On Probabilistic Pullback Metrics on Latent Hyperbolic Manifolds

Luis Augenstein[1] **Noémie Jaquier**[1,2] **Tamim Asfour**[1] **Leonel Rozo**[3]
[1]Karlsruhe Institute of Technology [2]KTH Royal Institute of Technology [3]Bosch Center for AI

Abstract

Gaussian Process Latent Variable Models (GPLVMs) have proven effective in capturing complex, high-dimensional data through lower-dimensional representations. Recent advances show that using Riemannian manifolds as latent spaces provides more flexibility to learn higher quality embeddings. This paper focuses on the hyperbolic manifold, a particularly suitable choice for modeling hierarchical relationships. While previous approaches relied on hyperbolic geodesics for interpolating the latent space, this often results in paths crossing low-data regions, leading to highly uncertain predictions. Instead, we propose augmenting the hyperbolic metric with a pullback metric to account for distortions introduced by the GPLVM's nonlinear mapping. Through various experiments, we 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 (Nickel and Kiela, 2017, 2018), as well as in social network analysis for modeling community structures (Krioukov et al., 2010; Doorenbos et al., 2024). Additionally, it finds applications in biology (Alanis-Lobato et al., 2018), human motion taxonomies (Jaquier et al., 2024), and computer vision (Khrulkov et al., 2020). However, many datasets in these disciplines are high dimensional and hetereogenous, 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 (Skopek et al., 2020; Cho et al., 2023; Jaquier et al., 2024). These embeddings leverage hyperbolic growth to accommodate hierarchical relationships, present in trees or graphs, that are difficult to model in Euclidean spaces (Cvetkovski and Crovella, 2009).

Despite the potential of hyperbolic embeddings, most of the 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 manifold. This problem can be understood from a differential-geometry point-of-view (Hauberg, 2019). For instance, recent works have focused on integrating the data manifold structure into latent spaces through stochastic pullback metrics (Tosi et al., 2014; Arvanitidis et al., 2018, 2021). This approach not only introduces uncertainty information into the latent space via the Riemannian metric, but also enables the generation of geodesics that more closely align with the true data distribution. The latter has been leveraged in robot motion generation (Beik-Mohammadi et al., 2021), protein sequencing (Detlefsen et al., 2022), and data augmentation for medical imaging (Chadebec et al., 2023). 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 latent variable models and 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. Second, we present a novel development of the Riemannian pullback metric tailored to Gaussian Process Hyperbolic Latent Variable Models (GPHLVMs) (Jaquier et al., 2024), incorporating appropriate Riemannian projections onto tangent spaces to account for the hyperbolic geometry. Third, we develop the kernel derivatives within this setting, high-

lighting 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, multicellular robot design (Dong et al., 2024), and human grasps generation (Jaquier et al., 2024).

2 Background

Hyperbolic Manifold: Before delving into the hyperbolic manifold, it is necessary to establish a basic understanding in Riemannian geometry (Lee, 2018). A Riemannian manifold $\mathcal{M}$ is a smooth manifold equipped with a smoothly varying inner product $g_{\boldsymbol{x}}: \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \times \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \to \mathbb{R} : (\boldsymbol{u}, \boldsymbol{v}) \mapsto \langle \boldsymbol{u}, \boldsymbol{v} \rangle_{\boldsymbol{x}} = \boldsymbol{u}^\intercal \boldsymbol{G}_{\boldsymbol{x}} \boldsymbol{v}$. The inner product $g_{\boldsymbol{x}}$ and equivalently the matrix $\boldsymbol{G}_{\boldsymbol{x}}$ are known as the Riemannian metric or metric tensor at each point $\boldsymbol{x} \in \mathcal{M}$. The tangent space $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ is the Euclidean space of tangent vectors of all the possible smooth curves passing through $\boldsymbol{x}$. To operate with Riemannian manifolds, it is common practice to exploit the Euclidean tangent spaces and shortest-path curves, so-called geodesics, between two points $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{M}$. The exponential map $\mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{u}) = \boldsymbol{y}$ maps a tangent space vector $\boldsymbol{u} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ to a point $\boldsymbol{y}$ on the manifold, so that $\boldsymbol{y}$ lies on a geodesic starting at $\boldsymbol{x}$ in the direction $\boldsymbol{u}$, and such that the geodesic distance $\mathrm{d}_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{y})$ between $\boldsymbol{x}$ and $\boldsymbol{y}$ equals the length of $\boldsymbol{u}$ given by the Riemannian norm $\|\boldsymbol{u}\|_{\boldsymbol{x}} = \sqrt{\langle \boldsymbol{u}, \boldsymbol{u} \rangle_{\boldsymbol{x}}}$. The inverse operation is the logarithmic map $\mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y}) = \boldsymbol{u}$. Finally, the parallel transport $\Gamma_{\boldsymbol{x} \to \boldsymbol{y}}(\boldsymbol{u}) = \boldsymbol{v}$ moves a tangent space vector $\boldsymbol{u}$ to the tangent space $\mathcal{T}_{\boldsymbol{y}}\mathcal{M}$ while preserving the Riemannian inner product.

The hyperbolic manifold is the only Riemannian manifold with constant negative curvature (Ratcliffe, 2019). It is commonly represented by either the Poincaré model $\mathbb{H}^D_{\mathcal{P}}$ (Poincaré, 1900), employing local coordinates within the unit sphere, or the Lorentz model $\mathbb{H}^D_{\mathcal{L}}$ (Jansen, 1909; Reynolds, 1993), which uses Cartesian coordinates to represent the surface embedded in $\mathbb{R}^{D+1}$. In this paper we mostly rely on the latter,

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

since it is numerically more stable. Further details on the hyperbolic manifold are provided in App. A.

Gaussian Process Hyperbolic Latent Variable Model (GPHLVM): A GPLVM generates observations $\boldsymbol{Y} = [\boldsymbol{y}_1 \dots \boldsymbol{y}_N]^\intercal \in \mathbb{R}^{N \times D_y}$ from latent variables $\boldsymbol{X} = [\boldsymbol{x}_1 \dots \boldsymbol{x}_N]^\intercal \in \mathbb{R}^{N \times D_x}$ with $D_x < D_y$ through a non-linear transformation modeled by Gaussian processes (GPs) (Lawrence, 2003). A GPLVM is formally described by,

$$p(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{\Theta}) = \prod_{d=1}^{D_y} \mathcal{N}(\boldsymbol{Y}_d \mid \boldsymbol{0}, \boldsymbol{K}_X + \sigma_y^2 \boldsymbol{I}_N), \quad (2)$$

where $\boldsymbol{Y}_d \in \mathbb{R}^N$ is the d-th observation column, $\boldsymbol{K}_X \in \mathbb{R}^{N \times N}$ is the kernel matrix $(\boldsymbol{K}_X)_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and $\boldsymbol{\Theta} = \{\tau, \kappa, \sigma_y^2\}$ is the set of scalar hyperparameters, namely, the kernel variance τ, the lengthscale κ, and the noise variance σ_y^2.

When the latent space is Euclidean, a common choice of kernel function is the Squared Exponential (SE) kernel $k^{\mathrm{SE}}(\boldsymbol{x}, \boldsymbol{x}') = \tau \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\kappa^2}\right)$. As discussed by Jaquier et al. (2024), an Euclidean latent space may not necessarily comply with structural biases in the observed data, and therefore curved geometries such as the hyperbolic manifold may be preferred. In such case the latent variables $\boldsymbol{x}_n \in \mathbb{H}^{D_x}_{\mathcal{L}}$ live in hyperbolic space, and consequently the kernel function is replaced by a hyperbolic kernel as introduced in the next section. Moreover, the latent variables are assigned a hyperbolic wrapped Gaussian distribution as proposed by (Jaquier et al., 2024), thus inducing the prior $p(\boldsymbol{X} \mid \alpha) = \prod_{n=1}^N \mathcal{N}_{\mathbb{H}^{D_x}_{\mathcal{L}}}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_0, \alpha \boldsymbol{I}_{D_x})$, with α controlling the spread of the latent variables (see App. A for more details). The GPHLVM latent points and hyperparameters are inferred via,

$$\underset{\boldsymbol{X}, \boldsymbol{\Theta}, \alpha}{\mathrm{argmax}}\ p(\boldsymbol{Y} \mid \boldsymbol{X}, \Theta)\, p(\boldsymbol{X} \mid \alpha), \quad (3)$$

which can be solved by finding a MAP estimate or via variational inference similar as in the Euclidean case.

Hyperbolic Kernels: The SE and Matérn kernels are standard choices when designing Gaussian Processes (Rasmussen and Williams, 2006). These kernels have been recently generalized to non-Euclidean spaces such as manifolds (Borovitskiy et al., 2020; Jaquier et al., 2021), or graphs (Borovitskiy et al., 2021).Grigoryan and Noguchi (1998) demonstrated that, in hyperbolic space, two- and three-dimensional heat kernels suffice since higher-dimensional kernels can be expressed as derivatives of these lower-dimensional ones. The two- and three-dimensional hyperbolic heat kernels are given as,

$$k^{\mathbb{H}^2_{\mathcal{L}}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{\tau}{C_\infty} \int_\rho^\infty \frac{s\, e^{-\frac{s^2}{2\kappa^2}}}{\sqrt{\cosh(s) - \cosh(\rho)}} \mathrm{d}s, \quad (4)$$

$$k^{\mathbb{H}^3_{\mathcal{L}}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{\tau}{C_\infty} \frac{\rho}{\sinh(\rho)} e^{-\frac{\rho^2}{2\kappa^2}}, \quad (5)$$

with points on the Lorentz model $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{H}^{D_x}_{\mathcal{L}}$, their geodesic distance $\rho = \mathrm{d}_{\mathbb{H}^{D_x}_{\mathcal{L}}}(\boldsymbol{x}, \boldsymbol{x}')$, kernel variance and lengthscale parameters $\tau, \kappa \in \mathbb{R}_+$, and a normalization

constant $C_\infty = (4\pi\kappa^2)^{\frac{3}{2}}$. In the 2D case no closed form solution is known. Instead, the kernel needs to be approximated via a discretization of the integral in (4). We use the Monte Carlo approximation introduced by Jaquier et al. (2024),

$$k^{\mathbb{H}^2_{\mathcal{L}}}(\boldsymbol{x},\boldsymbol{x}') \approx \frac{\tau}{C_\infty}\frac{1}{L}\sum_{l=1}^{L} s_l \tanh(\pi s_l)\Phi_l(\boldsymbol{x}_{\mathcal{P}},\boldsymbol{x}'_{\mathcal{P}})\,, \quad (6)$$

where $\Phi_l(\boldsymbol{x}_{\mathcal{P}},\boldsymbol{x}'_{\mathcal{P}}) = \phi_l(\boldsymbol{x}_{\mathcal{P}})\bar{\phi}_l(\boldsymbol{x}'_{\mathcal{P}})$, $\bar{\phi}_l$ denotes the complex conjugate of $\phi_l(\boldsymbol{x}_{\mathcal{P}}) = e^{(1+2s_l i)\langle \boldsymbol{x}_{\mathcal{P}},\boldsymbol{b}_l\rangle}$, $s_l \sim e^{-\frac{s^2\kappa^2}{2}}1_{[0,\infty]}(s)$ are samples from a truncated Gaussian distribution, $\boldsymbol{b}_l \sim U(\mathbb{S}^2)$ are samples from the unit circle, and $\langle \boldsymbol{x}_{\mathcal{P}},\boldsymbol{b}\rangle = \frac{1}{2}\log\left(\frac{1-|\boldsymbol{x}_{\mathcal{P}}|^2}{|\boldsymbol{x}_{\mathcal{P}}-\boldsymbol{b}|^2}\right)$ is the hyperbolic outer product with $\boldsymbol{x}_{\mathcal{P}} = \frac{1}{1+x_0}[x_1 \dots x_D]^\mathsf{T} \in \mathbb{H}^{D_x}_{\mathcal{P}}$ being the Poincaré representation of $\boldsymbol{x}$.

