PARAMETRIZING PRODUCT SHAPE MANIFOLDS BY COMPOSITE NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Parametrizations of data manifolds in shape spaces can be computed using the rich toolbox of Riemannian geometry. This, however, often comes with high computational costs, which raises the question if one can learn an efficient neural network approximation. We show that this is indeed possible for shape spaces with a special product structure, namely those smoothly approximable by a direct sum of low-dimensional manifolds. Our proposed architecture leverages this structure by separately learning approximations for the low-dimensional factors and a subsequent combination. After developing the approach as a general framework, we apply it to a shape space of triangular surfaces. Here, typical examples of data manifolds are given through datasets of articulated models and can be factorized, for example, by a Sparse Principal Geodesic Analysis (SPGA). We demonstrate the effectiveness of our proposed approach with experiments on synthetic data as well as manifolds extracted from data via SPGA.

1 INTRODUCTION

Modeling collections of shapes as data on Riemannian manifolds has enabled the usage of a rich set of mathematical tools in areas such as computer graphics and vision, medical imaging, computational biology, and computational anatomy. For example, Principal Geodesic Analysis (PGA), a generalization of Principal Component Analysis (PCA), can be used to parametrize nonlinear submanifolds that approximate given data points while preserving structure of the data such as its invariance to rigid motion. The evaluation of such a parametrization, however, typically comes at a high computational cost as the Riemannian exponential, which maps infinitesimal shape variations to shapes, needs to be evaluated.

This motivates trying to learn an efficient approximation for these parametrizations. Direct application of deep neural networks, however, proves ineffective for high-dimensional spaces with strongly nonlinear variations. Therefore, we consider more structured shape manifolds, namely, we assume that they can be approximated by an affine sum of low-dimensional submanifolds. In computer graphics, typical examples of data manifolds are given through datasets of articulated models, *e.g.* human bodies, faces or hands. Then, the desired structure of an affine sum of factor manifolds can be produced, for example, by a Sparse Principal Geodesic Analysis (SPGA). Motivated by this, we exploit the data manifolds' approximability with such affine sums: We separately approximate the exponential map on the factor manifolds by fully connected neural networks and the subsequent combination of factors by a convolutional network to yield our approximate parametrization.

We develop our approach focusing on the shape space of discrete shells, where shapes are given by triangle meshes and the manifold is equipped with an elasticity-based metric. In principle, our approach is also applicable to other shape spaces such as manifolds of images, and we will include remarks on how we propose this could work. We evaluate our approach with experiments on data manifolds of triangle meshes, both synthetic ones and ones extracted from data via SPGA, and we demonstrate that the proposed composite network architecture outperforms a monolithic fully connected network architecture as well as an approach based on the affine combination of the factors.

Contributions In summary, the contributions of this paper are

- combining the Riemannian exponential map on shape spaces and neural network methodology for the efficient parametrization of shape space data manifolds,
- demonstrating the applicability of such an approach for data manifolds which can be smoothly approximated via direct sums of low-dimensional submanifolds,
- using a combination of fully connected neural networks for the factorwise Riemannian exponential maps and a convolutional network to couple them,
- verifying that such a setup works well with existing methods to construct product manifolds, such as Sparse Principal Geodesic Analysis, and
- showing that the composite network architecture outperforms alternative approaches.

2 Related Work

Shape Spaces Shape spaces are manifolds in which each point is a shape, *e.g.*, a triangle mesh or an image. A Riemann metric on such a space provides means to define distances between shapes, to interpolate between shapes by computing shortest geodesic paths, and to explore the space by constructing the geodesic curve starting from a point into a given direction. Shape spaces have proven useful for applications in areas such as computer graphics (Kilian et al., 2007; Heeren et al., 2012; Wang et al., 2018) and vision (Heeren et al., 2018; Xie et al., 2014), medical imaging (Kurtek et al., 2011b; Samir et al., 2014; Kurtek et al., 2016; Bharath et al., 2018), computational biology (Laga et al., 2014), and computational anatomy (Miller et al., 2006; Pennec, 2009; Kurtek et al., 2011a). For an introduction to the topic, we refer to the textbook of Younes (2010).

