

On Probabilistic Pullback Metrics for Latent Hyperbolic Manifolds

Luis Augenstein¹ Noémie Jaquier^{1,2} Tamim Asfour¹ Leonel Rozo³
¹ Karlsruhe Institute of Technology ² KTH Royal Institute of Technology
³ Bosch Center for Artificial Intelligence

Abstract

Probabilistic Latent Variable Models (LVMs) excel at modeling complex, high-dimensional data through lower-dimensional representations. Recent advances show that equipping these latent representations with a Riemannian metric unlocks geometry-aware distances and shortest paths that comply with the underlying data structure. This paper focuses on hyperbolic embeddings, a particularly suitable choice for modeling hierarchical relationships. Previous approaches relying on hyperbolic geodesics for interpolating the latent space often generate paths crossing low-data regions, leading to highly uncertain predictions. Instead, we propose augmenting the hyperbolic manifold with a pullback metric to account for distortions introduced by the LVM’s nonlinear mapping and provide a complete development for pullback metrics of Gaussian Process LVMs (GPLVMs). Our experiments demonstrate that geodesics on the pullback metric not only respect the geometry of the hyperbolic latent space but also align with the underlying data distribution, significantly reducing uncertainty in predictions.

1 Introduction

Hyperbolic geometry is particularly useful in fields where data exhibits clear hierarchical structures, such as natural language processing for representing word hierarchies and taxonomies [39, 40], as well as in social network analysis for modeling community structures [31, 17]. Additionally, it finds applications in biology [1], human motion taxonomies [25], and computer vision [28]. However, many datasets in these disciplines are high dimensional and heterogeneous, making Latent Variable Models (LVMs) indispensable. Recently, hyperbolic embeddings gained significant attention due to their ability to capture hierarchical structures inherent in complex high-dimensional data [48, 13, 53]. These embeddings leverage hyperbolic growth to accommodate hierarchical relationships, present in trees or graphs, that are difficult to model in Euclidean spaces [14]. Hyperbolic embeddings excel at preserving the hierarchical relationships of the original data in low-dimensional spaces [47, 41]. This can be achieved by learning latent spaces in which embeddings organize according to the data hierarchy [37, 9], by leveraging a known taxonomy to guide the embedding process [39, 25], by estimating the original data hierarchy via diffusion geometry [35], or by learning a tree on the original data [49].

Despite the potential of hyperbolic embeddings, state-of-the-art techniques operate without accounting for the intrinsic properties of the data. For example, the learned embeddings should be distance-preserving and their distribution should closely match that of the observed data. By doing so, any operation on the hyperbolic latent space complies with the properties of the data

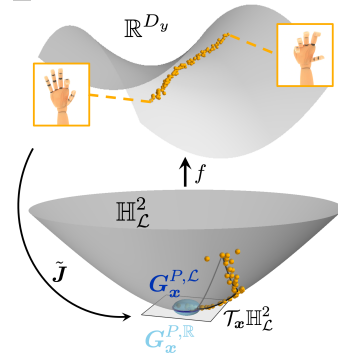


Figure 1: Hyperbolic pullback metric on the tangent space of the Lorentz model. The pullback geodesic (—) follows the data (•) manifold, in contrast to the hyperbolic geodesic (---).

manifold. Instead, existing techniques generate geodesics that often traverse low-density regions of the data space, leading to unrealistic interpolations and inaccurate representations. This problem of integrating the data manifold structure into latent spaces can be tackled via differential geometry [22], by leveraging stochastic pullback metrics [51, 3, 4]. This allows us to introduce uncertainty information into the latent space via a Riemannian metric, enabling the generation of geodesics that faithfully follow the true data distribution. This has been exploited in robot motion generation [5], protein sequencing [15], and data augmentation for medical imaging [12]. However, none of the aforementioned works considered hyperbolic geometry as inductive bias on the latent space.

This paper makes several key contributions to the field of hyperbolic LVMs and Gaussian Process Latent Variable Models (GPLVMs). First, we introduce a general formulation of the Riemannian pullback metric on hyperbolic latent spaces, under the assumption of a stochastic latent-to-ambient mapping (see Fig. 1). Second, we present a novel development of the Riemannian pullback metric tailored to Gaussian Process Hyperbolic Latent Variable Models (GPHLVMs) [25], incorporating appropriate Riemannian projections onto tangent spaces to account for the hyperbolic geometry. Third, we develop the kernel derivatives within this setting, highlighting the limitations of current autodifferentiation techniques when applied to our setting. Finally, we demonstrate the benefits of hyperbolic pullback-based geodesics on four experiments: A proof-of-concept C-shape example, MNIST data interpolation, multi-cellular robot design [16], and human grasps generation [25]. To the best of our knowledge, this paper is the first to consider pullback metrics of non-Euclidean LVMs. It sets the ground for further exploitation of Riemannian geometries in latent spaces.

2 Background

Riemannian geometry: Before delving into the hyperbolic manifold, it is necessary to establish a basic understanding in Riemannian geometry [34]. A Riemannian manifold \mathcal{M} is a smooth manifold equipped with a Riemannian metric, i.e., a smoothly-varying inner product $g_x : \mathcal{T}_x\mathcal{M} \times \mathcal{T}_x\mathcal{M} \rightarrow \mathbb{R}$ over tangent spaces $\mathcal{T}_x\mathcal{M}$. When considering coordinates, the Riemannian metric is represented in matrix form for each $x \in \mathcal{M}$ as $\langle u, v \rangle_x = u^\top G_x v$ with $u, v \in \mathcal{T}_x\mathcal{M}$. The Riemannian metric defines the length of curves in \mathcal{M} , leading to the notion of geodesics, defined as locally length-minimizing curves. To operate with Riemannian manifolds, it is common practice to exploit the Euclidean tangent spaces and the geodesics. The exponential map $\text{Exp}_x(u) = y$ maps $u \in \mathcal{T}_x\mathcal{M}$ to a point $y \in \mathcal{M}$, so that y lies on a geodesic starting at x in the direction u , and such that the geodesic distance $d_{\mathcal{M}}(x, y)$ equals the length of u given by $\|u\|_x = \sqrt{\langle u, u \rangle_x}$. The inverse operation is the logarithmic map $\text{Log}_x(y) = u$. Finally, the parallel transport $\Gamma_{x \rightarrow y}(u) = v$ moves a vector $u \in \mathcal{T}_x\mathcal{M}$ to $\mathcal{T}_y\mathcal{M}$ while preserving the Riemannian inner product.

Optimizing functions defined on manifold requires generalizing the notion of gradient. The Riemannian gradient $\text{grad}_x(f)$ of a function $f : \mathcal{M} \rightarrow \mathbb{R}$ at $x \in \mathcal{M}$ is the unique tangent vector in $\mathcal{T}_x\mathcal{M}$ that satisfies $D_u f(x) = \langle \text{grad}_x f(x), u \rangle_x$, where $D_u f(x)$ is the directional derivative of f in the direction $u \in \mathcal{T}_x\mathcal{M}$ [10, Chap. 3]. The Riemannian Jacobian \tilde{J} of a function $f : \mathcal{M} \rightarrow \mathbb{R}^D$ is composed by the Riemannian gradients for each output dimension, i.e.,

$$\tilde{J} = [\text{grad}_x(f_1) \dots \text{grad}_x(f_D)]^\top. \quad (1)$$

Hyperbolic Manifold: The hyperbolic manifold is the only Riemannian manifold with constant negative curvature [45]. It is commonly represented by either the Poincaré model $\mathbb{H}_{\mathcal{P}}^D$ [43], employing local coordinates within the unit ball, or the Lorentz model $\mathbb{H}_{\mathcal{L}}^D$ [24, 46], using Cartesian coordinates to represent the surface embedded in \mathbb{R}^{D+1} . In this paper we mostly rely on the latter, which is numerically more stable and defined as $\mathbb{H}_{\mathcal{L}}^D = \{x \in \mathbb{R}^{D+1} \mid \langle x, x \rangle_{\mathcal{L}} = -1, x_0 > 0\}$, where $\langle x, y \rangle_{\mathcal{L}} = x^\top G^{\mathcal{L}} y$ is the Lorentzian inner product with metric $G^{\mathcal{L}} = \text{diag}(-1, 1, \dots, 1)$. Further details on the hyperbolic manifold and its operations are provided in App. A.

Gaussian Process Hyperbolic Latent Variable Model (GPHLVM): A GPLVM defines a generative mapping from latent variables $x_n \in \mathbb{R}^{D_x}$ to observations $y_n \in \mathbb{R}^{D_y}$ with $D_x < D_y$ through a non-linear transformation modeled by Gaussian processes (GPs) [33]. A GPLVM assumes that the observations are normally distributed, i.e., $y_{n,d} \sim \mathcal{N}(f_{n,d}, \sigma_y^2)$, and described via a GP and a prior,

$$f_{n,d} \sim \text{GP}(0, k(x_n, x_n)) \quad \text{and} \quad x_n \sim \mathcal{N}(0, I), \quad (2)$$

where $y_{n,d}$ is the d -th dimension of y_n , $k : \mathbb{R}^{D_x} \times \mathbb{R}^{D_x} \rightarrow \mathbb{R}$ is the GP kernel function which measures the similarity between two inputs x_n, x_m , and σ_y^2 is the noise variance.

A Euclidean latent space may not necessarily comply with hierarchical properties of the data, and therefore curved geometries such as the hyperbolic manifold may be preferred. In such cases, the latent variables $\mathbf{x}_n \in \mathbb{H}_{\mathcal{L}}^{D_x}$ live in a hyperbolic space and the generative mapping is defined via a GPLVM [25]. Consequently the kernel function in (2) is replaced by a hyperbolic kernel $k^{\mathbb{H}_{\mathcal{L}}^{D_x}} : \mathbb{H}_{\mathcal{L}}^{D_x} \times \mathbb{H}_{\mathcal{L}}^{D_x} \rightarrow \mathbb{R}$ as introduced shortly. Moreover, the latent variables are assigned a hyperbolic wrapped Gaussian prior $\mathbf{x}_n \sim \mathcal{N}_{\mathbb{H}_{\mathcal{L}}^{D_x}}(\boldsymbol{\mu}_0, \alpha \mathbf{I})$, with $\boldsymbol{\mu}_0 = (1, 0, \dots, 0)^\top$ and α controlling the spread of the latent variables (see App. A for more details). The GPLVM latent variables and hyperparameters are inferred via MAP or variational inference similar as in the Euclidean case.

Hyperbolic Kernels: The SE and Matérn kernels are standard choices when designing GPs [44]. These kernels have been recently generalized to non-Euclidean spaces such as manifolds [8, 26], or graphs [7]. Grigorian and Noguchi [21] demonstrated that, in hyperbolic space, 2- and 3-dimensional SE kernels suffice since higher-dimensional kernels can be expressed as derivatives of these lower-dimensional ones. The 2D and 3D hyperbolic SE kernels are given as,

$$k^{\mathbb{H}_{\mathcal{L}}^2}(\mathbf{x}, \mathbf{z}) = \frac{\tau}{C_\infty} \int_\rho^\infty \frac{s e^{-\frac{s^2}{2\kappa^2}}}{\sqrt{\cosh(s) - \cosh(\rho)}} ds \quad \text{and} \quad k^{\mathbb{H}_{\mathcal{L}}^3}(\mathbf{x}, \mathbf{z}) = \frac{\tau}{C_\infty} \frac{\rho}{\sinh(\rho)} e^{-\frac{\rho^2}{2\kappa^2}}, \quad (3)$$

where $\rho = d_{\mathbb{H}_{\mathcal{L}}^{D_x}}(\mathbf{x}, \mathbf{z})$ is the geodesic distance between $\mathbf{x}, \mathbf{z} \in \mathbb{H}_{\mathcal{L}}^{D_x}$, $\tau, \kappa \in \mathbb{R}_+$ are the kernel variance and lengthscale, and C_∞ is a normalization constant. As no closed form solution is known in the 2D case, the kernel needs to be approximated via a discretization of the integral in (3). We use the positive semidefinite Monte Carlo approximation introduced in [25],

$$k^{\mathbb{H}_{\mathcal{L}}^2}(\mathbf{x}, \mathbf{z}) \approx \frac{\tau}{C_\infty} \frac{1}{L} \sum_{l=1}^L s_l \tanh(\pi s_l) \Phi_l(\mathbf{x}_{\mathcal{P}}, \mathbf{z}_{\mathcal{P}}), \quad (4)$$

where $\Phi_l(\mathbf{x}_{\mathcal{P}}, \mathbf{z}_{\mathcal{P}}) = \phi_l(\mathbf{x}_{\mathcal{P}}) \bar{\phi}_l(\mathbf{z}_{\mathcal{P}})$ is the only component depending on the kernel inputs, $\bar{\phi}_l$ is the complex conjugate of $\phi_l(\mathbf{x}_{\mathcal{P}}) = e^{(1+2s_l i)\langle \mathbf{x}_{\mathcal{P}}, \mathbf{b}_l \rangle}$, $\mathbf{x}_{\mathcal{P}} \in \mathbb{H}_{\mathcal{P}}^{D_x}$ is the Poincaré representation of \mathbf{x} , $\mathbf{b}_l \sim U(\mathbb{S}^1)$ are samples from the unit circle, $s_l \sim e^{-\frac{s^2 \kappa^2}{2}} 1_{[0, \infty]}(s)$ are samples from a truncated Gaussian distribution, and $\langle \mathbf{x}_{\mathcal{P}}, \mathbf{b} \rangle = \frac{1}{2} \log \left(\frac{1 - |\mathbf{x}_{\mathcal{P}}|^2}{|\mathbf{x}_{\mathcal{P}} - \mathbf{b}|^2} \right)$ is the hyperbolic outer product.

Pullback metrics: Consider an immersion $f : \mathcal{S} \rightarrow \mathcal{M}$ from a latent space \mathcal{S} to a Riemannian manifold \mathcal{M} equipped with a Riemannian metric g_y . The immersion f induces a pullback metric $g_x^{\mathcal{P}}$ on \mathcal{S} which, for $\mathbf{x} \in \mathcal{S}$ and $\mathbf{u}, \mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathcal{S}$, which is given by [34, Chap. 2],

$$g_x^{\mathcal{P}}(\mathbf{u}, \mathbf{v}) = g_{f(\mathbf{x})}(df_{\mathbf{x}}(\mathbf{u}), df_{\mathbf{x}}(\mathbf{v})). \quad (5)$$

With coordinates, the pullback metric is given in matrix form by $\mathbf{G}_x^{\mathcal{P}} = \tilde{\mathbf{J}}^\top \mathbf{G}_y \tilde{\mathbf{J}}$, where $\tilde{\mathbf{J}}$ is the Riemannian Jacobian (1) of f at \mathbf{x} . Intuitively, $g_x^{\mathcal{P}}$ evaluates on tangent vectors of $\mathcal{T}_{\mathbf{x}}\mathcal{S}$ by moving them to $\mathcal{T}_{f(\mathbf{x})}\mathcal{M}$ to compute their inner product. For an immersion $f : \mathcal{S} \rightarrow \mathbb{R}^{D_y}$ with Euclidean codomain, i.e., $\mathbf{G}_y = \mathbf{I}$, the pullback metric is $\mathbf{G}_x^{\mathcal{P}} = \tilde{\mathbf{J}}^\top \tilde{\mathbf{J}}$. For Euclidean domains $\mathcal{S} = \mathbb{R}^{D_x}$, $\tilde{\mathbf{J}}$ equals the Euclidean Jacobian $\mathbf{J} = [\frac{\partial f_1}{\partial \mathbf{x}} \dots \frac{\partial f_{D_y}}{\partial \mathbf{x}}]^\top \in \mathbb{R}^{D_y \times D_x}$. In this paper, we define pullback metrics of hyperbolic LVMs by explicitly leveraging immersions $f : \mathbb{H}^{D_x} \rightarrow \mathbb{R}^{D_y}$, where f is a GPLVM.