3 Metrics of Hyperbolic LVMs

Tosi et al. (2014) 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 latent variable models, specifically the GPHLVM introduced in Sec. 2. We show that the resulting metric can be computed similarly to the Euclidean case, with two key exceptions: *(1)* the derivatives of the GPHLVM nonlinear mapping must be projected onto appropriate tangent spaces; and *(2)* the computation of the derivatives for the hyperbolic heat kernels is significantly more challenging than in the Euclidean scenario.

3.1 Hyperbolic Pullback Metric

For a deterministic mapping $f : \mathbb{R}^{D_x} \to \mathbb{R}^{D_y}$, the pullback metric is defined as $\boldsymbol{G}_{\boldsymbol{x}^*} = \boldsymbol{J}^\mathsf{T}\boldsymbol{J}$, where $\boldsymbol{J} \in \mathbb{R}^{D_y \times D_x}$ is the Jacobian of f at $\boldsymbol{x}^*$. In the context of LVMs, the mapping f is stochastic. As Tosi et al. (2014), we consider LVMs for which the probability over $\boldsymbol{J}$ follows a Gaussian distribution. Specifically, assuming independent rows $\boldsymbol{J}_d \in \mathbb{R}^{D_x}$, each with its own mean but shared covariance matrix, the Jacobian distribution is of the form,

$$p(\boldsymbol{J}) = \prod_{d=1}^{D_y} \mathcal{N}(\boldsymbol{J}_d \mid \boldsymbol{\mu}_d, \boldsymbol{\Sigma}_J)\,. \quad (7)$$

Therefore, the metric tensor $\boldsymbol{G}_{\boldsymbol{x}^*}$ follows a non-central Wishart distribution (Anderson, 1946),

$$p(\boldsymbol{G}_{\boldsymbol{x}^*}) = \mathcal{W}_{D_x}(D_y, \boldsymbol{\Sigma}_J, \mathbb{E}[\boldsymbol{J}]^\mathsf{T}\mathbb{E}[\boldsymbol{J}])\,, \quad (8)$$

which we compute the expected metric tensor from,

$$\mathbb{E}[\boldsymbol{G}_{\boldsymbol{x}^*}] = \mathbb{E}[\boldsymbol{J}]^\mathsf{T}\mathbb{E}[\boldsymbol{J}] + D_y\boldsymbol{\Sigma}_J\,. \quad (9)$$

While the hyperbolic case follows a similar strategy, special care is required to ensure that the Jacobian rows lie on appropriate tangent spaces. Starting from the mapping $f : \mathbb{H}^{D_x}_{\mathcal{L}} \to \mathbb{R}^{D_y}$, the Riemannian Jacobian is composed by the Riemannian gradients for each output dimension, i.e., $\tilde{\boldsymbol{J}} = [\mathrm{grad}_{\boldsymbol{x}}(f_1) \dots \mathrm{grad}_{\boldsymbol{x}}(f_{D_y})]^\mathsf{T}$. As for Riemannian submanifolds (Boumal, 2023), the Lorentz gradient $\mathrm{grad}_{\boldsymbol{x}}(f) \in \mathcal{T}_{\boldsymbol{x}}\mathbb{H}^{D_x}_{\mathcal{L}}$ equals the orthogonal projectiong of the Euclidean gradient $\frac{\partial f}{\partial \boldsymbol{x}}$ onto the tangent space,

$$\mathrm{grad}_{\boldsymbol{x}}(f) = \mathrm{proj}_{\boldsymbol{x}}\left(\frac{\partial f}{\partial \boldsymbol{x}}\right) = \boldsymbol{P}_{\boldsymbol{x}}\frac{\partial f}{\partial \boldsymbol{x}}, \quad (10)$$

with $\mathrm{proj}_{\boldsymbol{x}} : \mathbb{R}^{D_x+1} \to \mathcal{T}_{\boldsymbol{x}}\mathbb{H}^{D_x}_{\mathcal{L}}$ (see App. A for details). Therefore, the Riemannian Jacobian $\tilde{\boldsymbol{J}}$ is computed from the Euclidean Jacobian $\boldsymbol{J}$ as $\tilde{\boldsymbol{J}} = \boldsymbol{J}\boldsymbol{P}^\mathsf{T}_{\boldsymbol{x}^*}$, and follows the Jacobian distribution,

$$p(\tilde{\boldsymbol{J}}) = \prod_{d=1}^{D_y} \mathcal{N}(\tilde{\boldsymbol{J}}_d \mid \boldsymbol{P}_{\boldsymbol{x}^*}\boldsymbol{\mu}_d, \boldsymbol{P}_{\boldsymbol{x}^*}\boldsymbol{\Sigma}_J\boldsymbol{P}^\mathsf{T}_{\boldsymbol{x}^*})\,. \quad (11)$$

Similarly to (9), the expected pullback metric tensor follows a non-central Wishart distribution,

$$\begin{aligned}\mathbb{E}[\boldsymbol{G}_{\boldsymbol{x}^*}] &= \mathbb{E}[\tilde{\boldsymbol{J}}]^\mathsf{T}\mathbb{E}[\tilde{\boldsymbol{J}}] + D_y\tilde{\boldsymbol{\Sigma}}_J \\ &= \boldsymbol{P}_{\boldsymbol{x}^*}(\mathbb{E}[\boldsymbol{J}]^\mathsf{T}\mathbb{E}[\boldsymbol{J}] + D_y\boldsymbol{\Sigma}_J)\boldsymbol{P}^\mathsf{T}_{\boldsymbol{x}^*}\,. \end{aligned} \quad (12)$$

3.2 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 $\boldsymbol{G}_{\boldsymbol{x}^*}$. As for GPLVMs (Tosi et al., 2014), the joint distribution of the Jacobian $\tilde{\boldsymbol{J}}$ and observations $\boldsymbol{Y}$ is given as,

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

where we rely on the hyperbolic kernel derivatives

$$\partial k(\boldsymbol{X},\boldsymbol{x}^*) = \frac{\partial}{\partial \boldsymbol{x}'}k(\boldsymbol{X},\boldsymbol{x}')\Big|_{\boldsymbol{x}'=\boldsymbol{x}^*}, \quad (14)$$

$$\partial^2 k(\boldsymbol{x}^*,\boldsymbol{x}^*) = \frac{\partial^2}{\partial \boldsymbol{x}'\,\partial \boldsymbol{x}}k(\boldsymbol{x},\boldsymbol{x}')\Big|_{\boldsymbol{x}=\boldsymbol{x}'=\boldsymbol{x}^*}. \quad (15)$$

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 $\boldsymbol{Y}_d \in \mathbb{R}^N$ results in the probability distribution of the Jacobian at $\boldsymbol{x}^*$ (Bishop, 2006),

$$p(\tilde{\boldsymbol{J}} \mid \boldsymbol{Y},\boldsymbol{X},\boldsymbol{x}_*) = \prod_{d=1}^{D_y}\mathcal{N}(\tilde{\boldsymbol{J}}_d \mid \boldsymbol{\mu}_d, \boldsymbol{\Sigma}_J), \quad (16)$$

$$\text{with} \quad \boldsymbol{\mu}_d = \boldsymbol{S}_J\boldsymbol{Y}_d\,, \quad (17)$$

$$\boldsymbol{\Sigma}_J = \partial^2 k(\boldsymbol{x}^*,\boldsymbol{x}^*) - \boldsymbol{S}_J\partial k(\boldsymbol{X},\boldsymbol{x}^*), \quad (18)$$

$$\boldsymbol{S}_J = \partial k(\boldsymbol{x}^*,\boldsymbol{X})(\boldsymbol{K}_X + \sigma_y^2\boldsymbol{I}_N)^{-1}\,. \quad (19)$$

Using Equation (12) we compute the final expected pullback metric tensor on the tangent space of $\boldsymbol{x}^*$ as,

$$\mathbb{E}[\boldsymbol{G}_{\boldsymbol{x}^*}] = \boldsymbol{P}_{\boldsymbol{x}^*}(\boldsymbol{\mu}_J^\mathsf{T}\boldsymbol{\mu}_J + D_y\boldsymbol{\Sigma}_J)\boldsymbol{P}_{\boldsymbol{x}^*}^\mathsf{T}\,, \tag{20}$$

with $\boldsymbol{\mu}_J = [\boldsymbol{\mu}_1 \ \dots \ \boldsymbol{\mu}_{D_y}]^\mathsf{T} \in \mathbb{R}^{D_y \times D_x}$.

3.3 Pullback Geodesics

Having endowed the hyperbolic latent space with a pullback metric, we are now interested in computing shortest paths, i.e., geodesics, between latent hyperbolic points. While hyperbolic geodesics have a closed-form solution, they may not always align with the underlying data distribution (see Fig. 1). Instead, pullback geodesics naturally follow the data distribution due to the pullback metric, but 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. When considering a discretized geodesic composed by a set of M points $\boldsymbol{x}_i \in \mathbb{H}^{D_x}_{\mathcal{L}}$, this boils down to minimize,

$$E = \sum_{i=0}^{M-2} \boldsymbol{v}_i^\mathsf{T}\boldsymbol{G}_{\boldsymbol{x}_i}\boldsymbol{v}_i\,, \quad \text{with} \quad \boldsymbol{v}_i = \mathrm{Log}_{\boldsymbol{x}_i}(\boldsymbol{x}_{i+1}). \tag{21}$$

While it is possible to iteratively optimize the curve points $\boldsymbol{x}_i$ directly, this often leads to uneven spacing among them. To address this issue, one could either use a parametric curve on the manifold (Gousenbourger et al., 2014) and optimize its parameters instead of the points directly; or alternatively, incorporate the spline energy as a regularization factor in the optimization process (Heeren et al., 2018). The latter is the approach we follow. Specifically, the spline energy is given as $E_{\text{spline}} \approx \sum_{i=1}^{M-2} \mathrm{d}_{\mathbb{H}^{D_x}_{\mathcal{L}}}(\boldsymbol{x}_i, \bar{\boldsymbol{x}}_i)^2$, where $\bar{\boldsymbol{x}}_i = \mathrm{Exp}_{\boldsymbol{x}_{i-1}}\left(\frac{1}{2}\,\mathrm{Log}_{\boldsymbol{x}_{i-1}}(\boldsymbol{x}_{i+1})\right)$ is the geodesic midpoint between $\boldsymbol{x}_{i-1}$ and $\boldsymbol{x}_{i+1}$. The final optimization problem is,

$$\min_{\boldsymbol{x}_0,\dots,\boldsymbol{x}_{M-1}} E + \lambda E_{\text{spline}}, \tag{22}$$

with λ weighting the influence of the regularization. Note that the optimization parameters are Riemannian, so we leverage Riemannian optimizers such as Riemannian Adam (Bécigneul and Ganea, 2019) to optimize (21).

4 Hyperbolic Kernel Derivatives

In this section, we discuss the challenges encountered when using autodiff tools, in particular PyTorch (Paszke et al., 2019), to compute the derivatives of the hyperbolic SE kernels. We observed that, as reported in Table 1, autodiff-based kernel derivatives are approximately five times slower than custom implementations in the two-dimensional case. More importantly, in the three-dimensional case, autodiff fails to compute the derivatives entirely.

	2D kernel ($L = 3000$)	3D kernel
PyTorch derivatives	0.83 ± 0.06	–
Custom derivatives	$\mathbf{0.16 \pm 0.02}$	$\mathbf{0.06 \pm 0.01}$

Table 1: Average computation times in seconds over 100 runs, each evaluating the pullback metric tensor and its derivatives at a random point. In the 3D case computing kernel derivatives with PyTorch is not feasible.