Shape Space of Meshes Triangle meshes are widely used to represent shapes in computer graphics and vision. Riemannian metrics on shape spaces of triangle meshes can be defined geometrically, using norms on function spaces on the meshes (Kilian et al., 2007), or physics-based, considering the meshes as thin shells and measuring the dissipation required to deform the shells (Heeren et al., 2012; 2014). The computation of geodesic curves in these spaces requires numerically solving high-dimensional nonlinear variational value problems, which can be costly. For shape interpolation problems, model reduction methods can be used to efficiently find approximate solutions (Brandt et al., 2016; von Radziewsky et al., 2016).

Statistics in Shape Spaces Data in a Riemannian shape spaces can be analyzed using Principal Geodesic Analysis (PGA) (Fletcher et al., 2004; Pennec, 2006). Analogous to principal component analysis (PCA) for data in Euclidean spaces, PGA can construct low-dimensional latent representations that preserve much of the variability in the data. This is achieved by mapping the data with a non-linear mapping, the Riemannian logarithmic map, from the manifold to a linear space, the tangential space at the mean shape, and computing a PCA there. Latent variables of the PCA are then mapped with the inverse mapping, the Riemannian exponential map, onto the manifold, so that the latent space describes a submanifold of the shape space. A PGA in shape spaces of meshes was introduced in (Heeren et al., 2018) and used to obtain a low-dimensional, nonlinear, rigid body motion invariant description of shape variation from data.

Sparse PGA While PCA modes involve all variables of the data, Sparse Principal Component Analysis (Zou et al., 2006) constructs modes that involve just few variables. This is achieved by integrating a sparsity encouraging term to the objective that defines the modes. Based on this idea, Neumann et al. (2013) proposed a scheme for extracting Sparse Localized Deformation Components (SPLOCS) from a dataset of non-rigid shapes. Since SPLOCS are linear modes, they are well-suited to describe small deformations such as face motions accurately. To increase the range of deformations and to compensate linearization artifacts Huang et al. (2014) integrated SPLOCS with gradient-domain techniques and Wang et al. (2017; 2021) with edge lengths and dihedral angles. In (Sassen et al., 2020b), a Sparse Principal Geodesic Analysis (SPGA) was introduced. Similar to PGA, the SPGA modes are nonlinear and rigid motion invariant. On top of that, however, the SPGA modes describe localized deformations. Due to the localization, many pairs of SPGA modes have

disjoint support and are therefore independent of each other. We want to take advantage of this property to effectively learn the reconstruction of points in the manifold from their latent representation through an adapted network structure.

Product Manifolds This work is focused on the parametrization of the product manifold structures obtained from SPGA. Alternative approaches for extracting product structures of data manifolds include the approach of Fumero et al. (2021), which finds a product structure of a manifold based on a geometric notion of disentanglement, the Geometric Manifold Component Estimator presented in (Pfau et al., 2020), which uses a Lie group of transformations to generate a symmetry-based disentanglement of data manifolds, and the approach of Zhang et al. (2021), which decomposes the eigenfunctions of the Laplace–Beltrami operator of a manifold in order to find a product structure in the manifold.

3 Preliminaries and Notation

We introduce step by step the background necessary to understand the context of our work.