3 Metrics of Hyperbolic LVMs

Tosi et al. [51] introduced the pullback metric for Euclidean GPLVMs and its use to compute geodesics that adhere to the data distribution. Here, we extend this approach to hyperbolic LVMs, with GPLVMs as a special case. Section 3.1 provides a general formulation of hyperbolic pullback metrics that applies to any hyperbolic LVM defined via an immersion $f : \mathbb{H}_{\mathcal{L}}^{D_x} \rightarrow \mathbb{R}^{D_y}$, provided that the Riemannian Jacobian of f is available and follows a Gaussian distribution. Section 3.2 develops the hyperbolic pullback metric for a specific type of hyperbolic LVM, i.e., the GPLVM, by deriving the probability distribution of the Riemannian Jacobian of the GPLVM's immersion function, which requires kernels and their derivatives. Finally, we compute hyperbolic pullback geodesics in Sec. 3.3.

3.1 A General Hyperbolic Pullback Metric

A deterministic immersion $f : \mathbb{R}^{D_x} \rightarrow \mathbb{R}^{D_y}$ pulls back the metric into the latent space following (5), i.e., $\mathbf{G}_x^{\mathcal{P}, \mathbb{R}} = \mathbf{J}^\top \mathbf{J}$. In the context of LVMs, the immersion f is stochastic. As developed in [51], the

stochastic immersion f induces a distribution over its Jacobian \mathbf{J} , which itself induces a distribution over the pullback metric. As [51], we consider LVMs for which the probability over \mathbf{J} follows a Gaussian distribution. Assuming independent rows $\mathbf{J}_d \in \mathbb{R}^{D_x}$, each with its own mean but shared covariance matrix, the Jacobian distribution is of the form,

$$p(\mathbf{J}) = \prod_{d=1}^{D_y} \mathcal{N}(\mathbf{J}_d \mid \boldsymbol{\mu}_{\mathbf{J}_d}, \boldsymbol{\Sigma}_{\mathbf{J}}). \quad (6)$$

Therefore, the metric tensor $\mathbf{G}_x^{\mathbf{P}, \mathbb{R}}$ follows a non-central Wishart distribution [2],

$$p(\mathbf{G}_x^{\mathbf{P}, \mathbb{R}}) = \mathcal{W}_{D_x}(D_y, \boldsymbol{\Sigma}_{\mathbf{J}}, \mathbb{E}[\mathbf{J}]^T \mathbb{E}[\mathbf{J}]), \quad (7)$$

which we compute the expected metric tensor from,

$$\mathbb{E}[\mathbf{G}_x^{\mathbf{P}, \mathbb{R}}] = \mathbb{E}[\mathbf{J}]^T \mathbb{E}[\mathbf{J}] + D_y \boldsymbol{\Sigma}_{\mathbf{J}}. \quad (8)$$

While the hyperbolic case follows a similar strategy, special care is required to ensure that the Jacobian rows lie on appropriate tangent spaces. Hyperbolic LVMs define a stochastic mapping $f : \mathbb{H}_{\mathcal{L}}^{D_x} \rightarrow \mathbb{R}^{D_y}$, whose Jacobian is formed by the Riemannian gradients of f as in (1). As for Riemannian submanifolds [10], each Riemannian gradient $\text{grad}_x(f_d)$ equals the orthogonal projection of the Euclidean gradient $\frac{\partial f_d}{\partial \mathbf{x}} \in \mathbb{R}^{D_x+1}$ onto $\mathcal{T}_x \mathbb{H}_{\mathcal{L}}^{D_x}$,

$$\text{grad}_x(f_d) = \text{proj}_x \left(\frac{\partial f_d}{\partial \mathbf{x}} \right) = \mathbf{P}_x \frac{\partial f_d}{\partial \mathbf{x}}, \quad (9)$$

with $\mathbf{P}_x = \mathbf{G}^{\mathcal{L}} + \mathbf{x}\mathbf{x}^T$. Therefore, the Riemannian Jacobian $\tilde{\mathbf{J}}$ is computed from the Euclidean Jacobian \mathbf{J} as $\tilde{\mathbf{J}} = [\mathbf{P}_x \frac{\partial f_1}{\partial \mathbf{x}} \dots \mathbf{P}_x \frac{\partial f_{D_y}}{\partial \mathbf{x}}]^T = \mathbf{J} \mathbf{P}_x^T$. Assuming independent rows $\tilde{\mathbf{J}}_d$ as in the Euclidean case, the Jacobian distribution is computed from (6) as [6, Chap. 2.3],

$$p(\tilde{\mathbf{J}}) = \prod_{d=1}^{D_y} \mathcal{N}(\tilde{\mathbf{J}}_d \mid \mathbf{P}_x \boldsymbol{\mu}_{\mathbf{J}_d}, \mathbf{P}_x \boldsymbol{\Sigma}_{\mathbf{J}} \mathbf{P}_x^T). \quad (10)$$

Similarly to $\mathbf{G}_x^{\mathbf{P}, \mathbb{R}}$ in (7), the metric tensor $\mathbf{G}_x^{\mathbf{P}, \mathcal{L}}$ follows a non-central Wishart distribution [2],

$$p(\mathbf{G}_x^{\mathbf{P}, \mathcal{L}}) = \mathcal{W}_{D_x}(D_y, \tilde{\boldsymbol{\Sigma}}_{\mathbf{J}}, \mathbb{E}[\tilde{\mathbf{J}}]^T \mathbb{E}[\tilde{\mathbf{J}}]), \quad (11)$$

with $\tilde{\boldsymbol{\Sigma}}_{\mathbf{J}} = \mathbf{P}_x \boldsymbol{\Sigma}_{\mathbf{J}} \mathbf{P}_x^T$, leading to the expected metric,

$$\mathbb{E}[\mathbf{G}_x^{\mathbf{P}, \mathcal{L}}] = \mathbb{E}[\tilde{\mathbf{J}}]^T \mathbb{E}[\tilde{\mathbf{J}}] + D_y \tilde{\boldsymbol{\Sigma}}_{\mathbf{J}} = \mathbf{P}_x (\mathbb{E}[\mathbf{J}]^T \mathbb{E}[\mathbf{J}] + D_y \boldsymbol{\Sigma}_{\mathbf{J}}) \mathbf{P}_x^T. \quad (12)$$

Intuitively, the pullback metric $\mathbf{G}_x^{\mathbf{P}, \mathcal{L}}$ is the orthogonal projection of the Euclidean pullback metric $\mathbf{G}_x^{\mathbf{P}, \mathbb{R}}$ onto the tangent space $\mathcal{T}_x \mathbb{H}_{\mathcal{L}}^{D_x}$ (see Fig. 1). It relates to the hyperbolic metric $\mathbf{G}^{\mathcal{L}}$ via the projection \mathbf{P}_x , which is orthogonal with respect to the Lorentzian inner product. Note that the presented approach allows us to obtain pullback metrics for all hyperbolic LVMs where the Jacobian \mathbf{J} of f follows a Gaussian distribution, including GPHLVMs [25] and hyperbolic VAEs [38, 37].

3.2 The GPHLVM Pullback Metric

Here, we consider the case where the mapping f is defined as a GPHLVM. We derive the distribution of the Riemannian Jacobian, which we then use to compute the expected GPHLVM pullback metric $\mathbf{G}_x^{\mathbf{P}, \mathcal{L}}$. As for GPLVMs [51], the joint distribution of the Euclidean Jacobian \mathbf{J} and observations $\mathbf{Y} = [\mathbf{y}_1 \dots \mathbf{y}_N]^T \in \mathbb{R}^{N \times D_y}$ is given as,

$$\mathcal{N} \left(\begin{bmatrix} \mathbf{Y}_d \\ \mathbf{J}_d \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{K}_X + \sigma_y^2 \mathbf{I}_N & \partial k(\mathbf{X}, \mathbf{x}^*) \\ \partial k(\mathbf{x}^*, \mathbf{X}) & \partial^2 k(\mathbf{x}^*, \mathbf{x}^*) \end{bmatrix} \right), \quad (13)$$

where $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_N]^T \in \mathbb{R}^{N \times (D_x+1)}$ with $\mathbf{x}_n \in \mathbb{H}_{\mathcal{L}}^{D_x}$, $\mathbf{K}_X \in \mathbb{R}^{N \times N}$ is the corresponding kernel matrix, \mathbf{Y}_d denotes the d -th row of \mathbf{Y} , and we rely on the hyperbolic kernel derivatives,

$$\partial k(\mathbf{X}, \mathbf{x}^*) = \frac{\partial}{\partial \mathbf{z}} k(\mathbf{X}, \mathbf{z}) \Big|_{\mathbf{z}=\mathbf{x}^*} \quad \text{and} \quad \partial^2 k(\mathbf{x}^*, \mathbf{x}^*) = \frac{\partial^2}{\partial \mathbf{z} \partial \mathbf{x}} k(\mathbf{x}, \mathbf{z}) \Big|_{\mathbf{x}=\mathbf{z}=\mathbf{x}^*}. \quad (14)$$

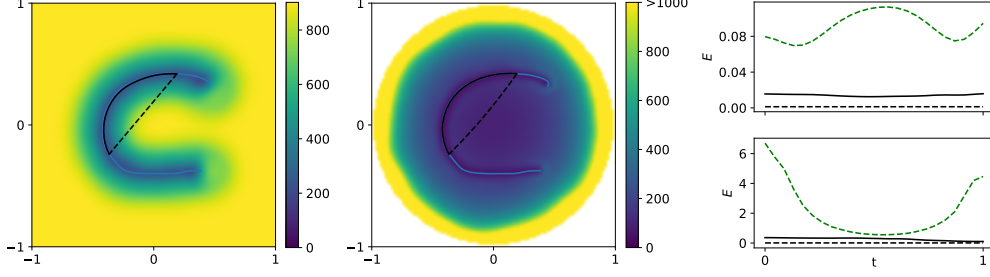


Figure 2: *Left and middle*: Euclidean and hyperbolic pullback metrics on an embedded C-shape trajectory (—) with base manifold (---) and pullback (—) geodesics. *Right*: Curve energy along the geodesics on the Euclidean (top) and hyperbolic (bottom) cases. These include the base manifold geodesic (---), the base manifold geodesic with energy (17) evaluated using the pullback metric (-.-), and the pullback geodesic (—).

While in the Euclidean case the kernel derivatives are straightforward, the hyperbolic setting is more challenging, as will be discussed in Sec. 4. Conditioning on the observations $\mathbf{Y}_d \in \mathbb{R}^N$ results in the probability distribution of the Jacobian at \mathbf{x}^* [6, Chap 2.3],

$$p(\mathbf{J} \mid \mathbf{Y}, \mathbf{X}, \mathbf{x}_*) = \prod_{d=1}^{D_y} \mathcal{N}(\mathbf{J}_d \mid \boldsymbol{\mu}_{\mathbf{J}_d}, \boldsymbol{\Sigma}_{\mathbf{J}}) \text{ with } \boldsymbol{\mu}_{\mathbf{J}_d} = \mathbf{S}_J \mathbf{Y}_d, \boldsymbol{\Sigma}_{\mathbf{J}} = \partial^2 k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{S}_J \partial k(\mathbf{X}, \mathbf{x}^*), \quad (15)$$

and $\mathbf{S}_J = \partial k(\mathbf{x}^*, \mathbf{X})(\mathbf{K}_X + \sigma_y^2 \mathbf{I}_N)^{-1}$. The distribution of the Riemannian Jacobian $\tilde{\mathbf{J}}$ is then obtained from (15) by using (10). The expected pullback metric is finally computed using (12) as,

$$\mathbb{E}[\mathbf{G}_{\mathbf{x}^*}^{\mathbf{P}, \mathcal{L}}] = \mathbf{P}_{\mathbf{x}^*} (\boldsymbol{\mu}_J^\top \boldsymbol{\mu}_J + D_y \boldsymbol{\Sigma}_J) \mathbf{P}_{\mathbf{x}^*}^\top, \quad \text{with } \boldsymbol{\mu}_J = [\boldsymbol{\mu}_{\mathbf{J}_1} \dots \boldsymbol{\mu}_{\mathbf{J}_{D_y}}]^\top \in \mathbb{R}^{D_y \times D_x}. \quad (16)$$

3.3 Hyperbolic Pullback Geodesics

Vanilla geodesics on hyperbolic latent spaces naturally adhere to the hyperbolic geometry. However, they do not account for the underlying data structure and can traverse sparse data regions. To overcome this, we leverage pullback metrics on the hyperbolic manifold to design geodesics that comply with the geometry of both the hyperbolic space and the data distribution (see Figs. 1 and 2). Unlike hyperbolic geodesics that have a closed-form solution, hyperbolic pullback geodesics require solving an optimization problem. Specifically, pullback geodesics are computed by minimizing the curve length, or equivalently the curve energy E with respect to the pullback metric. Considering a discretized geodesic composed by a set of M points $\mathbf{x}_i \in \mathbb{H}_{\mathcal{L}}^{D_x}$, this boils down to minimize,

$$E = \sum_{i=0}^{M-2} \mathbf{v}_i^\top \mathbf{G}_{\mathbf{x}_i}^{\mathbf{P}, \mathcal{L}} \mathbf{v}_i, \quad \text{with } \mathbf{v}_i = \text{Log}_{\mathbf{x}_i}(\mathbf{x}_{i+1}). \quad (17)$$

While it is possible to iteratively optimize the curve points \mathbf{x}_i directly, this often leads to uneven spacing among them. This issue can be addressed either by using a parametric curve on the manifold [20] and optimize its parameters instead of the points directly, or by incorporating the spline energy as a regularization factor in the optimization process [23]. We follow the latter. The spline energy is $E_{\text{spline}} \approx \sum_{i=1}^{M-2} d_{\mathbb{H}_{\mathcal{L}}^{D_x}}(\mathbf{x}_i, \bar{\mathbf{x}}_i)^2$, where $\bar{\mathbf{x}}_i = \text{Exp}_{\mathbf{x}_{i-1}}\left(\frac{1}{2} \text{Log}_{\mathbf{x}_{i-1}}(\mathbf{x}_{i+1})\right)$ is the geodesic midpoint between \mathbf{x}_{i-1} and \mathbf{x}_{i+1} . The final optimization problem is,

$$\min_{\mathbf{x}_0, \dots, \mathbf{x}_{M-1}} E + \lambda E_{\text{spline}}, \quad (18)$$

with λ weighting the influence of the regularization. As the optimization parameters are Riemannian, we leverage Riemannian optimizers such as Riemannian Adam [11] to optimize (17). The computation of pullback geodesics is summarized in Algorithm 1 in App. C.