4.1 2D Hyperbolic Heat Kernel Derivatives

In order to compute the 2D hyperbolic heat kernel $k^{\mathbb{H}^2_{\mathcal{L}}}(\boldsymbol{x}, \boldsymbol{z})$[1] we rely on the Monte Carlo approximation given in Equation (4). To compute the pullback metric tensor $\boldsymbol{G}_{\boldsymbol{x}^*}$ at a point $\boldsymbol{x}^*$ we need the kernel derivatives $\frac{\partial}{\partial \boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{z})|_{\boldsymbol{x}=\boldsymbol{x}^*}$ and $\frac{\partial^2}{\partial \boldsymbol{z}\partial \boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{z})|_{\boldsymbol{x}=\boldsymbol{z}=\boldsymbol{x}^*}$, see Equations (14) and (15). The only part of the kernel (6) that depends on input points is the helper function $\Phi_l(\boldsymbol{x}, \boldsymbol{z})$, whose derivatives are given by,

$$\frac{\partial}{\partial \boldsymbol{x}}\Phi_l(\boldsymbol{x}, \boldsymbol{z}) = \left[\frac{\partial}{\partial \boldsymbol{x}}\phi_l(\boldsymbol{x}_\mathcal{P})\right]\bar{\phi}_l(\boldsymbol{z}_\mathcal{P})\,, \tag{23}$$

$$\frac{\partial^2}{\partial \boldsymbol{z}\partial \boldsymbol{x}}\Phi_l(\boldsymbol{x}, \boldsymbol{z}) = \left[\frac{\partial}{\partial \boldsymbol{x}}\phi_l(\boldsymbol{x}_\mathcal{P})\right]\left[\frac{\partial}{\partial \boldsymbol{z}}\bar{\phi}_l(\boldsymbol{z}_\mathcal{P})\right]^\mathsf{T}. \tag{24}$$

Using these derivatives we can compute the pullback metric tensor itself. Minimizing the curve energy (21), however, requires the derivative of the pullback metric tensor $\frac{\partial}{\partial \boldsymbol{x}^*}\boldsymbol{G}^{\mathcal{L}}_{\boldsymbol{x}^*}$. This, in turn, requires the derivative of the Jacobian of the covariance matrix $\frac{\partial}{\partial \boldsymbol{x}^*}\boldsymbol{\Sigma}_J$, which depends on the kernel derivatives $\frac{\partial^3}{\partial \boldsymbol{x}\partial \boldsymbol{z}\partial \boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{z})|_{\boldsymbol{x}=\boldsymbol{z}=\boldsymbol{x}^*}$, and $\frac{\partial^3}{\partial \boldsymbol{z}^2\partial \boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{z})|_{\boldsymbol{x}=\boldsymbol{z}=\boldsymbol{x}^*}$. Again the derivatives are effectively determined by the derivatives of $\Phi_l(\boldsymbol{x}, \boldsymbol{z})$,

$$\frac{\partial^3}{\partial \boldsymbol{x}\partial \boldsymbol{z}\partial \boldsymbol{x}}\Phi_l(\boldsymbol{x}, \boldsymbol{z}) = \left[\frac{\partial^2}{\partial \boldsymbol{x}^2}\phi_l(\boldsymbol{x}_\mathcal{P})\right] \times_1 \left[\frac{\partial}{\partial \boldsymbol{z}}\bar{\phi}_l(\boldsymbol{z}_\mathcal{P})\right]^\mathsf{T}, \tag{25}$$

$$\frac{\partial^3}{\partial \boldsymbol{z}^2\partial \boldsymbol{x}}\Phi_l(\boldsymbol{x}, \boldsymbol{z}) = \left[\frac{\partial^2}{\partial \boldsymbol{z}^2}\bar{\phi}_l(\boldsymbol{z}_\mathcal{P})\right] \times_0 \left[\frac{\partial}{\partial \boldsymbol{x}}\phi_l(\boldsymbol{x}_\mathcal{P})\right]^\mathsf{T}, \tag{26}$$

where $\boldsymbol{A} \times_i \boldsymbol{B}$ refers to the tensor product with the i-th dimension of tensor $\boldsymbol{A}$ being multiplied with the first dimension of matrix $\boldsymbol{B}$. Additionally, if matrix $\boldsymbol{B}$ is a row vector, $\times_i$ indicates that the tensor is unsqueezed at dimension i before performing the multiplication. In Equations (25) and (26) the matrices $\frac{\partial^2}{\partial \boldsymbol{x}^2}\phi_l(\boldsymbol{x}_\mathcal{P}), \frac{\partial^2}{\partial \boldsymbol{z}^2}\phi_l(\boldsymbol{z}_\mathcal{P}) \in \mathbb{R}^{3\times 3}$ are therefore effectively interpreted as $\mathbb{R}^{3\times 1\times 3}$ and $\mathbb{R}^{1\times 3\times 3}$, respectively. The complete derivation of all kernel derivatives is provided in App. B.

[1] We use $\boldsymbol{z}$ to denote the second input and avoid using a prime symbol.

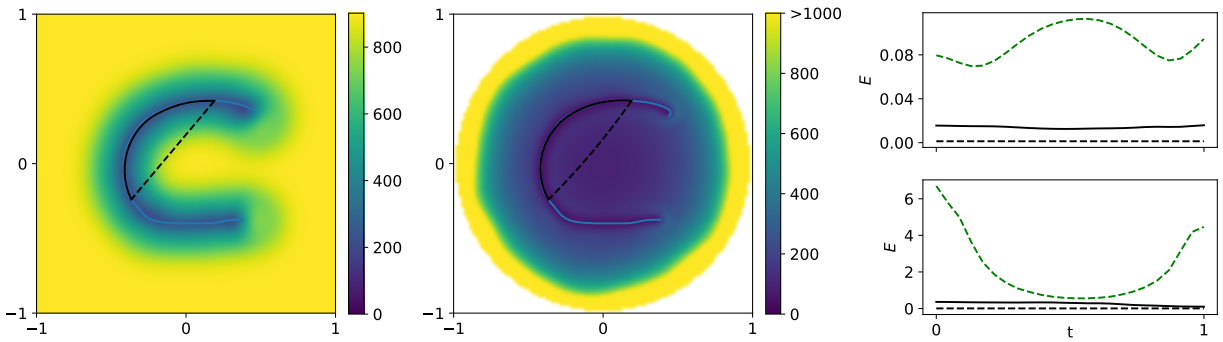


Figure 1: *Left* and *middle*: Euclidean and hyperbolic pullback metrics on a C-shape trajectory (—). The geodesic of the base manifold (- -) and the geodesic optimized on the pullback metric (—) are represented on both Euclidean and hyperbolic spaces. *Right*: Curve energy along the geodesics on the Euclidean (top) and hyperbolic (bottom) cases. These include the base manifold geodesic (- -), the base manifold geodesic with energy (21) evaluated using the pullback metric (- -), and the geodesic optimized with respect to the pullback metric (—).

4.2 3D Hyperbolic Heat Kernel Derivatives

In the three-dimensional case, the hyperbolic heat kernel is given as $k^{\mathbb{H}^3_{\mathcal{L}}}(\boldsymbol{x},\boldsymbol{z}) = \frac{\rho}{s}e^{-\frac{\rho^2}{\kappa}}$. Note that, compared to Equation (5), we simplified the notation by omitting the variance τ and normalization constant C_∞, by merging $2\kappa^2$ into just κ, and by introducing helper variables $u = \langle \boldsymbol{x},\boldsymbol{z}\rangle_{\mathcal{L}}$, and $s = \sqrt{u^2-1}$. As in the 2D case, we compute the first two kernel derivatives $\frac{\partial}{\partial \boldsymbol{x}}k(\boldsymbol{x},\boldsymbol{z})|_{\boldsymbol{x}=\boldsymbol{x}^*}$ and $\frac{\partial^2}{\partial \boldsymbol{z}\partial \boldsymbol{x}}k(\boldsymbol{x},\boldsymbol{z})|_{\boldsymbol{x}=\boldsymbol{z}=\boldsymbol{x}^*}$ in order to compute the pullback metric tensor $\boldsymbol{G}_{\boldsymbol{x}^*}$, as follows,

$$\frac{\partial}{\partial \boldsymbol{x}}k(\boldsymbol{x},\boldsymbol{z}) = g(u)\,\boldsymbol{G}\boldsymbol{z}\,e^{-\frac{\rho^2}{\kappa}}\,, \tag{27}$$

$$\frac{\partial^2}{\partial \boldsymbol{z}\partial \boldsymbol{x}}k(\boldsymbol{x},\boldsymbol{z}) = \left(h(u)\boldsymbol{G}\boldsymbol{z}\boldsymbol{x}^\mathsf{T}\boldsymbol{G} + g(u)\boldsymbol{G}\right)e^{-\frac{\rho^2}{\kappa}}\,, \tag{28}$$

where $\boldsymbol{G} = \mathrm{diag}(-1,1,...,1)$ is the Lorentzian metric tensor. The helper functions $g(u) = \left(\frac{2\rho^2}{\kappa s^2} - \frac{1}{s^2} - \frac{u\rho}{s^3}\right)$, and $h(u) = \frac{\mathrm{d}}{\mathrm{d}u}g(u) + \frac{g(u)2\rho}{\kappa s}$ are essential to understand why standard autodiff tools are unable to compute these derivatives. For equal kernel inputs $\boldsymbol{x} = \boldsymbol{z}$ the Lorentzian inner product u approaches -1 and both the distance ρ and helper variable s converge to 0. Analytically for the helper functions this limit is well behaved, e.g., $\lim_{u\to -1^-} g(u) = \frac{2}{\kappa} + \frac{1}{3}$. However, autodiff tools like PyTorch fail to compute these derivatives correctly because they are unable to cancel out the 0-approaching terms. Take for example $g(u)$,

$$\frac{2\rho^2}{\kappa s^2} - \frac{1}{s^2} - \frac{u\rho}{s^3} \to \overset{\text{autodiff}}{\underset{\boldsymbol{x}=\boldsymbol{z}}{\to}} \frac{0}{0} - \frac{1}{0} - \frac{0}{0} = \mathrm{NaN}\,.$$

In this scenario, autodiff ends up dividing by zero, resulting in undefined values (NaN). In other cases, e.g., when training a GPLVM, derivatives for equal kernel inputs $\boldsymbol{x} = \boldsymbol{z}$ may not be that relevant because different latent points rarely become that close. However, these derivatives are essential for computing the Jacobian covariance matrix (18). Although using symbolic differentiation libraries could provide a solution, they tend to be too slow for practical purposes. Therefore, the next best approach is to compute the necessary derivatives and their analytical limits manually. The analytical limits are then used whenever the inputs $\boldsymbol{x},\boldsymbol{z}$ are closer than a predefined threshold.

Finally, in order to optimize geodesics on the pullback metric we need the third-order kernel derivatives like in the 2D case. Let $\boldsymbol{K} = \frac{\partial^2}{\partial \boldsymbol{z}\partial \boldsymbol{x}}k(\boldsymbol{x},\boldsymbol{z}) \in \mathbb{R}^{4\times 4}$ now be the second kernel derivative. The third-order derivatives can then be obtained by stacking the individual derivatives of each row and column of $\boldsymbol{K}$. The derivative of each row $\boldsymbol{K}_i$ w.r.t. the first input $\boldsymbol{x}$ and each column $\boldsymbol{K}_j$ w.r.t the second input $\boldsymbol{z}$ is given by,

$$\frac{\partial \boldsymbol{K}_i}{\partial \boldsymbol{x}} = \left(q(u)\boldsymbol{G}\boldsymbol{z}\boldsymbol{x}^\mathsf{T}\boldsymbol{G}z_i + h(u)\boldsymbol{G}z_i + h(u)\boldsymbol{G}\boldsymbol{z}\boldsymbol{e}_i^\mathsf{T}\right)\Delta_{1i}e^{\frac{-\rho^2}{\kappa}},$$

$$\frac{\partial \boldsymbol{K}_j}{\partial \boldsymbol{z}} = \left(q(u)\boldsymbol{G}\boldsymbol{z}\boldsymbol{x}^\mathsf{T}\boldsymbol{G}x_j + h(u)\boldsymbol{G}x_j + h(u)\boldsymbol{e}_j\boldsymbol{x}^\mathsf{T}\boldsymbol{G}\right)\Delta_{1j}e^{\frac{-\rho^2}{\kappa}},$$

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

5 Experiments

In this section we present experiments to demonstrate that *(1)* The hyperbolic pullback metric augments the hyperbolic metric with the distortions introduced by the GPHLVM's nonlinear mapping; and *(2)* Similar to Euclidean pullback metrics, the hyperbolic pullback metric generates geodesics that adhere to the training data, allowing low uncertainty model predictions. Further experimental details are provided in App. C.