Riemannian Shape Space A shape space is a manifold $\mathcal{S} \subset \mathbb{R}^n$ whose elements are shapes. These could, for example, be images, curves, or surfaces described in various ways. Endowing such a shape manifold with the structure of a *Riemannian* manifold, *i.e.* a (smoothly) z-dependent inner product g_z on the tangent space $T_z\mathcal{S}$ at each point $z \in \mathcal{S}$, provides us with a rich set of geometric tools. For example, we can use geodesics $c\colon [0,1] \to \mathcal{S}$, *i.e.* arc length parametrized locally shortest paths, as mathematical formulation of shape interpolation. The Riemannian logarithm $\log_z \tilde{z} \in T_z\mathcal{S}$ is then defined as the time derivative $\dot{c}(0)$ of the geodesic c interpolating between $c(0) = z$ and $c(1) = \tilde{z}$. This allows to interpret the tangent space $T_z\mathcal{S}$ as the linear space of infinitesimal shape variations. Lastly, the Riemannian exponential map is the inverse of the logarithm, which means that for a tangent vector $v \in T_z\mathcal{S}$ one 'shoots' a geodesic in its direction, *i.e.* constructs a geodesic curve c with initial velocity v to obtain $\exp_z v \coloneqq c(1) \in \mathcal{S}$. The exponential map allows to transfer operations from infinitesimal shape variations back to actual shapes.

Principal Geodesic Analysis With these tools at hand, one can use Principal Geodesic Analysis (PGA) to compute submanifolds of the shape space approximating given data points $\{z^i\} \subset \mathcal{S}$: One first computes their Riemannian center of mass $\bar{z}$, *i.e.* the point with minimal sum of squared distances to all data points. Then one computes the logarithms $v^i = \log_{\bar{z}} z^i$ and thus linearizes the approximation problem at the center of mass by passing to the tangent space $T_{\bar{z}}\mathcal{S}$. In $T_{\bar{z}}\mathcal{S}$ one uses classical Principal Component Analysis (PCA) to compute the m dominant modes $\{u^j\}$, whose span $\mathcal{U}$ is the best m-dimensional subspace approximating the logarithms v^i. Then the submanifold $\mathcal{M}$ approximating the data points is parametrized by the exponential map, *i.e.* $\mathcal{M} \coloneqq \exp_{\bar{z}} \mathcal{U}$.

Sparse Principal Geodesic Analysis The coordinates on Riemannian shape spaces, *e.g.* pixel values, often correspond to different spatial locations of the shape, *e.g.* pixel positions, such that it makes sense to consider their support and sparsity: Sassen et al. (2020b) introduced Sparse Principal Geodesic Analysis (SPGA) to compute spatially localized dominant modes with widely disjoint supports. Those modes are easier to interpret semantically and in addition allow for efficient approximations of the exponential map. SPGA is performed by adding an appropriate sparsity-inducing regularization functional $\mathcal{R}$ to the variational formulation of PCA to compute such sparse deformation modes. The concrete choice of $\mathcal{R}$ depends on the specific shape space. Hence, for a given set $V \in \mathbb{R}^{n\times K}$ of K logarithms, the SPGA problem to compute the first m dominant modes $U \in \mathbb{R}^{n\times m}$ reads

$$\begin{aligned} \underset{\substack{U\in\mathbb{R}^{n\times m}\\ W\in\mathbb{R}^{m\times K}}}{\text{minimize}} \quad & \|V - UW\|_g^2 + \lambda\,\mathcal{R}(U) \\ \text{subject to} \quad & u_j \in T_{\bar{z}}\mathcal{S} \;\text{ and } |w_j|_\infty \le 1 \text{ for } j \in \{1,\dots,m\}. \end{aligned} \tag{1}$$

The bound on the magnitude of the weights ensures that the coordinates of the u_j do not shrink while the weights grow inverse proportionally. As before, the manifold approximating the data points is then parametrized using the exponential map. It can be equipped with a product structure by grouping the modes and applying the exponential map to the corresponding subspaces. This will be elucidated more in section 4.