4 Hyperbolic Kernel Derivatives

Here we discuss the challenges of computing derivatives for hyperbolic SE kernels for which standard autodiff tools cannot be used directly. Table 1 shows that autodiff-based kernel derivatives computed with PyTorch [42] are approximately five times slower than analytic implementations in the 2D case. In the 3D case, autodiff fails to compute the derivatives entirely. To overcome these issues, we derive analytic kernel derivatives and focus our analysis on the 2D and 3D cases. This choice is motivated

by the increased capacity of the hyperbolic manifold to embed hierarchical data in low-dimensional spaces, coupled with the computational efficiency and ease of visualization of 2D and 3D latent spaces. Note that the forthcoming derivations could serve as a basis for higher-dimensional hyperbolic kernels defined as derivatives of 2D and 3D kernels (3) [21].

Table 1: Average computation times in seconds over 100 runs, each evaluating the pullback metric $\mathbf{G}_{\mathbf{x}}^{\mathbb{P},\mathcal{L}}$ and its derivatives at a random point \mathbf{x} .

	$k^{\mathbb{H}^2_{\mathcal{L}}}(L = 3000)$ (4)	$k^{\mathbb{H}^3_{\mathcal{L}}}$ (3)
PyTorch	0.83 ± 0.06	—
Analytic	0.16 ± 0.02	0.06 ± 0.01

2D Hyperbolic SE Kernel Derivatives: We compute the 2D hyperbolic SE kernel $k^{\mathbb{H}^2_{\mathcal{L}}}(\mathbf{x}, \mathbf{z})$ via the Monte Carlo approximation (4). The pullback metric tensor $\mathbf{G}_{\mathbf{x}^*}^{\mathbb{P},\mathcal{L}}$ is computed in (16) using the kernel derivatives $\frac{\partial k(\mathbf{x}, \mathbf{z})}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{x}^*}$ and $\frac{\partial^2 k(\mathbf{x}, \mathbf{z})}{\partial \mathbf{z} \partial \mathbf{x}}|_{\mathbf{x}=\mathbf{z}=\mathbf{x}^*}$ (14). The only part of the kernel (4) that depends on the inputs \mathbf{x}, \mathbf{z} is the function $\Phi_l(\mathbf{x}, \mathbf{z})$, whose derivatives are given in App. B.1. Minimizing the curve energy (17) to compute pullback geodesics additionally requires the derivative $\frac{\partial}{\partial \mathbf{x}^*} \mathbf{G}_{\mathbf{x}^*}^{\mathbb{P},\mathcal{L}}$. This, in turn, requires the derivative of the Jacobian of the covariance matrix $\frac{\partial \Sigma_{\mathbf{J}}}{\partial \mathbf{x}^*}$, which depends on the kernel derivatives $\frac{\partial^3 k(\mathbf{x}, \mathbf{z})}{\partial \mathbf{x} \partial \mathbf{z} \partial \mathbf{x}}|_{\mathbf{x}=\mathbf{z}=\mathbf{x}^*}$, and $\frac{\partial^3 k(\mathbf{x}, \mathbf{z})}{\partial \mathbf{z}^2 \partial \mathbf{x}}|_{\mathbf{x}=\mathbf{z}=\mathbf{x}^*}$. Again, the kernel derivatives are determined by the derivatives of $\Phi_l(\mathbf{x}, \mathbf{z})$. The complete derivation of all derivatives is provided in App. B.1.

3D Hyperbolic SE Kernel Derivatives: The 3D hyperbolic SE kernel is given by (3). As in the 2D case, to obtain the expected pullback metric tensor (16), we compute the first two kernel derivatives $\frac{\partial k(\mathbf{x}, \mathbf{z})}{\partial \mathbf{x}}|_{\mathbf{x}=\mathbf{x}^*}$ and $\frac{\partial^2 k(\mathbf{x}, \mathbf{z})}{\partial \mathbf{z} \partial \mathbf{x}}|_{\mathbf{x}=\mathbf{z}=\mathbf{x}^*}$. As derived in App. B.2, these derivatives depend on the function $g(u) = \left(\frac{2\rho^2}{\nu s^2} - \frac{1}{s^2} - \frac{u\rho}{s^3} \right)$ with $\rho = d_{\mathbb{H}^D_{\mathcal{L}}}(\mathbf{x}, \mathbf{z})$, $u = \langle \mathbf{x}, \mathbf{z} \rangle_{\mathcal{L}}$, and $s = \sqrt{u^2 - 1}$. The function $g(u)$ is essential to understand why standard automatic differentiation tools fail to compute the 3D kernel derivatives. For equal inputs $\mathbf{x} = \mathbf{z}$, the inner product u approaches -1 , while the distance $\rho = d_{\mathbb{H}^D_{\mathcal{L}}}(\mathbf{x}, \mathbf{z})$ and variable s converge to 0. This limit is analytically well behaved, i.e., $\lim_{u \rightarrow -1^-} g(u) = \frac{2}{\nu} + \frac{1}{3}$. However, automatic differentiation tools fail to compute the kernel derivatives for equal inputs as they do not cancel out 0-approaching terms. For example, we have,

$$g(u) = \frac{2\rho^2}{\nu s^2} - \frac{1}{s^2} - \frac{u\rho}{s^3} \xrightarrow[\mathbf{x}=\mathbf{z}]{\text{autodiff}} \frac{0}{0} - \frac{1}{0} - \frac{0}{0} = \text{NaN}, \quad (19)$$

where divisions by 0 lead to undefined values (NaN). While derivatives for equal kernel inputs $\mathbf{x} = \mathbf{z}$ may be less relevant when training a GPLVM as different latent points rarely become so close, they are essential for computing the Jacobian covariance matrix $\Sigma_{\mathbf{J}}$ in (15). Although symbolic differentiation libraries could provide a solution, they tend to be too slow for practical purposes. Instead, we address this issue by computing the necessary derivatives and their analytical limits manually when the inputs \mathbf{x}, \mathbf{z} are closer than a predefined threshold. Complete derivations are provided in App. B.2.

5 Experiments

We test the proposed pullback metric in four distinct experiments to demonstrate that (1) the hyperbolic pullback metric augments the hyperbolic metric with the distortions introduced by the GPLVM’s nonlinear mapping; and (2) the hyperbolic pullback geodesics adhere to the data distribution, leading to low uncertainty model predictions. Note that a direct comparison against other state-of-the-art hyperbolic LVMs [37, 9] is not feasible at this time, as pullback metrics were not derived yet for these models. Additional experimental details are provided in App. D.

C-shape: The first experiment serves as a proof of concept to visualize and compare the Euclidean pullback metric $\mathbf{G}^{\mathbb{P},\mathbb{R}}$ from a GPLVM and the hyperbolic pullback metric $\mathbf{G}^{\mathbb{P},\mathcal{L}}$ from a GPLVM. We design a dataset of 2D C-shape points as both latent variables and observations. In the hyperbolic case, the latent variables and observations are encoded in the Lorentz model, i.e., $\mathbf{x}, \mathbf{y} \in \mathbb{H}^2_{\mathcal{L}}$, but visualized in the Poincaré model for a more intuitive understanding. We fully specify the LVMs by setting the variance, length scale, and noise variance as $(\tau, \kappa, \sigma_y^2) = (0.7, 0.15, 0.69)$.

Fig. 2 displays the Euclidean and hyperbolic pullback metric volumes $\sqrt{\det(\mathbf{G}^{\mathbb{P},\mathbb{R}})}$ and $\sqrt{\det(\mathbf{G}^{\mathbb{P},\mathcal{L}})}$. In the hyperbolic case, the pullback metric is a 3×3 matrix lying in the 2D tangent space $\mathcal{T}_{\mathbf{x}^*} \mathbb{H}^2_{\mathcal{L}}$. Therefore, one of its eigenvalues is always zero, which we exclude to effectively visualize the volume. As shown in Fig. 2-left, the Euclidean pullback metric volume is small near the C-shape data and increases away from it, until a maximum value is reached. The hyperbolic pullback metric volume is also low nearby the data and additionally follows the hyperbolic geometry: It is low near the origin

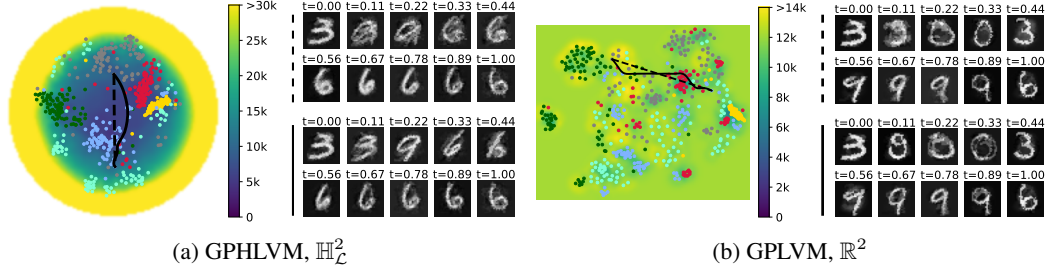


Figure 3: *Left*: Embeddings of a subset of the MNIST dataset with digits 0 (●), 1 (●), 2 (●), 3 (●), 6 (●), and 9 (●). The background color represents the pullback metric volume. The base manifold (---) and pullback (—) geodesics interpolate between a 3 and a 6. *Right*: Ten samples along the decoded geodesics in image space.

and increases when moving outwards until it becomes infinite at the boundary of the unit circle (see Fig. 2-middle). Overall, the hyperbolic pullback metric effectively integrates the properties of both the hyperbolic manifold and the data manifold.

Fig. 2 also depicts geodesics generated with and without the pullback metric on both latent manifolds and the curve energy (17) along the geodesics. In both cases, the pullback geodesic closely adheres to the training data, thus highlighting the successful adaptation of the pullback approach to hyperbolic spaces. However, a key difference lies in the curve energy cost of traversing the center, which is substantially lower in the hyperbolic space. Note that both the base manifold geodesics and the pullback geodesics exhibit constant energy, confirming that the optimized curves are true geodesics on their respective manifolds. For comparison, we also show the curve energy of the base manifold geodesic, but evaluated using the pullback metric tensor. The non-constant energy indicates that the base manifold geodesic is not a true geodesic under the pullback metric.

MNIST Digits: The natural clustering of MNIST images can be viewed as a hierarchy whose nodes are each of the 10 digit classes. Following this point of view, Mathieu et al. [37] showed that hyperbolic LVMs are better suited than their Euclidean counterparts for embedding MNIST images. Similarly to previous works [3, 27, 32], we here explore the interpolation of handwritten digits from a subset of the MNIST dataset. We embed 600 vectorized 28×28 images of 6 classes into 2D hyperbolic and Euclidean latent spaces using GPHLVM and GPLVM, and then use the induced pullback metrics.

Fig. 3 shows the learned latent spaces along with a base manifold and a pullback geodesic interpolating between a digit 3 and a digit 6. The decoded geodesics show the result of the interpolation in the image space. As shown in Fig. 3a, the GPHLVM embeddings tend to be grouped by digits. As expected, the GPHLVM predictions collapse to the non-informative GP mean in regions with sparse data, resulting in blurry images. This is particularly evident along the hyperbolic geodesic for $t = [0.11, 0.44]$. In contrast, the hyperbolic pullback geodesic tends to avoid sparse data regions by bending towards the red cluster (digit 9), leading to less blurry predictions overall. As shown in Fig. 3b, the GPLVM embeddings are not as clearly organized as in the hyperbolic latent space and form several groups per digit. As such, despite that the Euclidean pullback geodesic also adheres to the data distribution, the resulting interpolation oscillates between digits. A quantitative comparison of the GPHLVM and GPLVM likelihood values confirms that a hyperbolic geometry is better suited for this dataset (see Table 2). Moreover, the prediction uncertainty values confirm that pullback metrics better captures the data manifold geometry for both models. This validates the effectiveness of hyperbolic pullback geodesics, which thus may be used, e.g., to measure manifold-aware latent distances and probabilities.

Multi-cellular Robot Design: We build the coarse-to-fine framework for designing multi-cellular robots using hyperbolic embeddings as introduced by Dong et al. [16]. Each robot is composed of a 5×5 grid of cells, where each cell can be horizontally actuated (orange), vertically actuated (blue), rigid (black), soft (gray), or empty (white). The design process begins with a fully horizontally or vertically actuated robot and incrementally introduces changes, forming a hierarchical structure that refines from coarse to fine architectures. This hierarchy can then be queried to find suitable robots for specific tasks. Dong et al. [16] showed that their multi-cellular robot design approach in hyperbolic space outperformed its Euclidean counterpart in most of the considered design tasks.

The main goal of this experiment is to leverage pullback geodesics as a data augmentation mechanism, so that we can design new robot architectures by decoding geodesics that follow the pattern of previously-designed robots. To do so, we embed existing robot designs into 2D latent spaces using

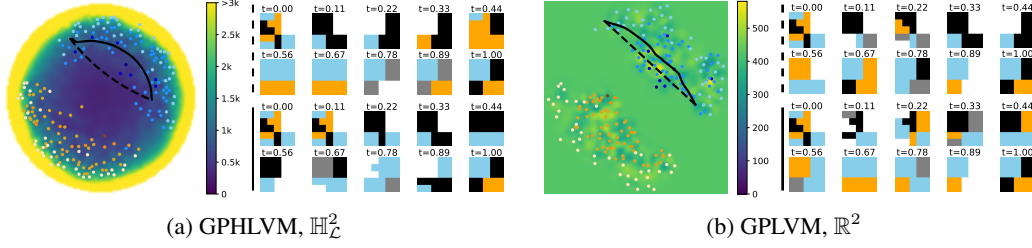


Figure 4: *Left*: Embeddings of multi-cellular robots from coarse (darker tone) to fine (lighter tone). The embeddings form two clusters originating from an all-vertically-actuated robot (•) and an all-horizontally-actuated robot (•). *Right*: Ten samples along the decoded base manifold (---) and pullback (—) geodesics.

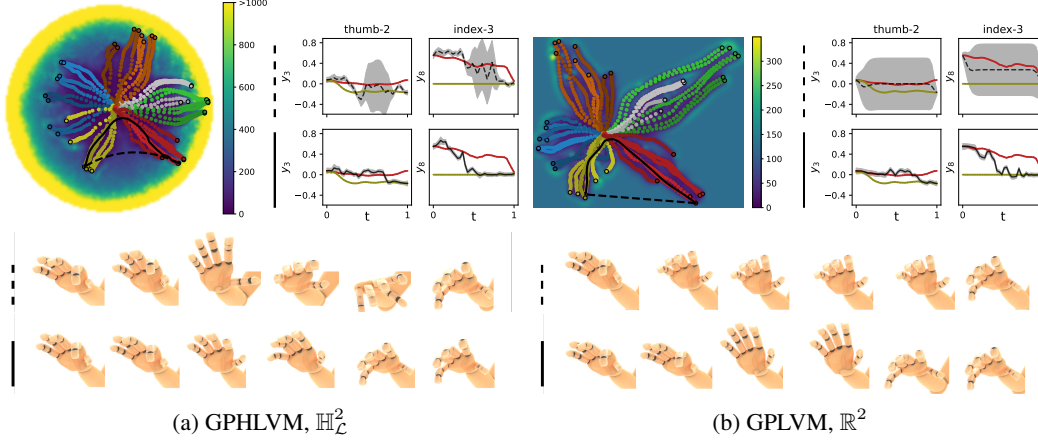


Figure 5: *Top left*: Embeddings of hand grasps colored according to their corresponding grasp class. The background color represents the pullback metric volume. The base manifold (---) and pullback (—) geodesics correspond to a transition from a ring (•) to a spherical grasp (•). *Top right*: Time-series plots of 2 dimensions of the joint space showing the mean of the decoded geodesics with their uncertainty as a gray envelope. A training trajectory to the spherical grasp (—) and a reversed training trajectory from the ring grasp (—) are included for reference. *Bottom*: Generated hand trajectories from the decoded geodesics.