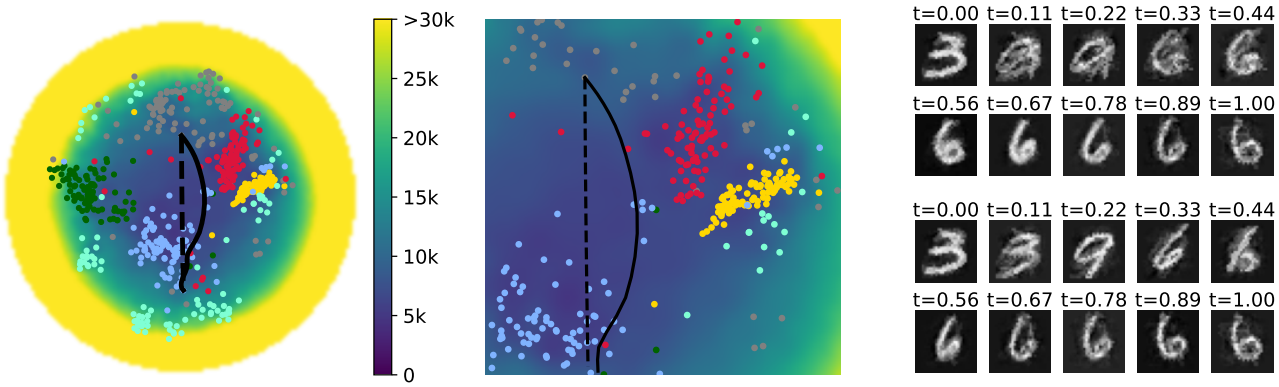


Figure 2: *Left*: 2D hyperbolic latent space of a subset of the MNIST dataset with 0 (•), 1 (•), 2 (•), 3 (•), 6 (•), and 9 (•). Both the hyperbolic (- -) and pullback (—) geodesics interpolate between a 3 and a 6. *Middle*: Zoomed-in view of the latent space to better visualize the metric volume. *Right*: Ten samples along the decoded geodesics in the image space, corresponding to the hyperbolic geodesic (top rows) and the the pullback geodesic (bottom rows).

5.1 C-shape

The first experiment serves as a proof of concept to visualize and compare the Euclidean and hyperbolic pullback metrics from a GPLVM and GPHLVM, respectively. We use a dataset of N two-dimensional C-shape points as both latent points and observations. In the hyperbolic case, the latent points and observations are represented in the Lorentz model, i.e, $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{H}^2_{\mathcal{L}}$. Only for visualization purposes we transform the latent points into the Poincaré model. To fully specify the models, we set the values for the variance, length scale, and noise variance, as $(\tau, \kappa, \sigma_y^2) = (0.7, 0.15, 0.69)$.

Figure 1 displays the Euclidean and hyperbolic pullback metric volumes. To generate these plots, we sampled a grid of points within the unit circle. Each grid point serves as a latent point $\boldsymbol{x}^*$, for which we compute the pullback metric tensor $\boldsymbol{G}_{\boldsymbol{x}^*}$ via Equation (20). The volume of $\boldsymbol{G}_{\boldsymbol{x}^*}$ is then given by $\sqrt{\det(\boldsymbol{G}_{\boldsymbol{x}^*})}$. In the hyperbolic case, the pullback metric tensor is a 3×3 matrix. Since this matrix lies in the two-dimensional tangent space of $\boldsymbol{x}^*$, one of its eigenvalues is always zero. To effectively visualize the volume, we exclude the zero eigenvalue. We observe in Figure 1 that, in the Euclidean case, the pullback metric volume is small near the C-shape data and increases away from it, until a maximum value is reached. In the hyperbolic case, the metric volume is also low nearby the data. However, it additionally follows the hyperbolic geometry: It is low near the origin and increases when moving outwards until it becomes infinite at the boundary of the unit circle. Overall, the hyperbolic pullback metric effectively integrates the properties of the hyperbolic manifold with those of the data manifold.

Figure 1 also depicts geodesics on both latent manifolds, which are generated with and without the pullback metric (solid and dashed black lines, respectively). In both cases, the pullback geodesic closely adheres to the training data. However, in the hyperbolic case, crossing the center experiences a much lower penalty, in terms of curve energy, than in the Euclidean case. The rightmost plots in Fig. 1 display the curve energy (21) along the geodesics. We observe that the geodesics on the base manifold and the pullback geodesic (dashed and solid black curves, respectively) both 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 (dashed green curve). Its energy is clearly not constant, indicating that the base manifold geodesic is not a true geodesic with respect to the pullback metric.

5.2 MNIST Digits

The natural clustering of MNIST images can be viewed as a hierarchy whose nodes are each of the 10 digit classes. From this point of view, hyperbolic spaces are well suited for embedding MNIST images (Mathieu et al., 2019). Similarly to previous works (Arvanitidis et al., 2018; Jørgensen and Hauberg, 2021; Lalchand et al., 2022), we here explore the interpolation of handwritten digits from a subset of the MNIST dataset. Each 28×28 image is vectorized and embedded via the GPHLVM. Figure 2 shows the 2D hyperbolic latent space with two geodesics interpolating between a sample of digit 3 and a sample of digit 6. The decoded hyperbolic geodesic (dashed black) and pullback geodesic (black), sampled at 10 time steps, show the result of geodesic interpolation in the image space.

As expected, the GPHLVM predictions collapse to the non-informative GP mean in regions with sparse

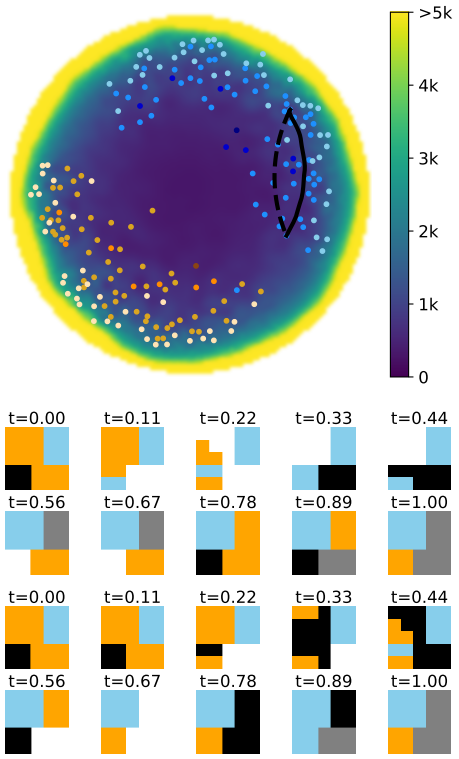


Figure 3: *Top*: 2D hyperbolic latent space of multicellular robot embeddings from coarse (darker tone) to fine (lighter tone). The embeddings form two clusters originating from an all-vertically-actuated robot (•) and an all-horizontally-actuated robot (•). *Bottom*: Ten samples along the decoded hyperbolic (—) and pullback (- -) geodesic.

data, resulting in blurry images. This is particularly evident along the hyperbolic geodesic, in the first half of the trajectory ($t = 0.11$ to $t = 0.44$). In contrast, the pullback geodesic tends to avoid empty regions by bending towards the red cluster (digit 9), leading to less blurry predictions overall. However, since MNIST mainly consists of distinct clusters without smooth transitions, the interpolation setting might be ill-posed. Despite this, our metric better captures the data manifold geometry, and it could thus be used to, for example, measure manifold-aware distances between latent variables.

5.3 Multi-cellular Robot Design

We build on the work of Dong et al. (2024), which introduced a coarse-to-fine framework for designing multicellular robots using hyperbolic embeddings. 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.

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 compute the pullback metric from the hierarchical multicellular robot embeddings and optimize geodesics to interpolate between existing robot designs, generating novel ones. Figure 3 shows the latent space and the mean predictions along both the hyperbolic and pullback geodesics. Note that, while the hyperbolic geodesic produces invalid robot designs, e.g., separated sections at $t = 0.22$, the pullback geodesic avoids this issue by complying with the geometry of the data support. This experiment suggests that pullback geodesics can assist the robot design process.

5.4 Hand Grasps Generation

Our final experiment centers around the grasps taxonomy of Stival et al. (2019), which categorizes human hand grasps into 20 distinct classes, organized within a hierarchical taxonomy graph. These grasps classes are further grouped into five categories: Flat grasps, distal grasps, cylindrical grasps, spherical grasps, and ring grasps. For each of the 20 grasp types, the hand movements of multiple subjects were recorded and made available through the KIT whole-body motion database (Mandery et al., 2016). For each grasp, subjects started the motion from the same resting pose before reaching out to grasp an object. The recorded motion resulted into trajectories $\boldsymbol{Y} \in \mathbb{R}^{N \times 24}$ in a 24-dimensional space, corresponding to the 24 degrees of freedom of the hand model. Training two GPHLVMs with dynamic prior akin to (Wang et al., 2008) on this dataset results in the 2D and 3D latent spaces shown in Fig. 4. All trajectories in the latent space start near the origin, representing the hand's initial resting pose, and progress outward. The final point of each trajectory reflects the final grasp.

The goal of this experiment is to use geodesics as a motion generation mechanism, thus enabling new motions that transition from one grasp to another. Figure 4 shows two geodesics interpolating from a ring grasp to a spherical grasp. While the hyperbolic geodesic crosses empty (high uncertainty) space, the pullback geodesic adheres more closely to the data support. We then decode the latent geodesics into the 24-dimensional hand joint space, to analyze the resulting hand motion. For a full analysis of all 24 predicted dimensions we refer to App. C.

As both geodesics start and end at the same grasps,

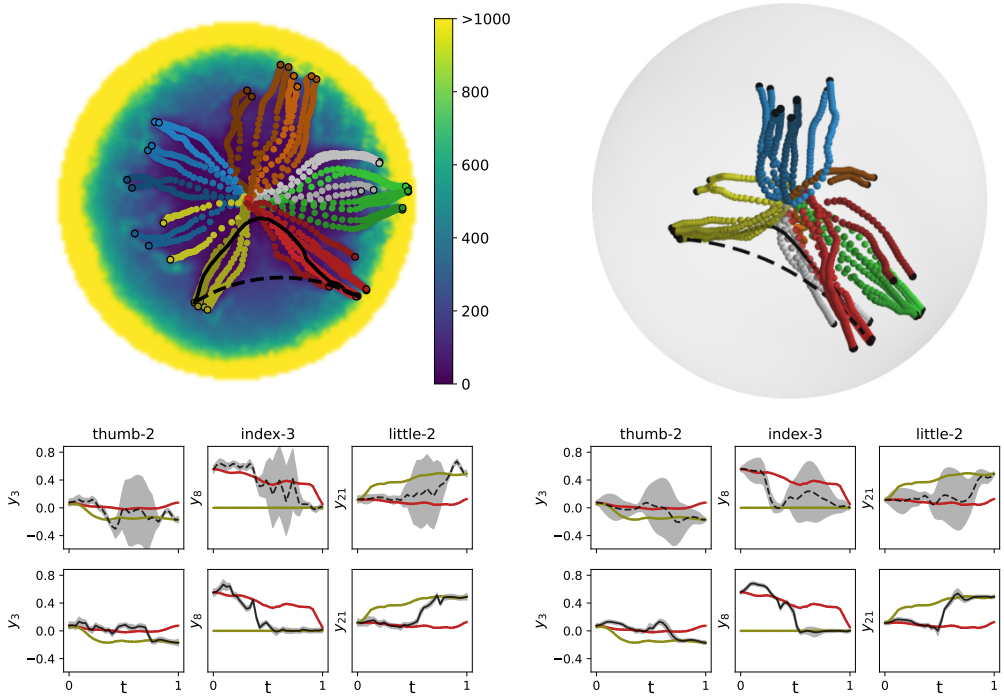


Figure 4: Hand grasps transition using geodesics optimized on the pullback metric (—) and the hyperbolic geodesic (- -), from a ring grasp (•) to a spherical grasp (•). *Top*: 2D and 3D latent spaces of the trained models, where the latent grasp trajectory points are coloured according to their corresponding grasp class. In the 2D plot, the background color represents the pullback metric volume. *Bottom*: Time-series plots of 3 dimensions of the joint space. The top and bottom rows show the mean of the decoded hyperbolic and pullback geodesics (- - and —) and their uncertainty as a gray envelope. For reference, a training trajectory to the spherical grasp (—) and a reversed training trajectory from the ring grasp (—) are included.

their predictions match at the beginning and the end. However, in the middle, the decoded hyperbolic geodesic shows high uncertainty. This is due to the GP reverting to its non-informative mean in regions with no data, where the GP prior dominates the posterior, leading to maximum uncertainty. In contrast, the decoded pullback geodesic shows much lower uncertainty, as the geodesic stays closer to the data support. Despite the low uncertainty, the predicted motions are not particularly smooth. They sometimes exhibit jerky motion trajectories produced by the fact that nearby latent points do not always correspond to equally similar hand configurations (i.e., similar joint values). However, the issue is partially mitigated in the 3D latent space, which provides more room to accommodate the latent embeddings. Improving the smoothness of trajectory predictions still remains an open problem for future work.