NRIC Manifold As shapes we will consider triangular surfaces with fixed connectivity, *i.e.* with shared sets of vertices $\mathcal{V}$, edges $\mathcal{E}$, and faces $\mathcal{F}$. For vertex positions $X \in \mathbb{R}^{3|\mathcal{V}|}$, we denote by $l(X) = (l_e(X))_{e\in\mathcal{E}}$ the vector of edge lengths and by $\theta(X) = (\theta_e(X))_{e\in\mathcal{E}}$ the vector of dihedral angles. We consider the vectors $z(X) = (l(X), \theta(X)) \in \mathbb{R}^{2|\mathcal{E}|}$ combining both. The manifold of all $z \in \mathbb{R}^{2|\mathcal{E}|}$ corresponding to immersed triangular surfaces is given by

$$\mathcal{S} \coloneqq \left\{ z \in \mathbb{R}^{2|\mathcal{E}|} \,\middle|\, \mathcal{T}(z) > 0,\ \mathcal{Q}(z) = 0 \right\}, \tag{2}$$

where $\mathcal{T}(z) > 0$ encodes the triangle inequalities and $\mathcal{Q}(z) = 0$ are the discrete integrability conditions from Wang et al. (2012), see also Sassen et al. (2020a) for more details. $\mathcal{S}$ is called the NRIC manifold, short for Nonlinear Rotation-Invariant Coordinates, and can be equipped with an elasticity-based Riemannian metric (Heeren et al., 2014). To evaluate the logarithm and exponential map, we use the time-discretization developed by Rumpf & Wirth (2015).

To obtain vertex positions for given lengths and angles, we use the nonlinear least-squares method from Fröhlich & Botsch (2011) based on L^p-norms to measure the difference between arbitrary z,

$$\|z^a - z^b\|_{p,\bar{z}}^p \coloneqq \sum_{e\in\mathcal{E}} w_{e,l}^p |l_e^b - l_e^a|^p + \eta \sum_{e\in\mathcal{E}} w_{e,\theta}^p |\theta_e^b - \theta_e^a|^p. \tag{3}$$

The weights are computed from a reference shape $\bar{z}$ with vertex positions $\bar{X}$, *e.g.* the center of mass from above, as $w_{e,l} = l_e(\bar{X})^{-1}$ and $w_{e,\theta} = l_e(\bar{X}) a_e(\bar{X})^{-1/2}$, where a_e is (one third of) the area of both faces adjacent to e. The bending weight η is the same as used in the elasticity-based metric.

To apply SPGA to this shape space, we need to specify the sparsity-inducing regularization $\mathcal{R}$. Sassen et al. (2020b) observed that the simple mode-wise L^1-norms

$$\mathcal{R}(U) = \sum_{j=1}^{m} \|u_j\|_1 \tag{4}$$

suffice due to the natural connection of NRIC variation with elastic distortions.

Neural Networks We indicate functions that are implemented as neural networks by the superscript θ as in φ^θ. This θ represents the network parameters and is the same for all occurring networks, with the implicit convention that different networks depend on different subsets of these parameters.

We denote by $\mathrm{MLP}_\rho^\theta[N_1, \ldots, N_T]$ a fully-connected network with layer sizes $N_1, \ldots, N_T$, the nonlinear activation function $\rho\colon \mathbb{R} \to \mathbb{R}$ after each layer, and parameters θ. For graph convolutional networks we adapt the approach by Kipf & Welling (2017) to data $z_e^0 \in \mathbb{R}^{N_0}$ on edges e: The tth layer is given by

$$z_e^{t+1} = \rho\left(W_1^{t+1} z_e^t + W_2^{t+1} \sum_{\tilde{e}\in\mathcal{N}(e)} z_{\tilde{e}}^t + b^{t+1} \right) \tag{5}$$

where $W_1^{t+1} \in \mathbb{R}^{N_{t+1}\times N_t}$ and $W_2^{t+1} \in \mathbb{R}^{N_{t+1}\times N_t}$ are small matrices with learnable parameters and $b^{t+1} \in \mathbb{R}^{N_{t+1}}$ is the bias, all stored in the parameters θ. We define the neighborhood of an edge in a triangle mesh as $\mathcal{N}(e) \coloneqq \{\tilde{e} \in \mathcal{E} \mid e \text{ and } \tilde{e} \text{ share a vertex}\}$. We used the Exponential Linear Unit (Clevert et al., 2016) as activation function in all our experiments.