GPLVM and GPLVM. These models are trained with a taxonomy graph-distance prior in the form of a stress loss as in [25], to preserve the hierarchical structure of the designs in the latent space (see App. D.3 for details). Then, we compute the pullback metric and optimize geodesics to interpolate between existing robot designs, generating novel ones. Fig. 4 shows the latent space and the mean predictions along both the base manifold and pullback geodesics for each model. As shown in Table 2, GPLVM achieves a lower stress than GPLVM, indicating that the hyperbolic embeddings better preserve the original hierarchy. For both models, the base manifold geodesics cross sparse data regions, while the pullback geodesics adhere to the data support resulting in low-uncertainty predictions (see Fig. 4 and Table 2). The superior hierarchy preservation in hyperbolic space yields improved transitions between designs, with robot shapes and cell types evolving smoothly along the decoded geodesic (see Fig. 4a). In contrast, the decoded Euclidean geodesic leads to abrupt design changes (see Fig. 4b). This suggests that hyperbolic pullback geodesics can assist robot design processes.

Hand Grasps Generation: Jaquier et al. [25] showed that GPLVMs outperform GPLVMs to learn hand grasp embeddings that comply with the hierarchical structure of human motion taxonomies. Here, we build on this model and explore the use of pullback geodesics as a motion generation mechanism aimed at creating new motions that transition from one grasp to another. We consider the hand grasp taxonomy of [50], which organizes common grasps types within a hierarchical taxonomy tree. We use a dataset from the KIT whole-body motion database [36] consisting of 38 motions of 19 grasp types obtained from recordings of humans grasping different objects. Each motion corresponds to a subject reaching out to grasp an object from an initial resting pose and results in a trajectory $\mathbf{Y} \in \mathbb{R}^{\tilde{N} \times 24}$ representing the temporal evolution of the 24 degrees of freedom of the wrist and fingers.

We train GPLVMs and GPLVMs augmented with a taxonomy graph-distance prior [25] and a dynamic prior akin to [52] (see App. D.4 for details). Fig. 5 shows the resulting 2D hyperbolic and Euclidean latent spaces. Fig. 7 in App. D.5 depicts the 3D latent spaces. All training trajectories start

Table 2: Marginal likelihood (MNIST), average stress (multi-cellular robots, hand grasps) and prediction uncertainty along decoded geodesics (all). The likelihood and the stress are compared across geometries. The prediction uncertainty is compared between the base manifold and pullback geodesic for each model.

Experiment	Model	Likelihood \uparrow / Stress \downarrow	Prediction Uncertainty ($\times 100$) \downarrow	
			Base Manifold	Pullback
MNIST Digits	GPLVM \mathbb{R}^2	-226.23	9.10 ± 4.28	7.95 ± 3.47
	GPHLVM $\mathbb{H}_{\mathcal{L}}^2$	-207.14	7.31 ± 4.74	5.14 ± 2.21
Multi-cellular Robots	GPLVM \mathbb{R}^2	0.73 ± 1.05	3.16 ± 2.77	2.55 ± 2.02
	GPHLVM $\mathbb{H}_{\mathcal{L}}^2$	0.58 ± 0.88	4.14 ± 4.65	0.97 ± 1.80
Hand Grasps	GPLVM \mathbb{R}^2	0.18 ± 0.47	5.24 ± 1.99	0.04 ± 0.02
	GPHLVM $\mathbb{H}_{\mathcal{L}}^2$	0.10 ± 0.26	1.50 ± 2.27	0.04 ± 0.02
	GPLVM \mathbb{R}^3	0.11 ± 0.21	0.30 ± 0.36	0.01 ± 0.01
	GPHLVM $\mathbb{H}_{\mathcal{L}}^3$	0.07 ± 0.10	1.29 ± 1.50	0.03 ± 0.03

near the latent space origin, corresponding to the hand’s initial resting pose, and progress outward until the final grasp. Table 2 shows that the hyperbolic models better capture the taxonomy structure than their Euclidean counterparts, consistent with [25]. Notice that the relatively high stress variance is due to the fact that the taxonomy graph distances of some grasps classes are better preserved by the embeddings than others. However, the distances are better preserved for all classes in hyperbolic latent spaces compared to Euclidean ones (see also App. D.5 and Fig. 8).

Fig. 5 shows the base manifold and pullback geodesics interpolating from a ring to a spherical grasp, alongside the decoded hand motions. While the base manifold geodesic crosses empty regions in the latent space, the pullback geodesic adheres closely to the data support. As both geodesics start and end with the same grasps, their initial and final predictions match. However, the decoded base manifold geodesic shows high uncertainty in the middle part of the motion (see Figs. 5a-5b *top right*). This is due to the GP reverting to its non-informative mean in regions with sparse data, where the GP prior dominates the posterior, leading to maximum uncertainty. Consequently, the obtained hand motions display large deviations from the initial ring and final spherical grasps. In contrast, as shown in Fig. 5 and Table 2, the decoded pullback geodesic shows a much lower uncertainty, as the geodesic stays closer to the data support. Moreover, the hand motions obtained from the decoded pullback geodesics produce more realistic transitions than those from the base manifold geodesics, with the hyperbolic pullback geodesics arguably exhibiting the smallest deviations from the start and end grasps. Similar results are obtained with 3D latent spaces, see App. D.5. Despite the low uncertainty, the motions predicted from 2D latent spaces sometimes lack smoothness due to the fact that nearby latent points do not always correspond to equally similar hand configurations. This is alleviated in the 3D latent space, which provides more volume to better accommodate the latent embeddings.

6 Conclusion

This paper advances the field of hyperbolic LVMs by augmenting the hyperbolic manifold with a Riemannian pullback metric, combining in a principled manner the hyperbolic geometry with the geometry of the data distribution. By minimizing the curve energy on the pullback metric, we computed geodesics that adhere to the data manifold in the hyperbolic latent space. To do so, we addressed the limitations of current auto-differentiation techniques by providing analytical solutions for the hyperbolic SE kernel derivatives. Via multiple experiments, we demonstrated that hyperbolic pullback geodesics outperform both hyperbolic geodesics and Euclidean pullback geodesics.

It is worth noting that the benefits of pullback geodesics are most evident when the data exhibits smooth transitions. They become less effective when the data is inherently comprised of distinct clusters, as the pullback geodesics cannot cross high-energy regions, i.e., the data manifold boundaries. This paper focused on 2D and 3D hyperbolic latent spaces as they are easier to visualize and computationally less expensive than higher-dimensional latent spaces, while benefiting from the increased capacity of hyperbolic spaces to embed hierarchical data. Future work will investigate extensions to higher-dimensional hyperbolic latent spaces, which require to deal with more complex expressions for hyperbolic kernels. Additionally, the need for computing manual kernel derivatives is unsatisfactory. Future work will explore alternative autodifferentiation techniques built on KeOps [19] to overcome this practical issue. Finally, the approximation of the 2D hyperbolic SE kernel is computationally expensive. Performance could potentially be increased by exploring different sampling strategies, e.g., by sampling from a Rayleigh distribution rather than from a Gaussian.

References

- [1] Gregorio Alanis-Lobato, Pablo Mier, and Miguel Andrade-Navarro. The latent geometry of the human protein interaction network. *Bioinformatics*, 34(16):2826–2834, 2018. 1
- [2] T.W. Anderson. The non-central wishart distribution and certain problems of multivariate statistics. *The Annals of Mathematical Statistics*, 17(4):409–431, 1946. 4
- [3] Georgios Arvanitidis, Lars K. Hansen, and Søren Hauberg. Latent space oddity: on the curvature of deep generative models. In *Intl. Conf. on Learning Representations (ICLR)*, 2018. 2, 7
- [4] Georgios Arvanitidis, Søren Hauberg, and Bernhard Schölkopf. Geometrically enriched latent spaces. In *Intl. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2021. 2
- [5] Hadi Beik-Mohammadi, Søren Hauberg, Georgios Arvanitidis, Gerhard Neumann, and Leonel Rozo. Learning Riemannian manifolds for geodesic motion skills. In *Robotics: Science and Systems (RSS)*, 2021. 2
- [6] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, January 2006. 4, 5
- [7] Viacheslav Borovitskiy, Iskander Azangulov, Alexander Terenin, Peter Mostowsky, Marc Deisenroth, and Nicolas Durrande. Matérn Gaussian processes on graphs. In *Intl. Conf. on Artificial Intelligence and Statistics (AISTATS)*, pages 2593–2601, 2021. 3
- [8] Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc P. Deisenroth. Matérn Gaussian processes on Riemannian manifolds. In *Neural Information Processing Systems (NeurIPS)*, pages 12426–12437, 2020. 3
- [9] Joey Bose, Ariella Smofsky, Renjie Liao, Prakash Panangaden, and Will Hamilton. Latent variable modelling with hyperbolic normalizing flows. In *International Conference on Machine Learning (ICML)*, pages 1045–1055, 2020. 1, 6
- [10] Nicolas Boumal. *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023. 2, 4
- [11] Gary Bécigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. In *International Conference on Learning Representations (ICLR)*, 2019. 5
- [12] Clément Chadebec, Elina Thibaud-Sutre, Ninon Burgos, and Stéphanie Allasonnière. Data augmentation in high dimensional low sample size setting using a geometry-based variational autoencoder. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):2879–2896, 2023. 2
- [13] Seunghyuk Cho, Juyong Lee, and Dongwoo Kim. Hyperbolic VAE via latent gaussian distributions. In *Neural Information Processing Systems (NeurIPS)*, 2023. 1
- [14] Andrej Cvetkovski and Mark Crovella. Hyperbolic embedding and routing for dynamic graphs. In *IEEE INFOCOM 2009*, pages 1647–1655, 2009. 1
- [15] Nicki S. Detlefsen, Søren Hauberg, and Wouter Boomsma. Learning meaningful representations of protein sequences. *Nature Communications*, 13(1):1914, 2022. 2
- [16] Heng Dong, Junyu Zhang, and Chongjie Zhang. Leveraging hyperbolic embeddings for coarse-to-fine robot design. In *Intl. Conf. on Learning Representations (ICLR)*, 2024. 2, 7, 18
- [17] Lars Doorenbos, Pablo Márquez Neila, Raphael Sznitman, and Pascal Mettes. Hyperbolic random forests. *Transactions on Machine Learning Research*, 2024. 1
- [18] Thomas Feix, Javier Romero, Heinz-Bodo Schmiedmayer, Aaron M. Dollar, and Danica Kragic. The GRASP taxonomy of human grasp types. *IEEE Transactions on Human-Machine Systems*, 46(1):66–77, 2016. 19
- [19] Jean Feydy, Joan Glaunès, Benjamin Charlier, and Michael Bronstein. Fast geometric learning with symbolic matrices. In *Neural Information Processing Systems (NeurIPS)*, volume 33, 2020. 9
- [20] Pierre-Yves Gousenbourger, Chafik Samir, and P.A. Absil. Piecewise-bézier c1 interpolation on riemannian manifolds with application to 2d shape morphing. In *Intl. Conf. on Pattern Recognition*, pages 4086–4091, 2014. 5
- [21] Alexander Grigoryan and Masakazu Noguchi. The heat kernel on hyperbolic space. *Bulletin of the London Mathematical Society*, 30(6):643–650, 1998. 3, 6

- [22] Søren Hauberg. Only Bayes should learn a manifold. *arXiv preprint arXiv:1806.04994*, 2019. 2
- [23] Behrend Heeren, Martin Rumpf, and Benedikt Wirth. Variational time discretization of Riemannian splines. *IMA Journal of Numerical Analysis*, 39(1):61–104, 2018. 5
- [24] H. Jansen. Abbildung der hyperbolischen geometrie auf ein zweischaliges hyperboloid. *Mitteilungen der Mathematischen Gesellschaft in Hamburg*, 4:409–440, 1909. 2
- [25] Noémie Jaquier, Leonel Rozo, Miguel González-Duque, Viacheslav Borovitskiy, and Tamim Asfour. Bringing motion taxonomies to continuous domains via GPLVM on hyperbolic manifolds. In *International Conference on Machine Learning (ICML)*, 2024. 1, 2, 3, 4, 8, 9, 18, 19, 22
- [26] Noémie Jaquier, Viacheslav Borovitskiy, Andrei Smolensky, Alexander Terenin, Tamim Asfour, and Leonel Rozo. Geometry-aware Bayesian optimization in robotics using Riemannian Matérn kernels. In *Conference on Robot Learning (CoRL)*, 2021. 3
- [27] Martin Jørgensen and Søren Hauberg. Isometric Gaussian process latent variable model for dissimilarity data. In *Intl. Conf. on Machine Learning (ICML)*, pages 5127–5136, 2021. 7
- [28] Valentin Khruikov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1
- [29] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009. 14
- [30] Tamara G. Kolda and Brett W. Bader. Tensor Decompositions and Applications. *SIAM Review*, 51(3):455–500, 2009. 15
- [31] Dmitri Krioukov, Fragiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. Hyperbolic geometry of complex networks. *Phys. Rev. E*, 82:036106, 2010. 1
- [32] Vidhi Lalchand, Aditya Ravuri, and Neil D. Lawrence. Generalised GPLVM with stochastic variational inference. In *Intl. Conf. on Artificial Intelligence and Statistics (AISTATS)*, pages 7841–7864, 2022. 7
- [33] Neil D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Neural Information Processing Systems (NeurIPS)*, 2003. 2
- [34] John Lee. *Introduction to Riemannian Manifolds*. Springer, 2nd edition, 2018. 2, 3
- [35] Ya-Wei Eileen Lin, Ronald R. Coifman, Gal Mishne, and Ronen Talmon. Hyperbolic diffusion embedding and distance for hierarchical representation learning. In *Intl. Conf. on Machine Learning (ICML)*, pages 21003–21025, 2023. 1
- [36] Christian Mandery, Ömer Terlemez, Martin Do, Nikolaus Vahrenkamp, and Tamim Asfour. Unifying representations and large-scale whole-body motion databases for studying human motion. *IEEE Trans. on Robotics*, 32(4):796–809, 2016. 8, 19, 20
- [37] Emile Mathieu, Charline Le Lan, Chris J. Maddison, Ryota Tomioka, and Yee Whye Teh. Continuous hierarchical representations with Poincaré variational auto-encoders. In *Neural Information Processing Systems (NeurIPS)*, 2019. 1, 4, 6, 7
- [38] Yoshihiro Nagano, Shoichiro Yamaguchi, Yasuhiro Fujita, and Masanori Koyama. A wrapped normal distribution on hyperbolic space for gradient-based learning. In *Intl. Conf. on Machine Learning (ICML)*, pages 4693–4702, 2019. 4, 13
- [39] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Neural Information Processing Systems (NeurIPS)*, 2017. 1
- [40] Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In *Intl. Conf. on Machine Learning (ICML)*, pages 3779–3788, 2018. 1
- [41] Richard Nock, Ehsan Amid, Frank Nielsen, Alexander Soen, and Manfred K Warmuth. Hyperbolic embeddings of supervised models. In *Neural Information Processing Systems (NeurIPS)*, 2024. 1
- [42] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: an imperative style,