6 Conclusion

This paper advances the field of hyperbolic LVMs by augmenting the hyperbolic manifold with a Riemannian pullback metric that combines 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 heat kernel derivatives. Via multiple experiments, we demonstrated that pullback geodesics outperform standard hyperbolic 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. Future work could focus on generating latent embeddings with smoother transitions. Additionally, the need for computing manual kernel derivatives is unsatisfactory. Future work will explore alternative autodifferentiation techniques built on KeOps (Feydy et al., 2020) to overcome this practical issue. Finally, the approximation of the 2D hyperbolic heat 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.

Acknowledgements

This work was supported by the Carl Zeiss Foundation under the project JuBot and the European Union's Horizon Europe Framework Programme under grant agreement No 101070596 (euROBIN).

References

Alanis-Lobato, G., Mier, P., and Andrade-Navarro, M. (2018). The latent geometry of the human protein interaction network. *Bioinformatics*, 34(16):2826–2834.

Anderson, T. (1946). The non-central wishart distribution andcertain problems of multivariate statistics. *The Annals of Mathematical Statistics*, 17(4):409–431.

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

Arvanitidis, G., Hauberg, S., and Schölkopf, B. (2021). Geometrically enriched latent spaces. In *Intl. Conf. on Artificial Intelligence and Statistic (AISTATS)*.

Beik-Mohammadi, H., Hauberg, S., Arvanitidis, G., Neumann, G., and Rozo, L. (2021). Learning Riemannian manifolds for geodesic motion skills. In *Robotics: Science and Systems (R:SS)*.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

Borovitskiy, V., Azangulov, I., Terenin, A., Mostowsky, P., Deisenroth, M., and Durrande, N. (2021). Matérn Gaussian processes on graphs. In *Intl. Conf. on Artificial Intelligence and Statistic (AISTATS)*, pages 2593–2601.

Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. P. (2020). Matérn Gaussian processes on Riemannian manifolds. In *Neural Information Processing Systems (NeurIPS)*, pages 12426–12437.

Boumal, N. (2023). *An introduction to optimization on smooth manifolds*. Cambridge University Press.

Bécigneul, G. and Ganea, O.-E. (2019). Riemannian adaptive optimization methods. In *International Conference on Learning Representations (ICLR)*.

Chadebec, C., Thibeau-Sutre, E., Burgos, N., and Allassonnière, S. (2023). 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.

Cho, S., Lee, J., and Kim, D. (2023). Hyperbolic VAE via latent gaussian distributions. In *Neural Information Processing Systems (NeurIPS)*.

Cvetkovski, A. and Crovella, M. (2009). Hyperbolic embedding and routing for dynamic graphs. In *IEEE INFOCOM 2009*, pages 1647–1655.

Detlefsen, N. S., Hauberg, S., and Boomsma, W. (2022). Learning meaningful representations of protein sequences. *Nature Communications*, 13(1):1914.

Dong, H., Zhang, J., and Zhang, C. (2024). Leveraging hyperbolic embeddings for coarse-to-fine robot design. In *Intl. Conf. on Learning Representations (ICLR)*.

Doorenbos, L., Neila, P. M., Sznitman, R., and Mettes, P. (2024). Hyperbolic random forests. *Transactions on Machine Learning Research*.

Feix, T., Romero, J., Schmiedmayer, H.-B., Dollar, A. M., and Kragic, D. (2016). The GRASP taxonomy of human grasp types. *IEEE Transactions on Human-Machine Systems*, 46(1):66–77.

Feydy, J., Glaunès, J., Charlier, B., and Bronstein, M. (2020). Fast geometric learning with symbolic matrices. In *Neural Information Processing Systems (NeurIPS)*, volume 33.

Gousenbourger, P.-Y., Samir, C., and Absil, P. (2014). Piecewise-bézier c1 interpolation on riemannian manifolds with application to 2d shape morphing. In *Intl. Conf. on Pattern Recognition*, pages 4086–4091.

Grigoryan, A. and Noguchi, M. (1998). The heat kernel on hyperbolic space. *Bulletin of the London Mathematical Society*, 30(6):643–650.

Hauberg, S. (2019). Only Bayes should learn a manifold. *arXiv preprint arXiv:1806.04994*.

Heeren, B., Rumpf, M., and Wirth, B. (2018). Variational time discretization of Riemannian splines. *IMA Journal of Numerical Analysis*, 39(1):61–104.

Jansen, H. (1909). Abbildung der hyperbolischen geometrie auf ein zweischaliges hyperboloid. *Mitteilungen der Mathematischen Gesellschaft in Hamburg*, 4:409–440.

Jaquier, N., Borovitskiy, V., Smolensky, A., Terenin, A., Asfour, T., and Rozo, L. (2021). Geometry-aware Bayesian optimization in robotics using Riemannian Matérn kernels. In *Conference on Robot Learning (CoRL)*.

Jaquier, N., Rozo, L., González-Duque, M., Borovitskiy, V., and Asfour, T. (2024). Bringing motion taxonomies to continuous domains via GPLVM on hyperbolic manifolds. In *International Conference on Machine Learning (ICML)*.

Jørgensen, M. and Hauberg, S. (2021). Isometric Gaussian process latent variable model for dissimilarity data. In *Intl. Conf. on Machine Learning (ICML)*, pages 5127–5136.

Khrulkov, V., Mirvakhabova, L., Ustinova, E., Oseledets, I., and Lempitsky, V. (2020). Hyperbolic image embeddings. In *Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM Review*, 51(3):455–500.

Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A., and Boguñá, M. (2010). Hyperbolic geometry of complex networks. *Phys. Rev. E*, 82:036106.

Lalchand, V., Ravuri, A., and Lawrence, N. D. (2022). Generalised GPLVM with stochastic variational inference. In *Intl. Conf. on Artificial Intelligence and Statistic (AISTATS)*, pages 7841–7864.

Lawrence, N. D. (2003). Gaussian process latent variable models for visualisation of high dimensional data. In *Neural Information Processing Systems (NeurIPS)*.

Lee, J. (2018). *Introduction to Riemannian Manifolds*. Springer, 2nd edition.

Mandery, C., Terlemez, O., Do, M., Vahrenkamp, N., and Asfour, T. (2016). Unifying representations and large-scale whole-body motion databases for studying human motion. *IEEE Trans. on Robotics*, 32(4):796–809.

Mathieu, E., Le Lan, C., Maddison, C. J., Tomioka, R., and Teh, Y. W. (2019). Continuous hierarchical representations with Poincaré variational autoencoders. In *Neural Information Processing Systems (NeurIPS)*.

Nagano, Y., Yamaguchi, S., Fujita, Y., and Koyama, M. (2019). A wrapped normal distribution on hyperbolic space for gradient-based learning. In *Intl. Conf. on Machine Learning (ICML)*, pages 4693–4702.

Nickel, M. and Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. In *Neural Information Processing Systems (NeurIPS)*.

Nickel, M. and Kiela, D. (2018). Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In *Intl. Conf. on Machine Learning (ICML)*, pages 3779–3788.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: an imperative style, high-performance deep learning library. In *Neural Information Processing Systems (NeurIPS)*.

Poincaré, H. (1900). Théorie des groupes fuchsiens. *Acta Mathematica*, 1:1 – 62.

Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Ratcliffe, J. G. (2019). *Foundations of Hyperbolic Manifolds*. Springer, 3rd edition.

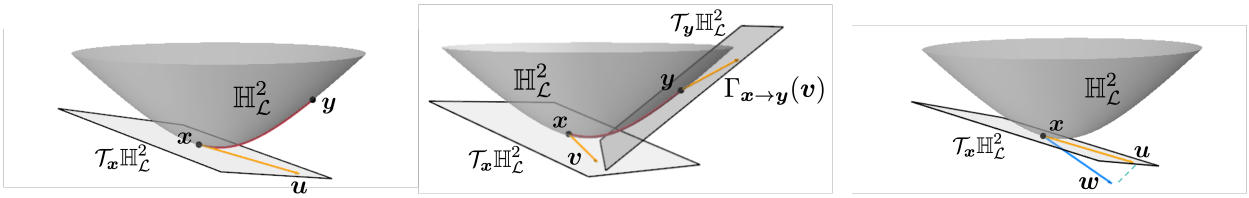
Reynolds, W. F. (1993). Hyperbolic geometry on a hyperboloid. *The American Mathematical Monthly*, 100(5):442–455.

Skopek, O., Ganea, O.-E., and Bécigneul, G. (2020). Mixed-curvature variational autoencoders. In *Intl. Conf. on Learning Representations (ICLR)*.

Stival, F., Michieletto, S., Cognolato, M., Pagello, E., Müller, H., and Atzori, M. (2019). A quantitative taxonomy of human hand grasps. *Journal of NeuroEngineering and Rehabilitation*, 16(28).

Tosi, A., Hauberg, S., Vellido, A., and Lawrence, N. D. (2014). Metrics for probabilistic geometries. In *Conference on Uncertainty in Artificial Intelligence (UAI)*.

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



(a) Exponential and logarithmic maps. (b) Parallel transport. (c) Tangent space projection.

Figure 5: Principal Riemannian operations on the Lorentz model $\mathbb{H}^2_{\mathcal{L}}$. **(a)** The geodesic (—) is the shortest path between the two points $\boldsymbol{x}$ to $\boldsymbol{y}$ on the manifold. Its length is equal to the geodesic distance $\mathrm{d}_{\mathbb{H}^D_{\mathcal{L}}}(\boldsymbol{x}, \boldsymbol{y})$. The vector $\boldsymbol{u}$ (⟶) lies on the tangent space of $\boldsymbol{x}$ such that $\boldsymbol{y} = \mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{u})$. **(b)** Parallel transport $\Gamma_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{u})$ of the vector $\boldsymbol{v}$ from $\mathcal{T}_{\boldsymbol{x}}\mathbb{H}^2_{\mathcal{L}}$ to $\mathcal{T}_{\boldsymbol{y}}\mathbb{H}^2_{\mathcal{L}}$. **(c)** The vector $\boldsymbol{w}$ (⟶) is projected onto the tangent space of $\boldsymbol{x}$ via the tangent space projection $\boldsymbol{u} = \mathrm{proj}_{\boldsymbol{x}}(\boldsymbol{w})$.

