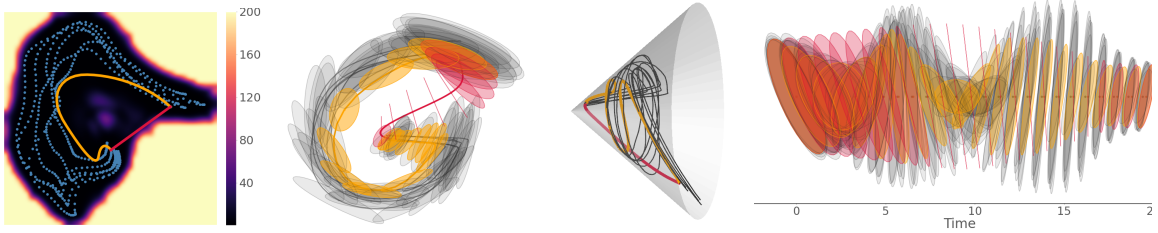
(a) WPGLVM and RIEMANN^2 with correlated output dimensions.



(b) WPGLVM and RIEMANN^2 with independent output dimensions.



(c) GPLVM and pGPLVM with correlated output dimensions.



(d) GPLVM and pGPLVM with independent output dimensions.

Figure 8: $\mathbb{R}^2 \times \mathcal{S}^2_{++}$: From *left* to *right*: Latent variables (●) with magnification factor of the pullback metrics, demonstrations (—,⬭) and reconstructions depicted as curves and ellipsoids in $\mathbb{R}^2$, on the manifold $\mathcal{S}^2_{++}$, and as ellipsoids over time. One Euclidean (—) and one Riemannian (—) geodesic are depicted for each model.