- high-performance deep learning library. In *Neural Information Processing Systems (NeurIPS)*, 2019. 5
- [43] Henri Poincaré. Théorie des groupes fuchsien. *Acta Mathematica*, 1:1 – 62, 1900. 2
 - [44] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. 3
 - [45] John G. Ratcliffe. *Foundations of Hyperbolic Manifolds*. Springer, 3rd edition, 2019. 2
 - [46] William F. Reynolds. Hyperbolic geometry on a hyperboloid. *The American Mathematical Monthly*, 100(5):442–455, 1993. 2
 - [47] Frederic Sala, Chris De Sa, Albert Gu, and Christopher Re. Representation tradeoffs for hyperbolic embeddings. In *Intl. Conf. on Machine Learning (ICML)*, pages 4460–4469, 2018. 1
 - [48] Ondrej Skopek, Octavian-Eugen Ganea, and Gary Bécigneul. Mixed-curvature variational autoencoders. In *Intl. Conf. on Learning Representations (ICLR)*, 2020. 1
 - [49] Rishi Sonthalia and Anna Gilbert. Tree! i am no tree! i am a low dimensional hyperbolic embedding. In *Neural Information Processing Systems (NeurIPS)*, volume 33, pages 845–856, 2020. 1
 - [50] Francesca Stival, Stefano Michieletto, Matteo Cognolato, Enrico Pagello, Henning Müller, and Manfredo Atzori. A quantitative taxonomy of human hand grasps. *Journal of NeuroEngineering and Rehabilitation*, 16(28), 2019. 8, 19
 - [51] Alessandra Tosi, Søren Hauberg, Alfredo Vellido, and Neil D. Lawrence. Metrics for probabilistic geometries. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2014. 2, 3, 4
 - [52] Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):283–298, 2008. 8, 19
 - [53] Menglin Yang, Min Zhou, Rex Ying, Yankai Chen, and Irwin King. Hyperbolic representation learning: Revisiting and advancing. In *Intl. Conf. on Machine Learning (ICML)*, pages 39639–39659, 2023. 1

A Hyperbolic Manifold

The D -dimensional hyperbolic manifold $\mathbb{H}_{\mathcal{L}}^D$ and its tangent space $\mathcal{T}_x \mathbb{H}_{\mathcal{L}}^D$ are given as

$$\mathbb{H}_{\mathcal{L}}^D = \{\mathbf{x} \in \mathbb{R}^{D+1} \mid \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = -1, x_0 > 0\}, \quad (20)$$

$$\mathcal{T}_x \mathbb{H}_{\mathcal{L}}^D = \{\mathbf{u} \in \mathbb{R}^{D+1} \mid \langle \mathbf{u}, \mathbf{x} \rangle_{\mathcal{L}} = 0\}, \quad (21)$$

where $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = \mathbf{x}^\top \mathbf{G}^{\mathcal{L}} \mathbf{y}$ is the Lorentzian inner product with the Lorentzian metric tensor $\mathbf{G}^{\mathcal{L}} = \text{diag}(-1, 1, \dots, 1)$. Note that, since the curvature of the hyperbolic manifold is constant, the metric tensor does not depend on \mathbf{x} as for general manifolds, i.e., $\mathbf{G}_{\mathbf{x}}^{\mathcal{L}} = \mathbf{G}^{\mathcal{L}}$ for all $\mathbf{x} \in \mathbb{H}_{\mathcal{L}}^D$. For the hyperbolic manifold, closed form solutions exist for the standard manifold operations. Specifically, the geodesic distance, exponential map, logarithmic map, parallel transport, and tangent space projection are given as,

$$d_{\mathbb{H}_{\mathcal{L}}^D}(\mathbf{x}, \mathbf{y}) = \text{arccosh}(-\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}), \quad (22)$$

$$\text{Exp}_{\mathbf{x}}(\mathbf{u}) = \cosh(\|\mathbf{u}\|_{\mathcal{L}})\mathbf{x} + \sinh(\|\mathbf{u}\|_{\mathcal{L}})\frac{\mathbf{u}}{\|\mathbf{u}\|_{\mathcal{L}}}, \quad (23)$$

$$\text{Log}_{\mathbf{x}}(\mathbf{y}) = d_{\mathbb{H}_{\mathcal{L}}^D}(\mathbf{x}, \mathbf{y}) \frac{\mathbf{y} + \langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} \mathbf{x}}{\sqrt{\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}^2 - 1}}, \quad (24)$$

$$\Gamma_{\mathbf{x} \rightarrow \mathbf{y}}(\mathbf{u}) = \mathbf{u} + \frac{\langle \mathbf{y}, \mathbf{u} \rangle_{\mathcal{L}}}{1 - \langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}}(\mathbf{x} + \mathbf{y}) \quad (25)$$

$$\text{proj}_{\mathbf{x}}(\mathbf{w}) = \mathbf{P}_{\mathbf{x}} \mathbf{w} = (\mathbf{G}^{\mathcal{L}} + \mathbf{x} \mathbf{x}^\top) \mathbf{w}. \quad (26)$$

These operations are illustrated by Fig. 6.

Working with probabilistic models on Riemannian manifolds requires probability distributions that account for their geometry. Therefore, the GPHLM prior $\mathbf{x}_n \sim \mathcal{N}_{\mathbb{H}_{\mathcal{L}}^D}(\boldsymbol{\mu}_0, \alpha \mathbf{I})$ relies on the hyperbolic wrapped distribution [38], which builds on a Gaussian distribution on the tangent space at the origin $\boldsymbol{\mu}_0 = [1 \ 0 \ \dots \ 0]^\top \in \mathbb{H}_{\mathcal{L}}^D$. Intuitively, the wrapped distribution is constructed as follows: (1) Sample a point $\tilde{\mathbf{v}} \in \mathbb{R}^D$ from the Euclidean Gaussian distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$; (2) Transform $\tilde{\mathbf{v}}$ to an element of the tangent space $\mathcal{T}_{\boldsymbol{\mu}_0} \mathbb{H}_{\mathcal{L}}^D$ at the origin by setting $\mathbf{v} = [0 \ \tilde{\mathbf{v}}]^\top$; (3) Parallel transport \mathbf{v} to the desired mean $\mathbf{u} = \Gamma_{\boldsymbol{\mu}_0 \rightarrow \boldsymbol{\mu}}(\mathbf{v})$; and (4) Project \mathbf{u} onto the hyperbolic space via the exponential map $\mathbf{x} = \text{Exp}_{\boldsymbol{\mu}}(\mathbf{u})$. The resulting probability density function is

$$\mathcal{N}_{\mathbb{H}_{\mathcal{L}}^D}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}(\tilde{\mathbf{v}} \mid \mathbf{0}, \boldsymbol{\Sigma}) \left(\frac{r}{\sinh(r)} \right)^{D-1}, \quad (27)$$

where $\mathbf{u} = \text{Log}_{\boldsymbol{\mu}}(\mathbf{x})$, $\mathbf{v} = \Gamma_{\boldsymbol{\mu} \rightarrow \boldsymbol{\mu}_0}(\mathbf{u})$, and $r = \|\mathbf{u}\|_{\mathcal{L}}$.

As mentioned in the main text, we mostly rely on the Lorentz model $\mathbb{H}_{\mathcal{L}}^D$ in our algorithms, but use the Poincaré model for visualization, as it provides a more intuitive representation. The Poincaré model is defined as

$$\mathbb{H}_{\mathcal{P}}^D = \{\mathbf{x}_{\mathcal{P}} \in \mathbb{R}^D \mid \|\mathbf{x}_{\mathcal{P}}\| < 1\}. \quad (28)$$

It is possible to map points from the Lorentz to the Poincaré model via an isometric mapping $f : \mathbb{H}_{\mathcal{L}}^D \rightarrow \mathbb{H}_{\mathcal{P}}^D$ such that

$$f(\mathbf{x}) = \frac{(x_1, \dots, x_d)^\top}{x_0 + 1}, \quad (29)$$

where $\mathbf{x} \in \mathbb{H}_{\mathcal{L}}^D$ with components x_0, x_1, \dots, x_d . The inverse mapping $f^{-1} : \mathbb{H}_{\mathcal{P}}^D \rightarrow \mathbb{H}_{\mathcal{L}}^D$ is defined as follows

$$f^{-1}(\mathbf{y}) = \frac{(1 + \|\mathbf{y}\|^2, 2y_1, \dots, 2y_d)^\top}{1 - \|\mathbf{y}\|^2}, \quad (30)$$

with $\mathbf{y} \in \mathbb{H}_{\mathcal{P}}^D$ with components y_1, \dots, y_d .

B Hyperbolic Kernel Derivatives

This section provides the complete derivations of the hyperbolic SE kernel derivatives that are necessary to compute the pullback metric tensor and to optimize geodesics on it. Eq. (16) defines the

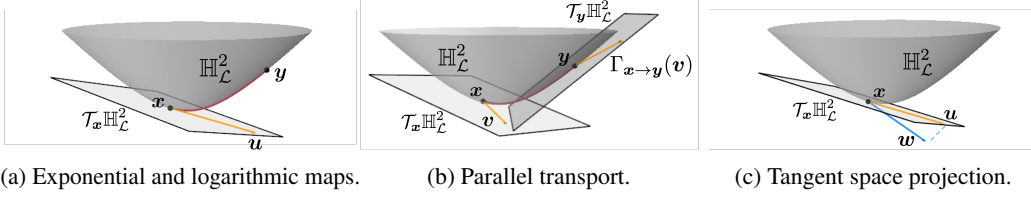


Figure 6: Principal Riemannian operations on the Lorentz model $\mathbb{H}_{\mathcal{L}}^2$. **(a)** The geodesic (—) is the shortest path between the two points x to y on the manifold. Its length is equal to the geodesic distance $d_{\mathbb{H}_{\mathcal{L}}^2}(x, y)$. The vector u (—) lies on the tangent space of x such that $y = \text{Exp}_x(u)$. **(b)** Parallel transport $\Gamma_{x \rightarrow y}(v)$ of the vector v from $\mathcal{T}_x \mathbb{H}_{\mathcal{L}}^2$ to $\mathcal{T}_y \mathbb{H}_{\mathcal{L}}^2$. **(c)** The vector w (—) is projected onto the tangent space of x via the tangent space projection $u = \text{proj}_x(w)$.

pullback metric tensor $G_{x^*}^{\text{P}, \mathcal{L}}$ based on the Jacobian mean μ_J and covariance matrix Σ_J (15), which in turn require the kernel derivatives $\partial k(x^*, \mathbf{X}) = \frac{\partial}{\partial x^*} k(x^*, \mathbf{X}) \in \mathbb{R}^{D_x \times N}$ and $\partial^2 k(x^*, x^*) = \frac{\partial^2}{\partial z \partial x} k(x, z)|_{x=z=x^*} \in \mathbb{R}^{D_x \times D_x}$. Additionally, minimizing the curve energy (17), requires the derivative of the pullback metric tensor, i.e., the derivative of $\mu_J^\top \mu_J + D_y \Sigma_J$, which requires the kernel derivative $\frac{\partial^3}{\partial x \partial z \partial x} k(x, z)$ and $\frac{\partial^3}{\partial z^2 \partial x} k(x, z)$. Specifically, we have

$$\frac{\partial}{\partial x^*} (\mu_J^\top \mu_J + D_y \Sigma_J) = \left[\frac{\partial}{\partial x^*} \mu_J^\top \right] \times_2 \mu_J^\top + \left[\frac{\partial}{\partial x^*} \mu_J \right] \times_1 \mu_J^\top + D_y \left[\frac{\partial}{\partial x^*} \Sigma_J \right], \quad (31)$$

$$\text{with } \frac{\partial}{\partial x^*} \mu_J^\top = \left[\frac{\partial}{\partial x^*} \partial k(x^*, \mathbf{X}) \right] \times_2 \mathbf{Y}^\top (\mathbf{K}_X + \sigma_y^2 \mathbf{I}_N)^{-1}, \quad (32)$$

$$\frac{\partial}{\partial x^*} \Sigma_J = \left[\frac{\partial}{\partial x^*} \partial^2 k(x^*, x^*) \right] - \left[\frac{\partial}{\partial x^*} \partial k(x^*, \mathbf{X}) \right] \times_2 \mathbf{S}_J - \left[\frac{\partial}{\partial x^*} \partial k(\mathbf{X}, x^*) \right] \times_1 \mathbf{S}_J, \quad (33)$$

$$\frac{\partial}{\partial x^*} \partial^2 k(x^*, x^*) = \left[\frac{\partial^3}{\partial x \partial z \partial x} k(x, z) + \frac{\partial^3}{\partial z^2 \partial x} k(x, z) \right] \Big|_{x=z=x^*}, \quad (34)$$

where we have $\mu_J \in \mathbb{R}^{D_y \times D_x}$, $\frac{\partial}{\partial x^*} \mu \in \mathbb{R}^{D_y \times D_x \times D_x}$, $\frac{\partial}{\partial x^*} \partial k(x^*, \mathbf{X}) \in \mathbb{R}^{D_x \times N \times D_x}$, and \times_n denotes the n -th mode tensor product [29]. Next, we provide the details of the derivation of the kernel derivatives for $\mathbb{H}_{\mathcal{L}}^2$ and $\mathbb{H}_{\mathcal{L}}^3$.