4 Composite Network Approximation

As previously explained, we want to learn an efficient parametrization $\Phi\colon \mathbb{R}^m \to \mathcal{M}$ of an m-dimensional Riemannian data manifold $\mathcal{M}$. Below, we will explain our structural assumptions to achieve this, detail how we include them in the network architecture and training, and finally discuss their applicability to practical examples.

Structural Assumptions One can think of the parametrization as the decoder part of an autoencoder, however, for the purpose of this article we want to be more specific: We would like the parametrization to have a geometric meaning, so we will focus on the situation in which Φ should encode the Riemannian exponential map at some point $z \in \mathcal{M}$. In principle, our approach can also

be combined with alternative concepts in which no ground truth parametrization is available, but this setting allows a particularly simple quantification of the results.

If no additional structure of the data manifold is known, one can of course not improve over computing the Riemannian exponential directly or approximating it by some standard neural network. In contrast, we consider the case where a specific structure is known. For the purpose of our article, this structure is a priori given, that is, we do not study how such a structure can be found or obtained, though we give an example for a corresponding procedure on Riemannian shape spaces later on. We require that the data manifold $\mathcal{M}$ to be parametrized is embedded in some $\mathbb{R}^n$ for possibly large n. The structure of $\mathcal{M}$ we want to exploit has three components:

(0) **Correlation** We assume a structural correlation between the different coordinates, *e.g.* graph-neighbour relations for triangular meshes or pixel-neighbour relations for image data, so that convolutional networks can be applied.

(1) **Factorization** We assume that the manifold can be smoothly approximated by a product of much lower-dimensional manifolds $\mathcal{M}_1, \mathcal{M}_2, \ldots \mathcal{M}_L$, which we will parametrize separately. Thereby, we exploit that the necessary network size as well as the required training effort decrease with smaller manifold dimension: For m-dimensional manifolds the network size should scale at least linearly in m, while the required training set and thus also training time will scale exponentially in m.

(2) **Combination** It is not sufficient that the single factor manifolds are easy to parametrize since a generic point on $\mathcal{M}$ has components in all factors. Thus, we assume that the direct sum of all factors

$$\mathcal{M}_1 \oplus_z \ldots \oplus_z \mathcal{M}_L \coloneqq \{z + (z_1 - z) + \ldots + (z_L - z) \,|\, z_1 \in \mathcal{M}_1, \ldots, z_L \in \mathcal{M}_L\}$$

for some $z \in \mathcal{M}$ already approximates the actual manifold $\mathcal{M}$ with a (possibly large, but) very smooth approximation error. This will ensure that a suitable map from $(z_1 - z, \ldots, z_L - z)$ to $\mathcal{M}$ can be efficiently learned.

The toy shape manifold $\mathcal{M}$ of tori from Figure 1 has the flat metric of $S^1 \times S^1 \times \mathbb{T}^2$ (the factors representing the orientation of the longitudinal and latitudinal ellipsoidal cross-sections as well as a bump position, see subsection 5.1) and thus satisfies condition (1). Condition (2) holds since each torus is represented as NRIC: For instance, the creation of the bump (which corresponds to changing the position in $\mathbb{T}^2$) is well described by simply adding fixed numbers in the right places of the edge length vector $l(X)$. However, the direct sum is indeed only an approximation – otherwise there would be no error in Figure 1 (green), but one clearly sees a remnant of the bump from the reference shape.