A Hyperbolic Manifold

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

$$\mathbb{H}^D_{\mathcal{L}} = \{\boldsymbol{x} \in \mathbb{R}^{D+1} \mid \langle \boldsymbol{x}, \boldsymbol{x}\rangle_{\mathcal{L}} = -1, x_0 > 0\}\,, \tag{29}$$

$$\mathcal{T}_{\boldsymbol{x}}\mathbb{H}^D_{\mathcal{L}} = \{\boldsymbol{u} \in \mathbb{R}^{D+1} \mid \langle \boldsymbol{u}, \boldsymbol{x}\rangle_{\mathcal{L}} = 0\}\,, \tag{30}$$

where $\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathcal{L}} = \boldsymbol{x}^\mathsf{T}\boldsymbol{G}\boldsymbol{y}$ is the lorentzian inner product with the Lorentzian metric tensor $\boldsymbol{G} = \mathrm{diag}(-1, 1, ..., 1)$. Note that, since the curvature of the hyperbolic manifold is constant, the metric tensor does not depend on $\boldsymbol{x}$ as for general manifolds, i.e., $\boldsymbol{G}_{\boldsymbol{x}} = \boldsymbol{G}$ for all $\boldsymbol{x} \in \mathbb{H}^D_{\mathcal{L}}$. 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,

$$\mathrm{d}_{\mathbb{H}^D_{\mathcal{L}}}(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{arccosh}(-\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathcal{L}})\,, \tag{31}$$

$$\mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{u}) = \cosh(\|\boldsymbol{u}\|_{\mathcal{L}})\boldsymbol{x} + \sinh(\|\boldsymbol{u}\|_{\mathcal{L}})\frac{\boldsymbol{u}}{\|\boldsymbol{u}\|_{\mathcal{L}}}\,, \tag{32}$$

$$\mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{y}) = \mathrm{d}_{\mathbb{H}^D_{\mathcal{L}}}(\boldsymbol{x}, \boldsymbol{y})\frac{\boldsymbol{y} + \langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathcal{L}}\boldsymbol{x}}{\sqrt{\langle \boldsymbol{x}, \boldsymbol{y}\rangle^2_{\mathcal{L}} - 1}}\,, \tag{33}$$

$$\Gamma_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{u}) = \boldsymbol{u} + \frac{\langle \boldsymbol{y}, \boldsymbol{u}\rangle_{\mathcal{L}}}{1 - \langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathcal{L}}}(\boldsymbol{x} + \boldsymbol{y}) \tag{34}$$

$$\mathrm{proj}_{\boldsymbol{x}}(\boldsymbol{w}) = \boldsymbol{P}_{\boldsymbol{x}}\boldsymbol{g} = (\boldsymbol{G} + \boldsymbol{x}\boldsymbol{x}^\mathsf{T})\boldsymbol{w}\,. \tag{35}$$

These operations are illustrated by Figure 5.

Working with probabilistic models on Riemannian manifolds requires probability distributions that account for their geometry. Therefore, for the GPHLVM prior in equation (3), we rely on the hyperbolic wrapped distribution (Nagano et al., 2019), which builds on a Gaussian distribution on the tangent space at the origin $\boldsymbol{\mu}_0 = [1\ 0\ \ldots\ 0]^\mathsf{T} \in \mathbb{H}^D_{\mathcal{L}}$. Intuitively, the wrapped distribution is constructed as follows: *(1)* Sample a point $\tilde{\boldsymbol{v}} \in \mathbb{R}^D$ from the Euclidean Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$; *(2)* Transform $\tilde{\boldsymbol{v}}$ to an element of the tangent space $\mathcal{T}_{\boldsymbol{\mu}_0}\mathbb{H}^D_{\mathcal{L}}$ at the origin by setting $\boldsymbol{v} = [0\ \tilde{\boldsymbol{v}}]^\mathsf{T}$; *(3)* Parallel transport $\boldsymbol{v}$ to the desired mean $\boldsymbol{u} = \Gamma_{\boldsymbol{\mu}_0\to\boldsymbol{\mu}}(\boldsymbol{v})$; and *(4)* Project $\boldsymbol{u}$ onto the hyperbolic space via the exponential map $\boldsymbol{x} = \mathrm{Exp}_{\boldsymbol{\mu}}(\boldsymbol{u})$. The resulting probability density function is

$$\mathcal{N}_{\mathbb{H}^D_{\mathcal{L}}}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}(\tilde{\boldsymbol{v}} \mid \boldsymbol{0}, \boldsymbol{\Sigma})\left(\frac{r}{\sinh(r)}\right)^{D-1}\,, \tag{36}$$

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

B Hyperbolic Kernel Derivatives

This section provides the complete derivations of the hyperbolic heat kernel derivatives that are necessary to compute the pullback metric tensor and to optimize geodesics on it. Equation (20) defines the pullback metric

tensor $\boldsymbol{G}_{\boldsymbol{x}^*}$ based on the Jacobian mean $\boldsymbol{\mu}_J$ (17) and covariance matrix $\boldsymbol{\Sigma}_J$ (18), which in turn require the kernel derivatives $\partial k(\boldsymbol{x}^*, \boldsymbol{X}) = \frac{\partial}{\partial \boldsymbol{x}^*} k(\boldsymbol{x}^*, \boldsymbol{X}) \in \mathbb{R}^{D_x \times N}$ and $\partial^2 k(\boldsymbol{x}^*, \boldsymbol{x}^*) = \frac{\partial^2}{\partial \boldsymbol{z} \partial \boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{z})|_{\boldsymbol{x}=\boldsymbol{z}=\boldsymbol{x}^*} \in \mathbb{R}^{D_x \times D_x}$. Additionally, minimizing the curve energy (21), requires the derivative of the pullback metric tensor, i.e., the derivative of $\boldsymbol{\mu}_J^\mathsf{T} \boldsymbol{\mu}_J + D_y \boldsymbol{\Sigma}_J$, which requires the kernel derivative $\dfrac{\partial^3}{\partial \boldsymbol{x} \partial \boldsymbol{z} \partial \boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{z})$ and $\dfrac{\partial^3}{\partial \boldsymbol{z}^2 \partial \boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{z})$. Specifically, we have

$$\frac{\partial}{\partial \boldsymbol{x}^*}(\boldsymbol{\mu}_J^\mathsf{T} \boldsymbol{\mu}_J + D_y \boldsymbol{\Sigma}_J) = \left[\frac{\partial}{\partial \boldsymbol{x}^*} \boldsymbol{\mu}_J^\mathsf{T}\right] \times_1 \boldsymbol{\mu}_J + \left[\frac{\partial}{\partial \boldsymbol{x}^*} \boldsymbol{\mu}_J\right] \times_0 \boldsymbol{\mu}_J + D_y \left[\frac{\partial}{\partial \boldsymbol{x}^*} \boldsymbol{\Sigma}_J\right], \tag{37}$$

$$\text{with} \quad \frac{\partial}{\partial \boldsymbol{x}^*} \boldsymbol{\mu}_J^\mathsf{T} = \left[\frac{\partial}{\partial \boldsymbol{x}^*} \partial k(\boldsymbol{x}^*, \boldsymbol{X})\right] \times_1 (\boldsymbol{K}_X + \sigma_y^2 \boldsymbol{I}_N)^{-1} \boldsymbol{Y}, \tag{38}$$

$$\frac{\partial}{\partial \boldsymbol{x}^*} \boldsymbol{\Sigma}_J = \left[\frac{\partial}{\partial \boldsymbol{x}^*} \partial^2 k(\boldsymbol{x}^*, \boldsymbol{x}^*)\right] - \left[\frac{\partial}{\partial \boldsymbol{x}^*} \partial k(\boldsymbol{x}^*, \boldsymbol{X})\right] \times_1 \boldsymbol{S}_J^\mathsf{T} - \left[\frac{\partial}{\partial \boldsymbol{x}^*} \partial k(\boldsymbol{X}, \boldsymbol{x}^*)\right] \times_0 \boldsymbol{S}_J^\mathsf{T}, \tag{39}$$

$$\frac{\partial}{\partial \boldsymbol{x}^*} \partial^2 k(\boldsymbol{x}^*, \boldsymbol{x}^*) = \left[\frac{\partial^3}{\partial \boldsymbol{x} \partial \boldsymbol{z} \partial \boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{z}) + \frac{\partial^3}{\partial \boldsymbol{z}^2 \partial \boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{z})\right]\Bigg|_{\boldsymbol{x}=\boldsymbol{z}=\boldsymbol{x}^*}, \tag{40}$$

where we have $\boldsymbol{\mu}_J \in \mathbb{R}^{D_y \times D_x}$, $\dfrac{\partial}{\partial \boldsymbol{x}^*} \boldsymbol{\mu} \in \mathbb{R}^{D_y \times D_x \times D_x}$, $\dfrac{\partial}{\partial \boldsymbol{x}^*} \partial k(\boldsymbol{x}^*, \boldsymbol{X}) \in \mathbb{R}^{D_x \times N \times D_x}$, and $\times_n$ denotes the n-th mode tensor product (Kolda and Bader, 2009). Next, we provide the details of the derivation of the kernel derivatives for $\mathbb{H}^2_{\mathcal{L}}$ and $\mathbb{H}^3_{\mathcal{L}}$.

B.1 2D Hyperbolic Kernel Derivatives

As discussed in Section 4.1, the derivatives of the 2D hyperbolic heat kernel (6) are entirely determined by the derivatives of $\Phi_l(\boldsymbol{x}, \boldsymbol{z})$. Equations (23), (24), (25), and (26) show that these derivatives in turn are defined by the first and second derivatives of $\phi_l(\boldsymbol{x}_\mathcal{P}) = e^{(1+2s_l i)\langle \boldsymbol{x}_\mathcal{P}, \boldsymbol{b}_l \rangle}$, which are computed as

$$\frac{\partial}{\partial \boldsymbol{x}} \phi_l(\boldsymbol{x}_\mathcal{P}) = (1 + 2s_l i)\, \phi_l(\boldsymbol{x}_\mathcal{P}) \left[\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}_\mathcal{P}\right]^\mathsf{T} \left[\frac{\partial}{\partial \boldsymbol{x}_\mathcal{P}} \langle \boldsymbol{x}_\mathcal{P}, \boldsymbol{b} \rangle\right], \tag{41}$$

$$\begin{aligned}
\frac{\partial^2}{\partial \boldsymbol{x}^2} \phi_l(\boldsymbol{x}_\mathcal{P}) = (1 + 2s_l i) \Bigg(& \left[\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}_\mathcal{P}\right]^\mathsf{T} \left[\frac{\partial}{\partial \boldsymbol{x}_\mathcal{P}} \langle \boldsymbol{x}_\mathcal{P}, \boldsymbol{b} \rangle\right] \left[\frac{\partial}{\partial \boldsymbol{x}} \phi_l(\boldsymbol{x}_\mathcal{P})\right]^\mathsf{T} \\
& + \phi_l(\boldsymbol{x}_\mathcal{P}) \left[\frac{\partial^2}{\partial \boldsymbol{x}^2} \boldsymbol{x}_\mathcal{P}\right] \times_0 \left[\frac{\partial}{\partial \boldsymbol{x}_\mathcal{P}} \langle \boldsymbol{x}_\mathcal{P}, \boldsymbol{b} \rangle\right] \\
& + \phi_l(\boldsymbol{x}_\mathcal{P}) \left[\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}_\mathcal{P}\right]^\mathsf{T} \left[\frac{\partial}{\partial \boldsymbol{x}} \frac{\partial}{\partial \boldsymbol{x}_\mathcal{P}} \langle \boldsymbol{x}_\mathcal{P}, \boldsymbol{b} \rangle\right] \Bigg),
\end{aligned} \tag{42}$$

$$\begin{aligned}
= (1 + 2s_l i)\, \phi_l(\boldsymbol{x}_\mathcal{P}) \Bigg(& (1 + 2s_l i) \left[\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}_\mathcal{P}\right]^\mathsf{T} \left[\frac{\partial}{\partial \boldsymbol{x}_\mathcal{P}} \langle \boldsymbol{x}_\mathcal{P}, \boldsymbol{b} \rangle\right] \left[\frac{\partial}{\partial \boldsymbol{x}_\mathcal{P}} \langle \boldsymbol{x}_\mathcal{P}, \boldsymbol{b} \rangle\right]^\mathsf{T} \left[\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}_\mathcal{P}\right] \\
& + \left[\frac{\partial^2}{\partial \boldsymbol{x}^2} \boldsymbol{x}_\mathcal{P}\right] \times_0 \left[\frac{\partial}{\partial \boldsymbol{x}_\mathcal{P}} \langle \boldsymbol{x}_\mathcal{P}, \boldsymbol{b} \rangle\right] \\
& + \left[\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}_\mathcal{P}\right]^\mathsf{T} \left[\frac{\partial^2}{\partial \boldsymbol{x}_\mathcal{P}^2} \langle \boldsymbol{x}_\mathcal{P}, \boldsymbol{b} \rangle\right] \left[\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}_\mathcal{P}\right] \Bigg).
\end{aligned} \tag{43}$$