B.1 2D Hyperbolic SE Kernel Derivatives

We compute the 2D hyperbolic SE kernel $k_{\mathbb{H}_{\mathcal{L}}^2}(x, z)$ via the Monte Carlo approximation (4). The pullback metric tensor $G_{x^*}^{\text{P}, \mathcal{L}}$ is computed using the kernel derivatives $\frac{\partial}{\partial x} k(x, z)|_{x=x^*}$ and $\frac{\partial^2}{\partial z \partial x} k(x, z)|_{x=z=x^*}$ (14). The only part of the kernel (4) that depends on the inputs x, z is the function $\Phi_l(x, z)$. Therefore, the derivatives of the 2D hyperbolic SE kernel (4) are entirely determined by the derivatives of $\Phi_l(x, z)$, which are given by,

$$\frac{\partial \Phi_l(x, z)}{\partial x} = \left[\frac{\partial \phi_l(x_{\mathcal{P}})}{\partial x} \right] \bar{\phi}_l(z_{\mathcal{P}}) \quad \text{and} \quad \frac{\partial^2 \Phi_l(x, z)}{\partial z \partial x} = \left[\frac{\partial \phi_l(x_{\mathcal{P}})}{\partial x} \right] \left[\frac{\partial \bar{\phi}_l(z_{\mathcal{P}})}{\partial z} \right]^\top. \quad (35)$$

Eq (35) shows that these derivatives in turn are defined by the first and second derivatives of $\phi_l(\mathbf{x}_{\mathcal{P}}) = e^{(1+2s_l i)\langle \mathbf{x}_{\mathcal{P}}, \mathbf{b}_l \rangle}$, which are computed as

$$\frac{\partial \phi_l(\mathbf{x}_{\mathcal{P}})}{\partial \mathbf{x}} = (1 + 2s_l i) \phi_l(\mathbf{x}_{\mathcal{P}}) \left[\frac{\partial}{\partial \mathbf{x}} \mathbf{x}_{\mathcal{P}} \right]^{\top} \left[\frac{\partial}{\partial \mathbf{x}_{\mathcal{P}}} \langle \mathbf{x}_{\mathcal{P}}, \mathbf{b} \rangle \right], \quad (36)$$

$$\frac{\partial^2 \phi_l(\mathbf{x}_{\mathcal{P}})}{\partial \mathbf{x}^2} = (1 + 2s_l i) \left(\left[\frac{\partial}{\partial \mathbf{x}} \mathbf{x}_{\mathcal{P}} \right]^{\top} \left[\frac{\partial}{\partial \mathbf{x}_{\mathcal{P}}} \langle \mathbf{x}_{\mathcal{P}}, \mathbf{b} \rangle \right] \left[\frac{\partial}{\partial \mathbf{x}} \phi_l(\mathbf{x}_{\mathcal{P}}) \right]^{\top} \right. \quad (37)$$

$$\begin{aligned} &+ \phi_l(\mathbf{x}_{\mathcal{P}}) \left[\frac{\partial^2}{\partial \mathbf{x}^2} \mathbf{x}_{\mathcal{P}} \right] \times_1 \left[\frac{\partial}{\partial \mathbf{x}_{\mathcal{P}}} \langle \mathbf{x}_{\mathcal{P}}, \mathbf{b} \rangle \right] \\ &+ \phi_l(\mathbf{x}_{\mathcal{P}}) \left[\frac{\partial}{\partial \mathbf{x}} \mathbf{x}_{\mathcal{P}} \right]^{\top} \left[\frac{\partial}{\partial \mathbf{x}} \frac{\partial}{\partial \mathbf{x}_{\mathcal{P}}} \langle \mathbf{x}_{\mathcal{P}}, \mathbf{b} \rangle \right] \Bigg), \\ &= (1 + 2s_l i) \phi_l(\mathbf{x}_{\mathcal{P}}) \left((1 + 2s_l i) \left[\frac{\partial}{\partial \mathbf{x}} \mathbf{x}_{\mathcal{P}} \right]^{\top} \left[\frac{\partial}{\partial \mathbf{x}_{\mathcal{P}}} \langle \mathbf{x}_{\mathcal{P}}, \mathbf{b} \rangle \right] \left[\frac{\partial}{\partial \mathbf{x}_{\mathcal{P}}} \langle \mathbf{x}_{\mathcal{P}}, \mathbf{b} \rangle \right]^{\top} \left[\frac{\partial}{\partial \mathbf{x}} \mathbf{x}_{\mathcal{P}} \right] \right. \\ &\quad + \left[\frac{\partial^2}{\partial \mathbf{x}^2} \mathbf{x}_{\mathcal{P}} \right] \times_1 \left[\frac{\partial}{\partial \mathbf{x}_{\mathcal{P}}} \langle \mathbf{x}_{\mathcal{P}}, \mathbf{b} \rangle \right] \\ &\quad \left. + \left[\frac{\partial}{\partial \mathbf{x}} \mathbf{x}_{\mathcal{P}} \right]^{\top} \left[\frac{\partial^2}{\partial \mathbf{x}_{\mathcal{P}}^2} \langle \mathbf{x}_{\mathcal{P}}, \mathbf{b} \rangle \right] \left[\frac{\partial}{\partial \mathbf{x}} \mathbf{x}_{\mathcal{P}} \right] \right). \end{aligned} \quad (38)$$

As shown by these expressions, the derivatives of $\phi_l(\mathbf{x}_{\mathcal{P}})$ are defined as functions of the derivatives of the hyperbolic outer product and of the derivatives of the Lorentz to Poincaré mapping. The derivatives of the hyperbolic outer product $\langle \mathbf{x}_{\mathcal{P}}, \mathbf{b} \rangle = \frac{1}{2} \log \left(\frac{1 - |\mathbf{x}_{\mathcal{P}}|^2}{|\mathbf{x}_{\mathcal{P}} - \mathbf{b}|^2} \right)$ are given as,

$$\frac{\partial}{\partial \mathbf{x}_{\mathcal{P}}} \langle \mathbf{x}_{\mathcal{P}}, \mathbf{b} \rangle = -\frac{1}{2} \left(\frac{2\mathbf{x}_{\mathcal{P}}}{1 - |\mathbf{x}_{\mathcal{P}}|^2} + \frac{2(\mathbf{x}_{\mathcal{P}} - \mathbf{b})}{|\mathbf{x}_{\mathcal{P}} - \mathbf{b}|^2} \right) = \frac{\mathbf{x}_{\mathcal{P}}}{|\mathbf{x}_{\mathcal{P}}|^2 - 1} - \frac{\mathbf{x}_{\mathcal{P}} - \mathbf{b}}{|\mathbf{x}_{\mathcal{P}} - \mathbf{b}|^2}, \quad (39)$$

$$\frac{\partial^2}{\partial \mathbf{x}_{\mathcal{P}}^2} \langle \mathbf{x}_{\mathcal{P}}, \mathbf{b} \rangle = \frac{1}{|\mathbf{x}_{\mathcal{P}}|^2 - 1} \mathbf{I}_{D_x} - \frac{2\mathbf{x}_{\mathcal{P}} \mathbf{x}_{\mathcal{P}}^{\top}}{(|\mathbf{x}_{\mathcal{P}}|^2 - 1)^2} - \frac{1}{|\mathbf{x}_{\mathcal{P}} - \mathbf{b}|^2} \mathbf{I}_{D_x} + \frac{2(\mathbf{x}_{\mathcal{P}} - \mathbf{b})(\mathbf{x}_{\mathcal{P}} - \mathbf{b})^{\top}}{|\mathbf{x}_{\mathcal{P}} - \mathbf{b}|^4}. \quad (40)$$

Finally, the derivatives of the Lorentz to Poincaré mapping $\mathbf{x}_{\mathcal{P}} = \frac{1}{1+x_0} [x_1 \dots x_D]^{\top} \in \mathbb{H}_{\mathcal{P}}^{D_x}$ are given as,

$$\left(\frac{\partial}{\partial \mathbf{x}} \mathbf{x}_{\mathcal{P}} \right)_{i,j} = \begin{cases} -\frac{x_{i+1}}{(1+x_0)^2} & j = 0 \\ \frac{1}{1+x_0} & j = i + 1 \\ 0 & \text{otherwise} \end{cases}, \quad \left(\frac{\partial^2}{\partial \mathbf{x}^2} \mathbf{x}_{\mathcal{P}} \right)_{i,j,k} = \begin{cases} \frac{2x_{i+1}}{(1+x_0)^3} & j = k = 0 \\ -\frac{1}{(1+x_0)^2} & j = 0 \text{ and } k = i + 1 \\ -\frac{1}{(1+x_0)^2} & k = 0 \text{ and } j = i + 1 \\ 0 & \text{otherwise} \end{cases}, \quad (41)$$

with $i \in \{0, \dots, D_x - 1\}$, $j, k \in \{0, \dots, D_x\}$, and $D_x = 2$.

Minimizing the curve energy (17) to compute pullback geodesics additionally requires the derivative $\frac{\partial}{\partial \mathbf{x}^*} \mathbf{G}_{\mathbf{x}^*}^{\mathcal{P}, \mathcal{L}}$. This, in turn, requires the derivative of the Jacobian of the covariance matrix $\frac{\partial \Sigma_{\mathcal{I}}}{\partial \mathbf{x}^*}$, which depends on the kernel derivatives $\frac{\partial^3 k(\mathbf{x}, \mathbf{z})}{\partial \mathbf{x} \partial \mathbf{z} \partial \mathbf{x}}|_{\mathbf{x}=\mathbf{z}=\mathbf{x}^*}$, and $\frac{\partial^3 k(\mathbf{x}, \mathbf{z})}{\partial \mathbf{z}^2 \partial \mathbf{x}}|_{\mathbf{x}=\mathbf{z}=\mathbf{x}^*}$. Again, the kernel derivatives are determined by the derivatives of $\Phi_l(\mathbf{x}, \mathbf{z})$,

$$\frac{\partial^3 \Phi_l(\mathbf{x}, \mathbf{z})}{\partial \mathbf{x} \partial \mathbf{z} \partial \mathbf{x}} = \left[\frac{\partial^2 \phi_l(\mathbf{x}_{\mathcal{P}})}{\partial \mathbf{x}^2} \right] \times_2 \left[\frac{\partial \bar{\phi}_l(\mathbf{z}_{\mathcal{P}})}{\partial \mathbf{z}} \right] \quad \text{and} \quad \frac{\partial^3 \Phi_l(\mathbf{x}, \mathbf{z})}{\partial \mathbf{z}^2 \partial \mathbf{x}} = \left[\frac{\partial^2 \bar{\phi}_l(\mathbf{z}_{\mathcal{P}})}{\partial \mathbf{z}^2} \right] \times_1 \left[\frac{\partial \phi_l(\mathbf{x}_{\mathcal{P}})}{\partial \mathbf{x}} \right], \quad (42)$$

where $\mathbf{A} \times_i \mathbf{B}$ refers to n-mode product [30] which multiplies the i -th dimension of \mathbf{A} with the second dimension of \mathbf{B} . If \mathbf{B} is a vector, \mathbf{A} is unsqueezed at dimension i before the multiplication. The matrices $\frac{\partial^2 \phi_l(\mathbf{x}_{\mathcal{P}})}{\partial \mathbf{x}^2}$, $\frac{\partial^2 \phi_l(\mathbf{z}_{\mathcal{P}})}{\partial \mathbf{z}^2} \in \mathbb{R}^{3 \times 3}$ in (42) are therefore interpreted as $\mathbb{R}^{3 \times 1 \times 3}$ and $\mathbb{R}^{1 \times 3 \times 3}$, respectively.

B.2 3D Hyperbolic SE Kernel Derivatives

This section provides the derivations of the 3D hyperbolic SE kernel derivatives. The 3D hyperbolic SE kernel is given by (3). In this section, we simplify its notation and write $k(\mathbf{x}, \mathbf{z}) = \frac{\rho}{s} e^{-\frac{\rho^2}{\nu}}$ by omitting the constant τ/C_∞ , setting $\nu = 2\kappa^2$, and introducing the helper variables $u = \langle \mathbf{x}, \mathbf{z} \rangle_{\mathcal{L}}$ and $s = \sqrt{u^2 - 1}$. Their derivatives are $\frac{\partial u}{\partial \mathbf{x}} = \mathbf{G}^{\mathcal{L}} \mathbf{z}$, $\frac{d\rho}{du} = \frac{-1}{s}$, and $\frac{ds}{du} = \frac{u}{s}$. Using these expressions, the first and second derivatives of the 3D hyperbolic SE kernel $k(\mathbf{x}, \mathbf{y}) = \frac{\rho}{s} e^{-\frac{\rho^2}{\nu}}$ are given as,

$$\begin{aligned} \frac{\partial}{\partial \mathbf{x}} k(\mathbf{x}, \mathbf{z}) &= \left[\frac{\partial u}{\partial \mathbf{x}} \frac{d}{du} \rho \right] \frac{1}{s} e^{-\frac{\rho^2}{\nu}} + \rho \left[\frac{\partial u}{\partial \mathbf{x}} \frac{d}{du} s^{-1} \right] e^{-\frac{\rho^2}{\nu}} + \frac{\rho}{s} \left[\frac{\partial u}{\partial \mathbf{x}} \frac{d\rho}{du} \frac{d}{d\rho} e^{-\frac{\rho^2}{\nu}} \right], \\ &= -\frac{1}{s^2} \mathbf{G}^{\mathcal{L}} \mathbf{z} e^{-\frac{\rho^2}{\nu}} - \frac{\rho u}{s^3} \mathbf{G}^{\mathcal{L}} \mathbf{z} e^{-\frac{\rho^2}{\nu}} + \frac{\rho}{s^2} \mathbf{G}^{\mathcal{L}} \mathbf{z} \frac{2\rho}{\nu} e^{-\frac{\rho^2}{\nu}}, \\ &= g(u) e^{-\frac{\rho^2}{\nu}} \mathbf{G}^{\mathcal{L}} \mathbf{z}, \end{aligned} \quad (43)$$

$$\begin{aligned} \frac{\partial^2}{\partial \mathbf{z} \partial \mathbf{x}} k(\mathbf{x}, \mathbf{z}) &= \mathbf{G}^{\mathcal{L}} \mathbf{z} \left[\frac{\partial u}{\partial \mathbf{z}} \frac{d}{du} g(u) \right] e^{-\frac{\rho^2}{\nu}} + \mathbf{G}^{\mathcal{L}} \mathbf{z} g(u) \left[\frac{\partial u}{\partial \mathbf{z}} \frac{d\rho}{du} \frac{d}{d\rho} e^{-\frac{\rho^2}{\nu}} \right] + g(u) e^{-\frac{\rho^2}{\nu}} \left[\frac{\partial}{\partial \mathbf{z}} \mathbf{G}^{\mathcal{L}} \mathbf{z} \right], \\ &= \left[\frac{d}{du} g(u) \right] e^{-\frac{\rho^2}{\nu}} \mathbf{G}^{\mathcal{L}} \mathbf{z} \mathbf{z}^{\top} \mathbf{G}^{\mathcal{L}} + g(u) \frac{-1}{s} \frac{-2\rho}{\nu} e^{-\frac{\rho^2}{\nu}} \mathbf{G}^{\mathcal{L}} \mathbf{z} \mathbf{z}^{\top} \mathbf{G}^{\mathcal{L}} + g(u) e^{-\frac{\rho^2}{\nu}} \mathbf{G}^{\mathcal{L}}, \\ &= (h(u) \mathbf{G}^{\mathcal{L}} \mathbf{z} \mathbf{z}^{\top} \mathbf{G}^{\mathcal{L}} + g(u) \mathbf{G}^{\mathcal{L}}) e^{-\frac{\rho^2}{\nu}}, \end{aligned} \quad (44)$$

$$\frac{\partial^2}{\partial \mathbf{x} \partial \mathbf{x}} k(\mathbf{x}, \mathbf{z}) = h(u) \mathbf{G}^{\mathcal{L}} \mathbf{z} \mathbf{z}^{\top} \mathbf{G}^{\mathcal{L}} e^{-\frac{\rho^2}{\nu}}, \quad (45)$$

with $g(u) = \left(\frac{2\rho^2}{\nu s^2} - \frac{1}{s^2} - \frac{u\rho}{s^3} \right)$ and $h(u) = \frac{d}{du} g(u) + \frac{g(u)2\rho}{\nu s}$. The helper functions $g(u)$ and $h(u)$ are essential to understand why standard automatic differentiation tools fail to compute the derivatives (43)-(45). For equal inputs $\mathbf{x} = \mathbf{z}$, the inner product u approaches -1 , while the distance $\rho = d_{\mathbb{H}_{\mathcal{L}^x}}(\mathbf{x}, \mathbf{z})$ and variable s converge to 0. This limit is analytically well behaved for the helper functions, e.g., $\lim_{u \rightarrow -1-} g(u) = \frac{2}{\nu} + \frac{1}{3}$. However, automatic differentiation tools fail to compute the kernel derivatives for equal inputs correctly because they do not cancel out 0-approaching terms. For example, we have for $g(u)$,

$$\frac{2\rho^2}{\nu s^2} - \frac{1}{s^2} - \frac{u\rho}{s^3} \xrightarrow[\mathbf{x}=\mathbf{z}]{\text{autodiff}} \frac{0}{0} - \frac{1}{0} - \frac{0}{0} = \text{NaN}.$$