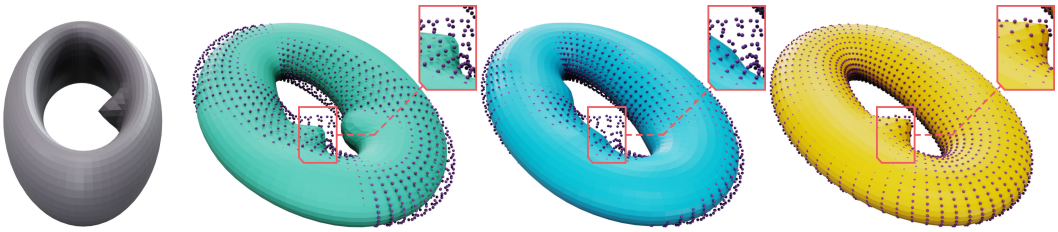

Figure 1: Approximation of the Freaky Torus. We show the reference shape in grey, the approximation by affine combination of exact factors in green, by a monolithic network in blue, and by our composite network in yellow. The correct vertex positions are shown as purple points.

Network Representation and Training Before we discuss the previous conditions, let us describe how we exploit them in our network architecture and training procedure. To learn the parametrization $\Phi\colon \mathbb{R}^m \to \mathcal{M}$, we decompose it as

$$\Phi = \Psi^\theta \circ (\psi_1^\theta, \ldots, \psi_L^\theta),$$

where $\psi_l^\theta\colon \mathbb{R}^{m_l} \to \mathbb{R}^n$ is the parametrization of the m_l-dimensional factor manifold $\mathcal{M}_l$ and $\Psi^\theta\colon (\mathbb{R}^n)^L \to \mathbb{R}^n$ is the combination of the single factors, behaving approximately like $(x_1, \ldots, x_n) \mapsto z + (x_1 - z) + \ldots + (x_n - z)$, see Figure 2.

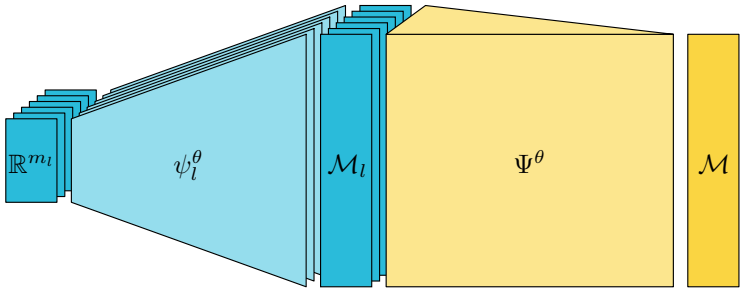

Figure 2: Structure of composite network.

The maps ψ_l^θ will be fully connected neural networks. Because they approximate the smooth exponential map on rather low-dimensional spaces, they are expected to achieve a high approximation quality that is typically stable under variation of the concrete network architecture (number and size of layers). For the maps Ψ^θ, we exploit that they operate on a structured domain, *e.g.* a mesh, and thus they will be convolutional neural networks. This allows for efficient training and storage even though they operate on high-dimensional data. One could alternatively also use fully connected networks for Ψ^θ, but the observed quality of the results was similar despite significantly increased memory requirements to train, store, and evaluate.

These networks are trained separately: To train the map ψ_l^θ, we first fix a basis $e_1^l, \ldots, e_{m_l}^l$ of the Riemannian logarithm of $\mathcal{M}_j$ with respect to z, which naturally induces the map

$$\omega_l \colon \mathbb{R}^{m_l} \to T_z\mathcal{M}, \quad \omega_l(a) = a_1 e_1^l + \ldots + a_{m_l} e_{m_l}^l.$$

We then consider a random sample $S^l \subset \mathbb{R}^{m_l}$ as training data (for instance normally distributed or uniformly on a ball) and minimize the loss function

$$\mathcal{J}_l(\theta) = \frac{1}{|S^l|} \sum_{a \in S^l} \|\exp_z(\omega_l(a)) - \psi_l^\theta(a)\|^2$$

in some norm $\|\cdot\|$ depending on the application. To train the map Ψ^θ, we subsequently consider a random training set $S \subset \mathbb{R}^{m_1} \times \ldots \times \mathbb{R}^{m_L} = \mathbb{R}^m$ and minimize the loss function

$$\mathcal{J}(\theta) = \frac{1}{|S|} \sum_{(a^1,\ldots,a^L) \in S} \|\exp_z(\omega_1(a^1) + \ldots + \omega_L(a^L)) - \Psi^\theta(\psi_1^\theta(a^1), \ldots, \psi_L^\theta(a^L)\|^2,$$

where we may or may not keep the maps ψ_l^θ fixed.