As shown by these expressions, the derivatives of $\phi_l(\boldsymbol{x}_\mathcal{P})$ 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 $\langle \boldsymbol{x}_\mathcal{P}, \boldsymbol{b} \rangle = \frac{1}{2} \log\left(\frac{1 - |\boldsymbol{x}_\mathcal{P}|^2}{|\boldsymbol{x}_\mathcal{P} - \boldsymbol{b}|^2}\right)$ are given as,

$$\frac{\partial}{\partial \boldsymbol{x}_\mathcal{P}} \langle \boldsymbol{x}_\mathcal{P}, \boldsymbol{b} \rangle = -\frac{1}{2} \left(\frac{2\boldsymbol{x}_\mathcal{P}}{1 - |\boldsymbol{x}_\mathcal{P}|^2} + \frac{2(\boldsymbol{x}_\mathcal{P} - \boldsymbol{b})}{|\boldsymbol{x}_\mathcal{P} - \boldsymbol{b}|^2}\right) = \frac{\boldsymbol{x}_\mathcal{P}}{|\boldsymbol{x}_\mathcal{P}|^2 - 1} - \frac{\boldsymbol{x}_\mathcal{P} - \boldsymbol{b}}{|\boldsymbol{x}_\mathcal{P} - \boldsymbol{b}|^2}, \tag{44}$$

$$\frac{\partial^2}{\partial \boldsymbol{x}_\mathcal{P}^2} \langle \boldsymbol{x}_\mathcal{P}, \boldsymbol{b} \rangle = \frac{1}{|\boldsymbol{x}_\mathcal{P}|^2 - 1} \boldsymbol{I}_{D_x} - \frac{2\boldsymbol{x}_\mathcal{P} \boldsymbol{x}_\mathcal{P}^\mathsf{T}}{(|\boldsymbol{x}_\mathcal{P}|^2 - 1)^2} - \frac{1}{|\boldsymbol{x}_\mathcal{P} - \boldsymbol{b}|^2} \boldsymbol{I}_{D_x} + \frac{2(\boldsymbol{x}_\mathcal{P} - \boldsymbol{b})(\boldsymbol{x}_\mathcal{P} - \boldsymbol{b})^\mathsf{T}}{|\boldsymbol{x}_\mathcal{P} - \boldsymbol{b}|^4}. \tag{45}$$

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

$$\left(\frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{x}_{\mathcal{P}}\right)_{i,j} = \begin{cases} -\frac{x_{i+1}}{(1+x_0)^2} & j=0 \\ \frac{1}{1+x_0} & j=i+1 \\ 0 & \text{otherwise} \end{cases}, \qquad \left(\frac{\partial^2}{\partial \boldsymbol{x}^2}\boldsymbol{x}_{\mathcal{P}}\right)_{i,j,k} = \begin{cases} \frac{2x_{i+1}}{(1+x_0)^3} & j=k=0 \\ -\frac{1}{(1+x_0)^2} & j=0 \text{ and } k=i+1 \\ -\frac{1}{(1+x_0)^2} & k=0 \text{ and } j=i+1 \\ 0 & \text{otherwise} \end{cases}, \tag{46}$$

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

B.2 3D Hyperbolic Kernel Derivatives

This section provides the derivations of the 3D hyperbolic heat kernel derivatives. Section 4.2 introduced the helper variables $u = \langle \boldsymbol{x}, \boldsymbol{z} \rangle_{\mathcal{L}}$ and $s = \sqrt{u^2 - 1}$ with derivatives $\frac{\partial u}{\partial \boldsymbol{x}} = \boldsymbol{G}\boldsymbol{z}$, $\frac{\mathrm{d}\rho}{\mathrm{d}u} = \frac{-1}{s}$, and $\frac{\mathrm{d}s}{\mathrm{d}u} = \frac{u}{s}$. Using these expressions, the first and second derivatives of the 3D hyperbolic heat kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \frac{\rho}{s} e^{-\frac{\rho^2}{\kappa}}$ are given as,

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{z}) &= \left[\frac{\partial u}{\partial \boldsymbol{x}} \frac{\mathrm{d}}{\mathrm{d}u} \rho\right] \frac{1}{s} e^{-\frac{\rho^2}{\kappa}} + \rho \left[\frac{\partial u}{\partial \boldsymbol{x}} \frac{\mathrm{d}}{\mathrm{d}u} s^{-1}\right] e^{-\frac{\rho^2}{\kappa}} + \frac{\rho}{s} \left[\frac{\partial u}{\partial \boldsymbol{x}} \frac{\mathrm{d}\rho}{\mathrm{d}u} \frac{\mathrm{d}}{\mathrm{d}\rho} e^{-\frac{\rho^2}{\kappa}}\right], && (47) \\
&= -\frac{1}{s^2} \boldsymbol{G}\boldsymbol{z} e^{-\frac{\rho^2}{\kappa}} - \frac{\rho u}{s^3} \boldsymbol{G}\boldsymbol{z} e^{-\frac{\rho^2}{\kappa}} + \frac{\rho}{s^2} \boldsymbol{G}\boldsymbol{z} \frac{2\rho}{\kappa} e^{-\frac{\rho^2}{\kappa}}, && (48) \\
&= g(u) e^{-\frac{\rho^2}{\kappa}} \boldsymbol{G}\boldsymbol{z}, && (49) \\
\frac{\partial^2}{\partial \boldsymbol{z} \partial \boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{z}) &= \boldsymbol{G}\boldsymbol{z} \left[\frac{\partial u}{\partial \boldsymbol{z}} \frac{\mathrm{d}}{\mathrm{d}u} g(u)\right] e^{-\frac{\rho^2}{\kappa}} + \boldsymbol{G}\boldsymbol{z} g(u) \left[\frac{\partial u}{\partial \boldsymbol{z}} \frac{\mathrm{d}\rho}{\mathrm{d}u} \frac{\mathrm{d}}{\mathrm{d}\rho} e^{-\frac{\rho^2}{\kappa}}\right] + g(u) e^{-\frac{\rho^2}{\kappa}} \left[\frac{\partial}{\partial \boldsymbol{z}} \boldsymbol{G}\boldsymbol{z}\right], && (50) \\
&= \left[\frac{\mathrm{d}}{\mathrm{d}u} g(u)\right] e^{-\frac{\rho^2}{\kappa}} \boldsymbol{G}\boldsymbol{z}\boldsymbol{x}^\mathsf{T}\boldsymbol{G} + g(u) \frac{-1}{s} \frac{-2\rho}{\kappa} e^{-\frac{\rho^2}{\kappa}} \boldsymbol{G}\boldsymbol{z}\boldsymbol{x}^\mathsf{T}\boldsymbol{G} + g(u) e^{-\frac{\rho^2}{\kappa}} \boldsymbol{G}, && (51) \\
&= \left(h(u) \boldsymbol{G}\boldsymbol{z}\boldsymbol{x}^\mathsf{T}\boldsymbol{G} + g(u)\boldsymbol{G}\right) e^{-\frac{\rho^2}{\kappa}}, && (52) \\
\frac{\partial^2}{\partial \boldsymbol{x} \partial \boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{z}) &= h(u) \boldsymbol{G}\boldsymbol{z}\boldsymbol{z}^\mathsf{T}\boldsymbol{G} e^{-\frac{\rho^2}{\kappa}}. && (53)
\end{aligned}$$

For the third-order derivatives, we consider each column $\boldsymbol{K}_j \in \mathbb{R}^{4\times 1}$ and row $\boldsymbol{K}_i \in \mathbb{R}^{1\times 4}$ of $\boldsymbol{K} = \frac{\partial^2}{\partial \boldsymbol{z} \partial \boldsymbol{x}} k(\boldsymbol{x}, \boldsymbol{z})$ individually, and we proceed as follows,

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{z}} \boldsymbol{K}_j &= \frac{\partial}{\partial \boldsymbol{z}} \left(h(u) \boldsymbol{G}\boldsymbol{z} x_j + g(u) \boldsymbol{e}_j\right) \Delta_{1j} e^{-\frac{\rho^2}{\kappa}}, && (54) \\
&= \left(\left[\frac{\mathrm{d}}{\mathrm{d}u} h(u)\right] \boldsymbol{G}\boldsymbol{z}\boldsymbol{x}^\mathsf{T}\boldsymbol{G} x_j + h(u) \boldsymbol{G} x_j + \left[\frac{\mathrm{d}}{\mathrm{d}u} g(u)\right] \boldsymbol{e}_j \boldsymbol{x}^\mathsf{T}\boldsymbol{G}\right) \Delta_{1j} e^{-\frac{\rho^2}{\kappa}} && (55) \\
&\quad + \left(h(u) \boldsymbol{G}\boldsymbol{z} x_j + g(u) \boldsymbol{e}_j\right) \Delta_{1l} \frac{2\rho}{\kappa s} e^{-\frac{\rho^2}{\kappa}} \boldsymbol{x}^\mathsf{T}\boldsymbol{G}, && (56) \\
&= \left(q(u) \boldsymbol{G}\boldsymbol{z}\boldsymbol{x}^\mathsf{T}\boldsymbol{G} x_j + h(u) \boldsymbol{G} x_j + h(u) \boldsymbol{e}_j \boldsymbol{x}^\mathsf{T}\boldsymbol{G}\right) \Delta_{1j} e^{-\frac{\rho^2}{\kappa}}, && (57) \\
\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{K}_i &= \left(q(u) \boldsymbol{G}\boldsymbol{z}\boldsymbol{x}^\mathsf{T}\boldsymbol{G} z_i + h(u) \boldsymbol{G} z_i + h(u) \boldsymbol{G}\boldsymbol{z}\boldsymbol{e}_i^\mathsf{T}\right) \Delta_{1i} e^{-\frac{\rho^2}{\kappa}}. && (58)
\end{aligned}$$

By stacking the row derivatives $\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{K}_i \in \mathbb{R}^{1\times 4\times 4}$ along the first dimension and the column derivatives $\frac{\partial}{\partial \boldsymbol{z}} \boldsymbol{K}_j \in \mathbb{R}^{4\times 1\times 4}$ along the second, we get the full third-order kernel derivatives $\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{K}, \frac{\partial}{\partial \boldsymbol{z}} \boldsymbol{K} \in \mathbb{R}^{4\times 4\times 4}$. As discussed in Section 4.2, automatic differentiation tools such as PyTorch cannot evaluate the helper functions $g(u)$, $h(u)$, and $q(u)$ as u approaches -1 from below. To address this issue, we replace the general function definitions with their analytical limits, when u gets too close to -1. This is equivalent to the hyperbolic distance between the two kernel inputs falling below a predefined threshold value $\rho = \operatorname{arccosh}(-u) = \mathrm{d}_{\mathbb{H}^{D_x}_{\mathcal{L}}}(\boldsymbol{x}, \boldsymbol{z}) \leq 1\mathrm{e}{-4}$. The analytical

limit expressions for the helper functions are given as

$$\lim_{u\to-1^-} g(u) = \lim_{u\to-1^-}\left(\frac{2\rho^2}{\kappa s^2} - \frac{1}{s^2} - \frac{u\rho}{s^3}\right) = \frac{2}{\kappa} + \frac{1}{3}\,, \tag{59}$$

$$\lim_{u\to-1^-} h(u) = \lim_{u\to-1^-}\left(\frac{\mathrm{d}}{\mathrm{d}u}g(u) + \frac{g(u)2\rho}{\kappa s}\right) = \frac{4}{\kappa^2} + \frac{6}{3\kappa} + \frac{4}{15}\,, \tag{60}$$

$$\lim_{u\to-1^-} q(u) = \lim_{u\to-1^-}\left(\frac{\mathrm{d}}{\mathrm{d}u}h(u) + \frac{h(u)2\rho}{\kappa s}\right) = \frac{8}{\kappa^3} + \frac{8}{\kappa^2} + \frac{14}{5\kappa} + \frac{12}{35}\,. \tag{61}$$

C Experimental Details and Additional Results

Apps. C.1-C.4 provide additional details regarding the data and GPHLVM training process for each of the experiments described in Section 5. App C.5 provides additional insights into the hand grasps experiment of Section 5.4.