In this scenario, autodiff divides by 0, leading to undefined values (NaN). In other cases, e.g., when training a GPLVM, derivatives for equal kernel inputs $\mathbf{x} = \mathbf{z}$ may not be that relevant because different latent points rarely become that close. However, these derivatives are essential for computing the Jacobian covariance matrix Σ_J in (15). Although symbolic differentiation libraries could provide a solution, they tend to be too slow for practical purposes. To address this issue, we replace the general function definitions with their analytical limits, when u gets close to -1 . This is equivalent to the hyperbolic distance between the two kernel inputs falling below a predefined threshold value $\rho = \text{arccosh}(-u) = d_{\mathbb{H}_{\mathcal{L}^x}}(\mathbf{x}, \mathbf{z}) \leq 1e-4$. The analytical limit expressions for the helper functions are given as

$$\lim_{u \rightarrow -1-} g(u) = \lim_{u \rightarrow -1-} \left(\frac{2\rho^2}{\nu s^2} - \frac{1}{s^2} - \frac{u\rho}{s^3} \right) = \frac{2}{\nu} + \frac{1}{3}, \quad (46)$$

$$\lim_{u \rightarrow -1-} h(u) = \lim_{u \rightarrow -1-} \left(\frac{d}{du} g(u) + \frac{g(u)2\rho}{\nu s} \right) = \frac{4}{\nu^2} + \frac{6}{3\nu} + \frac{4}{15}, \quad (47)$$

$$\lim_{u \rightarrow -1-} q(u) = \lim_{u \rightarrow -1-} \left(\frac{d}{du} h(u) + \frac{h(u)2\rho}{\nu s} \right) = \frac{8}{\nu^3} + \frac{8}{\nu^2} + \frac{14}{5\nu} + \frac{12}{35}. \quad (48)$$

Finally, we need the third-order kernel derivatives to optimize pullback geodesics as in the 2D case. Denoting the second kernel derivative as $\mathbf{K} = \frac{\partial^2 k(\mathbf{x}, \mathbf{z})}{\partial \mathbf{z} \partial \mathbf{x}} \in \mathbb{R}^{4 \times 4}$, the third-order derivatives is obtained by stacking the individual derivatives of each row and column of \mathbf{K} . The derivative of each

Algorithm 1 Computation of hyperbolic pullback geodesics

Input: Start and end points $\mathbf{x}_1, \mathbf{x}_M \in \mathbb{H}_{\mathcal{L}}^{D_x}$ of the pullback geodesic, pullback metric $\mathbf{G}^{\mathcal{P}, \mathcal{L}}$.
Output: Pullback geodesic represented as a set of points $\{\mathbf{x}_i\}_{i=1}^M$ with $\mathbf{x}_i \in \mathbb{H}_{\mathcal{L}}^{D_x}$.
Initialization:
Initialize the pullback geodesic, e.g., as the hyperbolic geodesic from \mathbf{x}_1 to \mathbf{x}_M .
Discretize the pullback geodesic with M points $\{\mathbf{x}_i\}_{i=1}^M$.
Geodesic computation:
repeat
 Compute the curve energy E (17).
 Compute the spline energy E_{spline} .
 $\mathbf{x}_2, \dots, \mathbf{x}_{M-1} \leftarrow \text{RiemannianOptStep}(E + \lambda E_{\text{spline}})$.
until convergence

row \mathbf{K}_i w.r.t. the first input \mathbf{x} and each column \mathbf{K}_j w.r.t the second input \mathbf{z} are given by,

$$\frac{\partial}{\partial \mathbf{z}} \mathbf{K}_j = \frac{\partial}{\partial \mathbf{z}} (h(u) \mathbf{G}^{\mathcal{L}} \mathbf{z} \mathbf{x}_j + g(u) \mathbf{e}_j) \Delta_{1j} e^{-\frac{\rho^2}{\nu}}, \quad (49)$$

$$= \left(\left[\frac{d}{du} h(u) \right] \mathbf{G}^{\mathcal{L}} \mathbf{z} \mathbf{x}^{\top} \mathbf{G}^{\mathcal{L}} \mathbf{x}_j + h(u) \mathbf{G}^{\mathcal{L}} \mathbf{x}_j + \left[\frac{d}{du} g(u) \right] \mathbf{e}_j \mathbf{x}^{\top} \mathbf{G}^{\mathcal{L}} \right) \Delta_{1j} e^{-\frac{\rho^2}{\nu}} \quad (50)$$

$$+ (h(u) \mathbf{G}^{\mathcal{L}} \mathbf{z} \mathbf{x}_j + g(u) \mathbf{e}_j) \Delta_{1l} \frac{2\rho}{\nu s} e^{-\frac{\rho^2}{\nu}} \mathbf{x}^{\top} \mathbf{G}^{\mathcal{L}}, \quad (51)$$

$$= (q(u) \mathbf{G}^{\mathcal{L}} \mathbf{z} \mathbf{x}^{\top} \mathbf{G}^{\mathcal{L}} \mathbf{x}_j + h(u) \mathbf{G}^{\mathcal{L}} \mathbf{x}_j + h(u) \mathbf{e}_j \mathbf{x}^{\top} \mathbf{G}^{\mathcal{L}}) \Delta_{1j} e^{-\frac{\rho^2}{\nu}}, \quad (52)$$

$$\frac{\partial}{\partial \mathbf{x}} \mathbf{K}_i = (q(u) \mathbf{G}^{\mathcal{L}} \mathbf{z} \mathbf{x}^{\top} \mathbf{G}^{\mathcal{L}} \mathbf{z}_i + h(u) \mathbf{G}^{\mathcal{L}} \mathbf{z}_i + h(u) \mathbf{G} \mathbf{z} \mathbf{e}_i^{\top}) \Delta_{1i} e^{-\frac{\rho^2}{\nu}}, \quad (53)$$

where $q(u) = \frac{d}{du} h(u) + \frac{h(u)2\rho}{\nu s}$ and Δ_{ij} represents a modified Kronecker delta equal to -1 for equal indices and to 1 otherwise. Stacking the row and column matrices \mathbf{K}_i and \mathbf{K}_j along the first and second dimension accordingly gives the third kernel derivatives $\frac{\partial \mathbf{K}}{\partial \mathbf{x}}, \frac{\partial \mathbf{K}}{\partial \mathbf{z}} \in \mathbb{R}^{4 \times 4 \times 4}$.

C Hyperbolic Pullback Geodesics

In Sec. 3.3, we described the computation of geodesics according to the hyperbolic pullback metric introduced in Sec. 3.1, 3.2. Algorithm 1 summarizes the computation of a hyperbolic pullback geodesic by minimizing (18) with Riemannian optimization.

D Experimental Details and Additional Results

Apps. D.1-D.4 provide additional details regarding the data and GPHLVM training process for each of the experiments described in Sec. 5. App. D.5 provides additional insights into the hand grasps experiment of Sec. 5, and App. D.6 provides runtimes for the computation of pullback metrics and geodesics.

D.1 C-shape

The proof-of-concept experiment on the C-shape dataset was designed as a minimalist example with a simple setup to visualize and compare Euclidean and hyperbolic pullback metrics. The training data consists of $N = 1000$ two-dimensional points arranged in a C shape and scaled to fit within the unit circle. In the Euclidean case, the datapoints are used as both latent points and observations, i.e., $\mathbf{X} = \mathbf{Y} \in \mathbb{R}^{1000 \times 2}$. In the hyperbolic case, the datapoints can be interpreted as elements of the Poincaré model. We represent each latent point and observation in the Lorentz model using the Poincaré to Lorentz mapping $\mathbf{x}_{\mathcal{L}} = \frac{1}{1-\|\mathbf{x}\|^2} [1 + \|\mathbf{x}\|^2 \quad 2\mathbf{x}^{\top}]^{\top} \in \mathbb{H}_{\mathcal{L}}^2$. We fully specify the LVMs by manually setting the kernel variance, lengthscale, and noise variance of both GPLVM and GPHLVM to $(\tau, \kappa, \sigma_y^2) = (0.7, 0.15, 0.69)$. We used constant zero mean for both models. For the GPHLVM, we used the 2D hyperbolic SE kernel with $L = 3000$ rejection samples.

The start- and end-points of the geodesics shown in Fig. 2 correspond to one of the C-shape latent points. The geodesics are represented by $M = 25$ regularly-spaced points. Geodesics on the Euclidean base manifold correspond to straight lines from \mathbf{x}_0 to \mathbf{x}_{M-1} . In the hyperbolic case, the geodesics on the base manifold are given as $\mathbf{x}_i = \text{Exp}_{\mathbf{x}_0}(t_i \text{Log}_{\mathbf{x}_0}(\mathbf{x}_{M-1}))$ with $t_i = i/(M-1)$ and $i = \{0, \dots, M-1\}$. The geodesics on the base manifolds were used as initialization for the pullback geodesic optimization. We obtain the pullback geodesics by minimizing the combined loss function (18) based on both curve energy and spline energy (see Sec. 3.3). We used Riemannian Adam for 200 steps with a learning rate of 0.005. In this experiment, the curve energy and spline energy were weighted equally, i.e., $\lambda = 1$ in Eq. (17).

We visualize the volume of the pullback metrics on a 110×110 grid of points, filtering out points lying outside the unit circle in the hyperbolic case. In the Euclidean case, we computed the volume at each grid point \mathbf{x}^* by evaluating the determinant $\sqrt{\det(\mathbf{G}_{\mathbf{x}^*}^{\text{P},\mathbb{R}})}$ of the pullback metric tensor $\mathbf{G}^{\text{P},\mathbb{R}}$. In the hyperbolic case, the pullback metric tensor is a 3×3 matrix that lies on the 2-dimensional tangent space, leading to one of its eigenvalues being zero. To address this, we compute the volume $\sqrt{\det(\mathbf{G}_{\mathbf{x}^*}^{\text{P},\mathcal{L}})}$ as the product of the two nonzero eigenvalues.

D.2 MNIST Digits

We use a subset of the MNIST dataset composed of 100 datapoints for each of the classes 0, 1, 2, 3, 6, and 9. Each datapoint is represented as a 28×28 binary image. We optimize the parameters of the GPHLVM using Riemannian Adam for 500 steps with a learning rate of 0.05. We set a Gamma prior with concentration $\alpha = 2$ and rate $\beta = 2$ on the kernel lengthscale and a Gamma prior with concentration $\alpha = 5$ and rate $\beta = 0.8$ on the kernel variance. The hyperbolic embeddings are initialized in the tangent space of the origin μ_0 using PCA and projected onto the manifold using the exponential map. Similarly, we optimize the parameters of the GPLVM using Adam for 500 steps with a learning rate of 0.01. The Euclidean embeddings are initialized using PCA.

D.3 Multi-cellular Robot Design

We consider multi-cellular robots composed of a 5×5 grid of cells. Possible cell types are empty, solid, soft, a horizontal actuator or a vertical actuator. We generate a dataset of 216 multi-cellular robots following the hierarchical approach of [16], which we explain next for completeness. The root nodes of the hierarchy are two coarse robots composed of only vertical and only horizontal actuator cells. A component is defined as a group of cells of the same type, i.e., the root robots have a single component. Children nodes of each robot are designed by dividing large components into 2 smaller components using K-means and by assigning a different type of cell to one of these components. This process is repeated iteratively to obtain robots with up to 8 components. This results in a tree of multi-cellular robots, with coarse-grained robots close to the roots and fine-grained robots at the leaves.

To account for the hierarchical structure associated with the tree structure of the multi-cellular robots, we incorporate graph-based priors in both GPHLVM and GPLVM as proposed in [25]. This is achieved by leveraging the stress loss,

$$\ell_{\text{stress}}(\mathbf{X}) = \sum_{i < j} (\text{d}_{\mathbb{G}}(c_i, c_j) - \text{d}_{\mathcal{M}}(\mathbf{x}_i, \mathbf{x}_j))^2, \quad (54)$$

to match the pairwise latent distances with the tree distances, and where c_i denotes the tree node, $\text{dist}_{\mathbb{G}}$, and $\text{dist}_{\mathcal{M}}$ are the graph distance and the geodesic distance on \mathcal{M} , respectively. Therefore, training the GPHLVM and GPLVM is achieved by solving,

$$\arg\max_{\mathbf{X}} \log p(\mathbf{Y} | \mathbf{X}, \Theta) - \gamma \bar{\ell}_{\text{stress}}(\mathbf{X}), \quad (55)$$

where γ is a parameter balancing the two losses and $\bar{\ell}_{\text{stress}}(\mathbf{X})$ is the stress loss averaged over the embeddings. We set $\gamma = 6000$. For the GPHLVM, we use a Gamma prior with concentration $\alpha = 2$ and rate $\beta = 2$ on the kernel lengthscale. We optimize the parameters of the GPHLVM using Riemannian Adam for 500 steps with a learning rate of 0.05 and the parameters of the GPLVM using Adam for 500 steps with a learning rate of 0.01.

D.4 Hand Grasps Generation

For the hand grasp experiment, we used data from the KIT Whole-Body Motion Database [36]. The dataset consists of 38 trajectories, with motion recordings from two human subjects (IDs 2122 and 2123). Each subject performed 19 different grasp types¹ of the GRASP taxonomy [18]. Each grasp consists of the subject grasping an object from a table, lifting it, and placing it back.

We applied several preprocessing steps to the recorded data before training the GPHLVM: (1) We applied a low-pass filter to remove high-frequency noise; (2) We trimmed the start and end of each trajectory to focus solely on the motion from the initial resting pose to the point where the grasp was completed. Since detecting the exact point of grasp completion is non-trivial, we cut off the trajectories at the moment when the grasped object was first moved by the subject; (3) We subsample the trajectories; and (4) We centered the data to allow for the use of a zero mean function in the GPHLVM. After preprocessing, each trajectory is composed of 30 to 40 data points. Stacking all trajectories together results in a total of $N = 1321$ data points $\mathbf{Y} \in \mathbb{R}^{1321 \times 24}$.