Applicability of Assumptions While condition (0) essentially has to hold for any approach employing neural networks, conditions (1) and (2) are specific to our approach. Condition (1) expresses that the *intrinsic geometry* of the data manifold $\mathcal{M}$ has a simplifying structure, while condition (2) is about the *extrinsic geometry* of how $\mathcal{M}$ is embedded in $\mathbb{R}^n$ – only both conditions together characterize a structure that can efficiently be learned. Typical situations where our conditions hold include the following two examples:

- The different factor manifolds correspond to different spatial regions. For instance, images may sometimes be partitioned into different regions that can vary more or less independently of each other. If the regions are fully disjoint, the manifold $\mathcal{M}$ can exactly be written as a direct sum of factor manifolds; if on the other hand the regions slightly overlap, then this is only fulfilled approximately. Examples in shape spaces are where the single factor manifolds correspond to shape variations with disjoint support such as articulation of the different extremities of a character. Our computational examples mostly belong to this category.
- On shapes one can also often observe variations at different length scales that are independent of each other, for example, geometric texture on small length scales versus global shape variations at large length scales. The above toy example of the torus, in which the small bump is more or less independent of the global shape variations, belongs to this category.

As mentioned before, we assume the product manifold structure to be given a priori (in the torus example, for instance, it was given by design of the data manifold). The product manifold structure would typically be a result of some data manifold analysis, for example from disentanglement learning. In the setting of Riemannian shape spaces, one way to obtain this structure is the SPGA introduced earlier: It makes use of the decomposition into independent product manifolds due to disjoint support of the corresponding shape variations.

5 Experiments and Applications

In this section, we present experimental results on the aforementioned synthetic shape manifold of deformed tori and manifolds extracted via SPGA. We will compare our method to approaches based on (Sassen et al., 2020b) and a straightforward approximation of the parametrization by a single fully connected network. To quantify the approximation quality of different approaches, we use the coefficient of determination R^2. For approximations $\tilde{z}_i$ of NRIC z_i with mean $\bar{z}$, it is defined as

$$R^2(z, \tilde{z}) = 1 - \frac{\sum_i \|\tilde{z}_i - z_i\|_{2,\bar{z}}^2}{\sum_i \|z_i - \bar{z}\|_{2,\bar{z}}^2}.$$

From a statistical point of view, it quantifies the proportion of variation of the data that is explainable by a given model. This means an R^2 of one is optimal, the smaller it is the worse is the approximation, and a negative R^2 means that the model is worse than simply using the mean.

Training & Implementation We used Adam (Kingma & Ba, 2015) as descent method for training all networks, where the initial learning rate was 10^{-3} and was reduced by a factor of 10 every time the loss did not decrease for multiple iterations. For regularization, we used batch normalization after each layer and a moderate dropout regularization ($p = 0.1$) after each convolutional layer. We implemented the neural networks in PyTorch (Paszke et al., 2019) using the PyTorch Geometric library (Fey & Lenssen, 2019). The tools for the NRIC manifold were implemented in C++ based on OpenMesh (Botsch et al., 2002), where we use the Eigen library (Guennebaud et al., 2010) for numerical linear algebra. We follow an approach similiar to Kilian et al. (2007) to perform all computations on a coarsened mesh and prolongate solutions to a fine one only for visualizations.