C.1 C-shape

The proof-of-concept experiment on the C-shape dataset was designed as a minimalist example with a simple setup. 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, these datapoints are used as both the latent points and the observations, so that $\boldsymbol{X} = \boldsymbol{Y} \in \mathbb{R}^{1000\times 2}$. 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 $\boldsymbol{x}_{\mathcal{L}} = \frac{1}{1-\|\boldsymbol{x}\|^2}\begin{bmatrix}1+\|\boldsymbol{x}\|^2 & 2\boldsymbol{x}^\mathsf{T}\end{bmatrix}^\mathsf{T} \in \mathbb{H}^2_{\mathcal{L}}$. In this proof-of-concept experiment, we manually set the kernel variance, lengthscale, and noise variance of both GPLVM and GPHLVM to $(\tau, \kappa, \sigma_y^2) = (0.7, 0.15, 0.69)$. Additionally, for the GPHLVM, we used the 2D hyperbolic heat kernel with $L = 3000$ rejection samples. We used constant zero mean for both models.

The start- and end-points of the geodesics shown in Figure 1 correspond to one of the C-shape latent points. The geodesics are represented by $M = 25$ regularly-spaced points. Geodesics on the base Euclidean manifold correspond to straight lines from $\boldsymbol{x}_0$ to $\boldsymbol{x}_{M-1}$. In the hyperbolic case, the geodesics on the base manifold are given as $\boldsymbol{x}_i = \mathrm{Exp}_{\boldsymbol{x}_0}(t_i \mathrm{Log}_{\boldsymbol{x}_0}(\boldsymbol{x}_{M-1}))$ with $t_i = i/(M-1)$ and $i = \{0, ..., M-1\}$. The geodesics on the base manifolds were used as initialization for the pullback geodesic optimization. To compute the pullback geodesic, we applied Riemannian Adam for 200 steps with a learning rate of 0.005, optimizing the combined loss function (22) based on both curve energy and spline energy, as introduced in Section 3.3. In this experiment, the curve energy and spline energy were weighted equally, i.e., $\lambda = 1$ in Equation (21).

Finally, we generated a 110×110 grid of points to visualize the volume of the pullback metric, filtering out those points lying outside the unit circle in the hyperbolic case. Then, we evaluated the pullback metric tensor at each remaining grid point. In the Euclidean case, we computed the volume at each grid point $\boldsymbol{x}^*$ by evaluating the determinant of the 2×2 pullback metric tensor, $\sqrt{\det(\boldsymbol{G}_{\boldsymbol{x}^*})}$. 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 as the product of the two nonzero eigenvalues.

C.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 embeddings are initialized using PCA.

C.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 multicellular robots following the hierarchical approach of Dong et al. (2024), 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 our GPHLVM as proposed by Jaquier et al. (2024). This is achieved by leveraging the stress loss,

$$\ell_{\text{stress}}(\boldsymbol{X}) = \sum_{i<j} \left(\mathrm{d}_{\mathbb{G}}(c_i, c_j) - \mathrm{d}_{\mathcal{M}}(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)^2, \tag{62}$$

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 is achieved by solving,

$$\underset{\boldsymbol{X}}{\operatorname{argmax}} \log p(\boldsymbol{Y} \mid \boldsymbol{X}, \Theta) - \gamma \bar{\ell}_{\text{stress}}(\boldsymbol{X}), \tag{63}$$

where γ is a parameter balancing the two losses and $\bar{\ell}_{\text{stress}}(\boldsymbol{X})$ is the stress loss averaged over the embeddings. We set $\gamma = 6000$. Moreover, 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.

C.4 Hand Grasps Generation

For the hand grasp experiment, we used data from the KIT Whole-Body Motion Database (Mandery et al., 2016). The dataset consists of 38 trajectories, with motion recordings from two human subjects (IDs 2122 and 2123). Each subject performed 19 different grasp types[2] of the GRASP taxonomy (Feix et al., 2016). 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 model: *(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 model. 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 $\boldsymbol{Y} \in \mathbb{R}^{1321 \times 24}$.

In contrast to the previous experiments, this experiment involves trajectory data. To preserve the trajectory structure during training, we incorporated a dynamics prior $p(\boldsymbol{X}_{2:N} \mid \boldsymbol{X}_{1:N-1})$, similar to Wang et al. (2008), but using the wrapped Gaussian distribution (36). We also used back constraints $\boldsymbol{X} = k(\boldsymbol{Y}, \boldsymbol{Y})\boldsymbol{C}$, 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 lead node of the quantitative taxonomy of hand grasps (Stival et al., 2019). 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, $\mathrm{d}_{\mathbb{G}}(c_i, c_j)$. 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. (2024). This is achieved by leveraging the stress loss (62) 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,

$$\underset{\boldsymbol{X}}{\operatorname{argmax}} \log p(\boldsymbol{Y} \mid \boldsymbol{X}, \Theta) + 2\frac{D_y}{D_x} \log p(\boldsymbol{X}_{2:N} \mid \boldsymbol{X}_{1:N-1}) - D_y \ell_{\text{stress}}(\boldsymbol{X}). \tag{64}$$

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

[2]The three-finger sphere grasp of the GRASP taxonomy (Feix et al., 2016) is missing from the dataset.

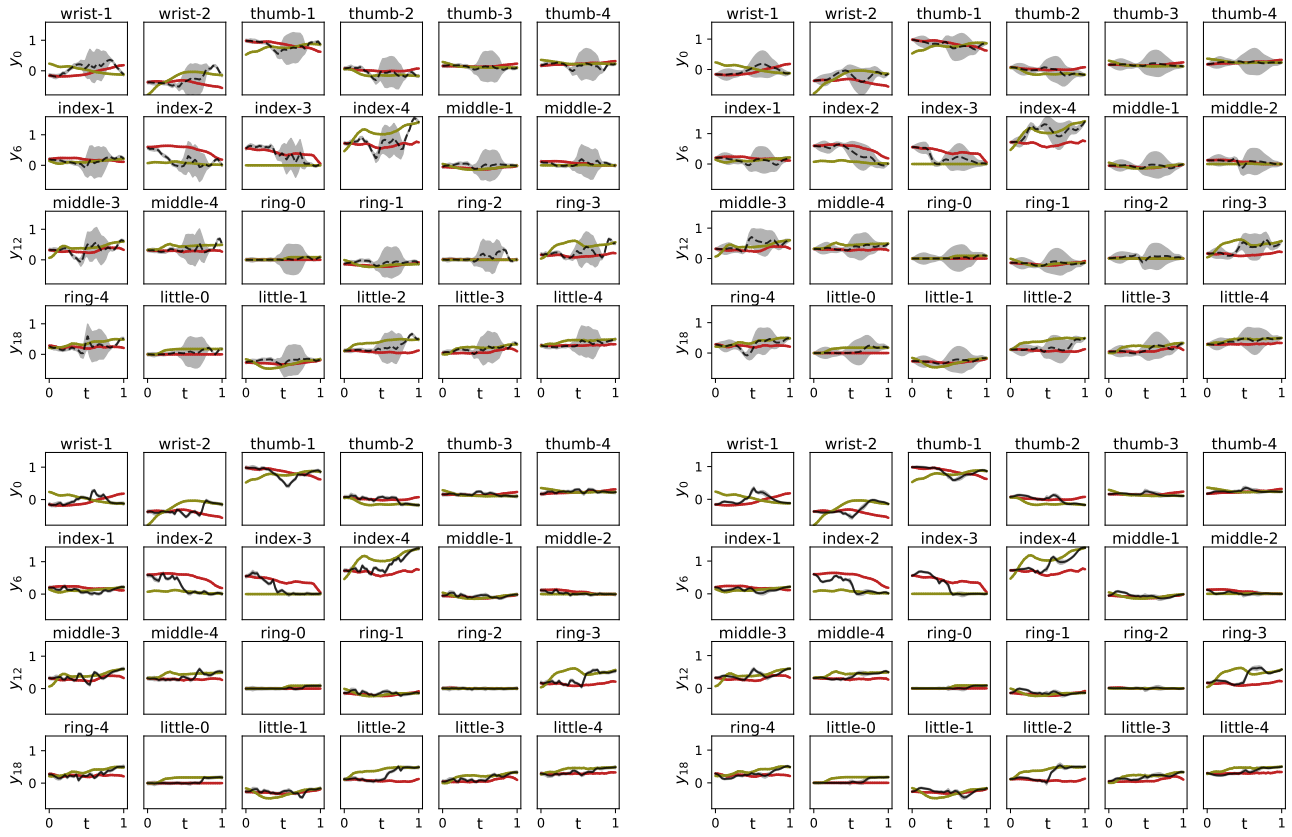


Figure 6: Time-series plots of all 24 dimensions of the joint space introduced in Figure 4. The left and right columns show the decoded geodesics of the 2D and 3D latent spaces. The top and bottom rows show the mean of the decoded hyperbolic and pullback geodesics (- - and —) and their uncertainty as a gray envelope. For reference, the same training trajectory to the spherical grasp (—) and reversed training trajectory from the ring grasp (—) are included.

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.,

$$\boldsymbol{X}_{\text{init}} = \min_{\boldsymbol{X}} \ell_{\text{stress}}(\boldsymbol{X}). \tag{65}$$

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 $\lambda = 100$.

C.5 Additional Hand Grasps Results

This section provides further insights into the hand grasps experiment described in Section 5.4. First, we present additional visualizations of the decoded geodesics. While Figure 4 only showed 3 dimensions of the joint space, Figure 6 depicts the decoded geodesics across all 24 dimensions, i.e., over the 24 degrees of freedom (DoF) of the hand model introduced in Mandery et al. (2016). This model counts two 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.

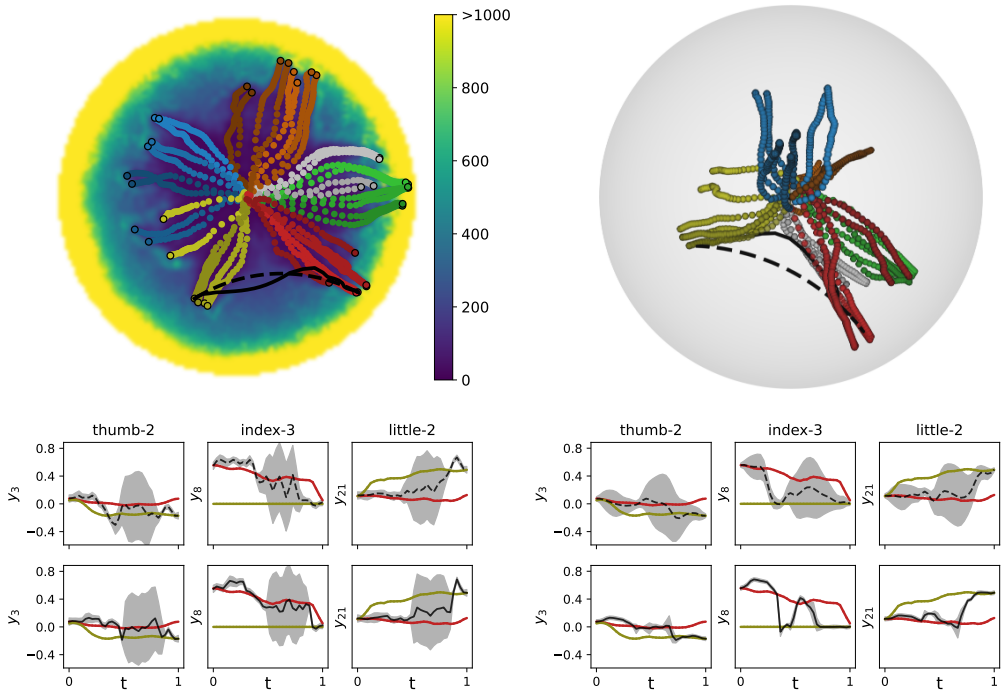


Figure 7: Hand grasps transition when initializing the geodesic optimization with the hyperbolic geodesic. As in Figure 4 the optimized geodesic on the pullback metric (——) and the hyperbolic geodesic (- -), transition from a ring grasp (●) to a spherical grasp (●). *Top*: 2D and 3D latent spaces of the trained models. *Bottom*: Time-series plots of 3 dimensions of the joint space for the decoded hyperbolic geodesic (- -) and pullback geodesic (——).

Second, we discuss the influence of initialization on optimizing the pullback geodesics. As described in Section C.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, Figure 7 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 Figure 4. 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 heat 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 Figure 4.