In contrast to the previous experiments, this experiment involves trajectory data. To preserve the trajectory structure during training, we incorporated a dynamics prior $p(\mathbf{X}_{2:N} \mid \mathbf{X}_{1:N-1})$, similar to [52], but using the wrapped Gaussian distribution (27). We also used back constraints $\mathbf{X} = k(\mathbf{Y}, \mathbf{Y})\mathbf{C}$, which allow for a smooth mapping from the observation space to the latent space. The back-constraints kernel is defined as a Euclidean SE kernel with variance $\tau = 1$ and a lengthscale $\kappa = 0.4$. Moreover, each grasp type is identified with a leaf node of the quantitative taxonomy of hand grasps [50]. More specifically, the first point of each trajectory, which corresponds to the resting state, is assigned to the root node c_0 of the taxonomy graph, and the last point to the taxonomy node representing the corresponding grasp type. The taxonomy node c_{14} , for instance, represents the stick grasp. The number of edges between different grasp types in the taxonomy graph defines a distance function, $d_G(c_i, c_j)$. To account for the hierarchical structure associated with the taxonomy, we additionally incorporate graph-based priors in our GPHLVM as proposed by Jaquier et al. [25]. This is achieved by leveraging the stress loss (54) to match the pairwise latent distances with the graph distances. During training, we minimize the stress loss over the first and last latent points of each trajectory along with maximizing the GPHLVM marginal likelihood by optimizing,

$$\operatorname{argmax}_{\mathbf{X}} \log p(\mathbf{Y} \mid \mathbf{X}, \Theta) + 2 \frac{D_y}{D_x} \log p(\mathbf{X}_{2:N} \mid \mathbf{X}_{1:N-1}) - D_y \ell_{\text{stress}}(\mathbf{X}). \quad (56)$$

We performed the optimization using Riemannian Adam for 10000 steps, with a learning rate of 0.001.

As GPLVMs are generally prone to local optima during training, they benefit from a good initialization. Therefore, we initialize the first and last latent points of each trajectory to minimize the stress loss, i.e.,

$$\mathbf{X}_{\text{init}} = \min_{\mathbf{X}} \ell_{\text{stress}}(\mathbf{X}). \quad (57)$$

The optimized first and last points are then connected using a hyperbolic geodesic with the same number of points as the original motion recording. To ensure distinct initialization for each subject, we added random noise to the geodesics.

For the geodesic optimization, we selected the final points of two trajectories for generating a motion from a ring grasp to a spherical grasp. We compared the hyperbolic geodesic with the pullback geodesic. Since the latent space in this experiment captures the taxonomy structure, we incorporated the expected path along the taxonomy nodes into the initialization for the pullback geodesic. Specifically, we concatenated two hyperbolic geodesics: the first connecting the spherical grasp to the origin which represents the root node of the taxonomy graph, and the second connecting the origin to the ring grasp. Starting from this initialization, we applied Riemannian Adam to optimize the geodesic over 200 steps, using a learning rate of 0.005 and a spline energy weighting of $\lambda = 100$.

D.5 Additional Hand Grasps Results

This section provides further insights into the hand grasps experiment described in Sec. 5. First, Fig. 7 shows the 3D hyperbolic and Euclidean latent spaces obtained from the GPHLVM and GPLVM of

¹The three-finger sphere grasp of the GRASP taxonomy [18] is missing from the dataset.

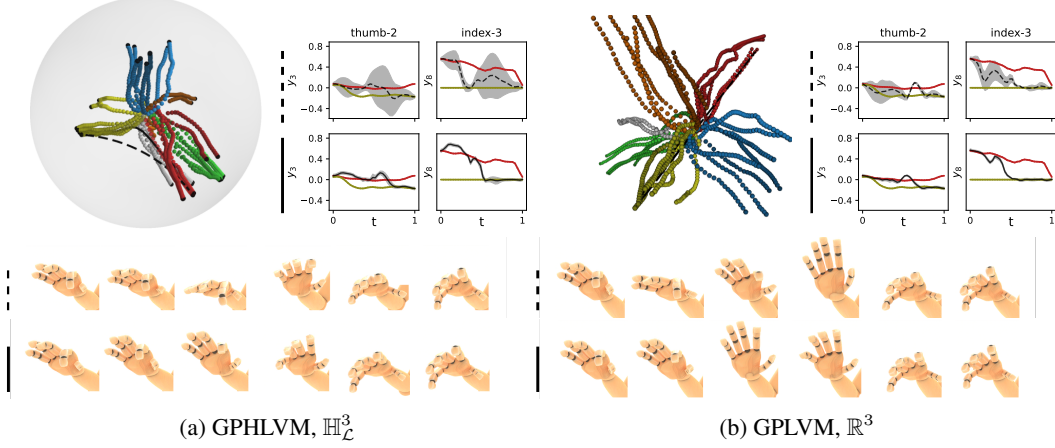


Figure 7: *Top left*: Embeddings of hand grasps colored according to their corresponding grasp class. The background color represents the pullback metric volume. The base manifold (---) and pullback (—) geodesics correspond to a transition from a ring (•) to a spherical grasp (•). *Top right*: Time-series plots of 2 dimensions of the joint space showing the mean of the decoded geodesics with their uncertainty as a gray envelope. A training trajectory to the spherical grasp (—) and a reversed training trajectory from the ring grasp (—) are included for reference. *Bottom*: Generated hand trajectories from the decoded geodesics.

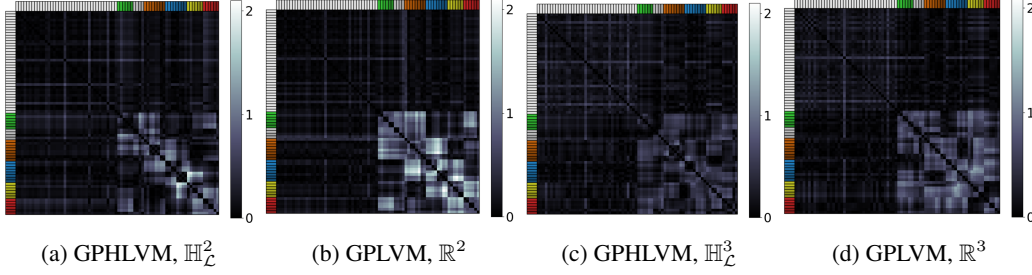


Figure 8: Pairwise error matrices between geodesic and taxonomy graph distances.

Section 5, alongside the base manifold and pullback geodesic interpolating from a ring to a spherical grasps and their corresponding decoded hand motions.

Second, we discuss the stress values provided in Table 2, which measure the average difference between the hyperbolic distances between embeddings and the corresponding taxonomy graph distances. Figure 8 complements the statistics presented in Table 2 and depicts the pairwise error matrices between the graph and manifold distances for the 2D and 3D GPLVMs and GPLVM. The left and top bar colors represent the embedding’s taxonomy classes following the color code of Figs. 5 and 7. The rest pose class (root node of the tree) is represented in white. We observe that, for all models, the pairwise errors between two leaf nodes (i.e., grasps), represented in the bottom-right corner of the plots, are higher than the pairwise errors between rest poses and grasps (i.e., root and leaf nodes), represented in the top-left, top-right, and bottom-left corners. This explains the large variance in the stress values of Table 2, and is due to the fact that a low error between the root and the leaves simply involves the leaf embeddings to be at the right distance from the origin, where the root is roughly located. Low errors between leaf nodes additionally involve the adequate relative placement of the corresponding embeddings. It is important to emphasize that the hyperbolic models lead to lower pairwise errors compared to their Euclidean counterparts for all taxonomy classes, as shown by the darker colors of Figs. 8a and 8c compared to Figs. 8b and 8d. Moreover, the 3D hyperbolic model leads to the lowest pairwise errors overall.

Third, we present additional visualizations of the decoded geodesics. While Figs. 5 and 7 only show 2 dimensions of the joint space, Fig. 9 depicts the decoded geodesics across all 24 dimensions, i.e., over the 24 degrees of freedom (DoF) of the hand model introduced in [36]. This model counts two

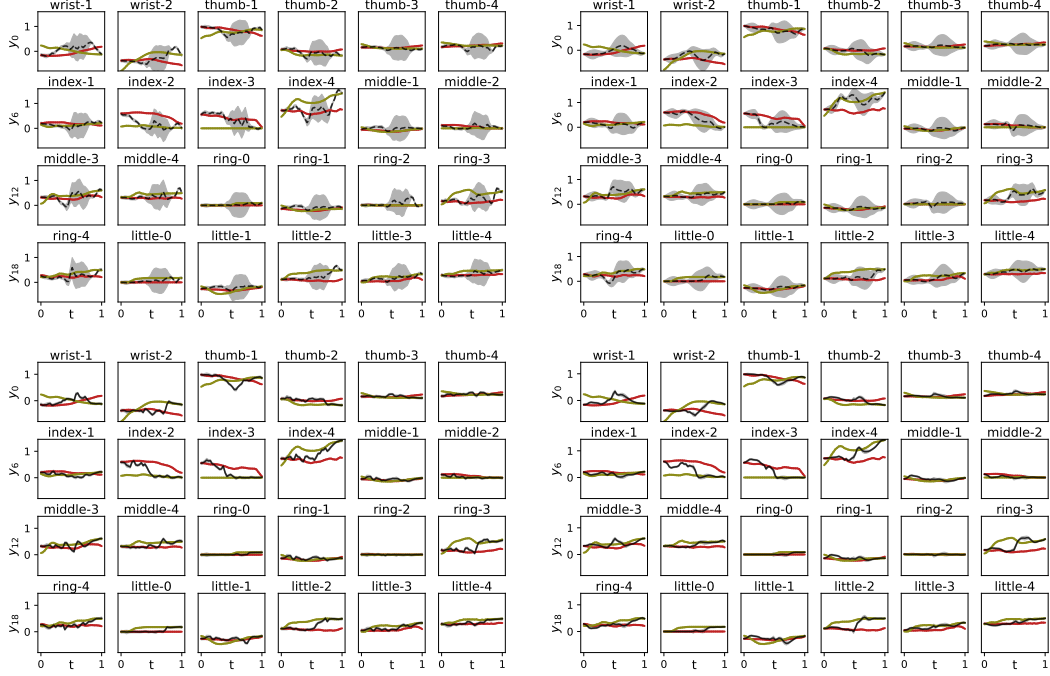


Figure 9: Time-series plots of all 24 dimensions of the joint space introduced in Fig. 5. The left and right columns show the decoded geodesics of the 2D and 3D latent spaces. The top and bottom rows show the mean of the decoded hyperbolic (---) and pullback geodesics (—) and their uncertainty as a gray envelope. For reference, the same training trajectory to the spherical grasp (—) and reversed training trajectory from the ring grasp (—) are included.

Table 3: Average computation times in seconds over 10 computations of the pullback metric at a random point \mathbf{x} and of a pullback geodesic between two random points \mathbf{x}, \mathbf{y} .

Model	Pullback metric	Geodesic
GPLVM \mathbb{R}^2	$0.65 \times 10^{-3} \pm 0.06 \times 10^{-3}$	4.14 ± 0.27
GPHLVM $\mathbb{H}_{\mathcal{L}}^2$	$44.2 \times 10^{-3} \pm 1.3 \times 10^{-3}$	38.08 ± 0.52
GPLVM \mathbb{R}^3	$0.63 \times 10^{-3} \pm 0.01 \times 10^{-3}$	4.19 ± 0.23
GPHLVM $\mathbb{H}_{\mathcal{L}}^3$	$1.77 \times 10^{-3} \pm 0.07 \times 10^{-3}$	13.67 ± 0.33

joints for the wrist, four joints for each finger, and two additional joints for the ring and little fingers to allow a better fist closure.

Finally, we discuss the influence of initialization on optimizing the pullback geodesics. As described in Sec. D.4, we initialize the pullback geodesic with two concatenated hyperbolic geodesics: One from the ring grasp to the origin and another from the origin to the spherical grasp. This initialization incorporates knowledge about the expected path in the taxonomy graph. In contrast, Fig. 10 shows the optimized pullback geodesics when the optimization is initialized solely with a hyperbolic geodesic from the ring grasp to the spherical grasp. In the two-dimensional case, we observe that the optimized pullback geodesic path differs significantly from the one shown in Fig. 5. This discrepancy is likely due to the geodesic optimization getting stuck in a local minimum, which may be a consequence of the Monte Carlo approximation of the 2D hyperbolic SE kernel. Interestingly, the three-dimensional pullback geodesic does not exhibit this issue, further supporting the hypothesis that the Monte Carlo approximation may be responsible since the 3D case does not rely on this approximation. In the 3D case, the optimized pullback geodesic follows a path very similar to the one shown in Fig. 5.

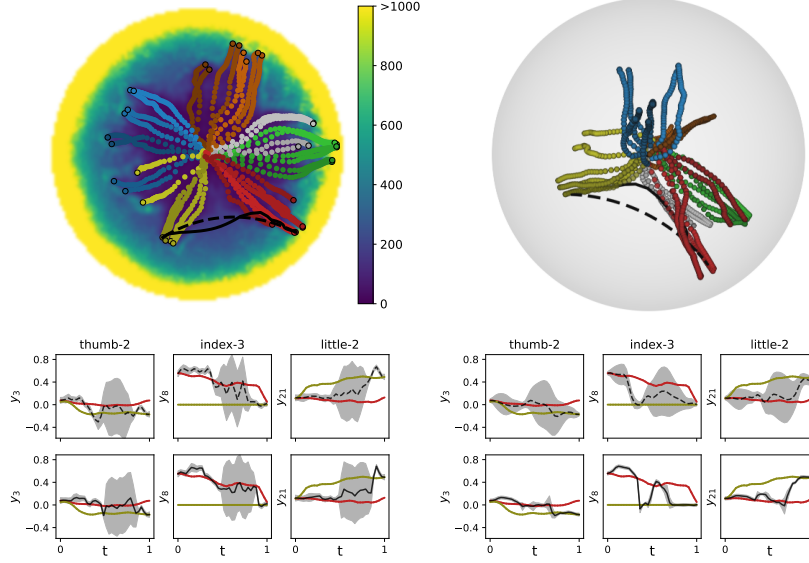


Figure 10: Hand grasps transition when initializing the geodesic optimization with the hyperbolic geodesic. As in Fig. 5, a transition from a ring (\bullet) to a spherical grasp (\bullet) is obtained using hyperbolic (---) and pullback (—) geodesics in \mathbb{H}_L^2 (left) and \mathbb{H}_L^3 (right). *Top*: 2D and 3D latent spaces of the trained models. *Bottom*: Time-series plots of 3 dimensions of the joint space for the decoded hyperbolic geodesic (---) and pullback geodesic (—).

D.6 Runtimes

Table 3 shows the runtime measurements for the computation of pullback metrics and pullback geodesics of GPLVM and GPLVM of the hand grasps generation experiment. The implementations are fully developed on Python, and the runtime measurements were taken using a Macbook Pro with Apple M3 Max chip with 14-core CPU and 36GB RAM. The main computational burden arises in the GPLVM with 2D latent space, which is in contrast with the experiments using a 3D latent space. We attribute this increase of computational cost mainly to the 2D hyperbolic kernel, as also observed in [25]. This may be alleviated by reducing the number of samples or via more efficient sampling strategies, as briefly discussed in Section 6.