5.1 Synthetic Data: Freaky Torus

For the *Freaky Torus* dataset, we construct a synthetic shape space with factors $S^1 \times S^1 \times \mathbb{T}^2$, where $\mathbb{T}^2$ refers to the flat 2-dimensional torus. It is realized in NRIC by (i) deforming the two cross-sectional circles of a torus to ellipses of fixed aspect ratio and orientation controlled by the first two S^1 factors and (ii) growing a bump in normal direction whose position is controlled by the last $\mathbb{T}^2$ factor. These torus deformations are applied to a regular mesh of a regular torus embedded in $\mathbb{R}^3$, and the deformed meshes' NRIC are extracted to obtain our datapoints. We used a mesh with 2048 vertices and uniformly drew 1000 samples from $S^1 \times S^1 \times \mathbb{T}^2$.[1] More details on the data generation can be found in Appendix A.

Figure 1 shows that a single fully connected network struggles with approximating the high frequency detail of the bump, while our composite network is able to handle this well. This can also be observed in the approximation quality quantified using the R^2. The composite network achieves an R^2 of 0.99 and the monolithic network one of 0.95. This difference may sound small, however, this is because the bump is a detail and the error is dominated by the overall shape of the torus.

5.2 Application: SPGA Manifolds

Lastly, we report the results of applying our method to shape manifolds whose approximate product structure is found with the help of SPGA. To this end, we repeat three of the examples discussed by Sassen et al. (2020b) and one new dataset. The repeated examples are a humanoid dataset from (Anguelov et al., 2005), a dataset of face meshes from (Zhang et al., 2004), and a set of hand meshes from (Yeh et al., 2011). For the new example, we examine a humanoid dataset based on SMPL-X (Pavlakos et al., 2019), where we consider their *expressive hands and faces* (EHF) dataset,

[1] The data and the generating code can be found at [to be added in final publication]

containing 100 shapes, and 49 additional shapes from the SMPL+H dataset, which feature more expressive arm and leg movements, adding to a total of 149 input shapes. We interpret all shapes as elements of the nonlinear NRIC space so that this small amount of data points already suffices to span a high-dimensional, nonlinear NRIC submanifold that serves as our data manifold.

Based on this data, we follow the numerical approach of Sassen et al. (2020b) to solve (1) and thereby compute the sparse tangent modes. We report the chosen number m of included modes in Table 1, where we used the same number as Sassen et al. (2020b) for the repeated examples. To factor the resulting data manifold, we again follow the approach outlined by Sassen et al. (2020b), which is a clustering based on the spatial overlap of the modes. For the hand and face examples, we choose the same number of factors L, while for the SCAPE example we decreased the number to account for the possibility to handle higher-dimensional factors with our method. Our choices are again documented in Table 1. Each cluster then spans exactly one of the factor manifolds and the range of their dimensions is also reported in said table.

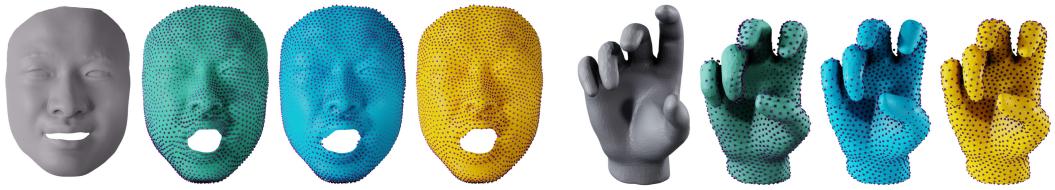

Figure 3: Examples from the face and hands datasets. We use the same colors as in Figure 1.

Data Generation Next, we sample the exponential map on the space spanned by the SPGA modes to generate the training (and test) data. To this end, we consider the hypercube in $\mathbb{R}^m$ given by the minimal and maximal coefficients of projections of the input data onto the SPGA subspace. Then we draw our parametrization coefficient samples $S \subset \mathbb{R}^m$ uniformly from this hypercube. To create the samples S^l for the factor manifolds, we simply take the corresponding subcomponents of coefficient vectors from S. The corresponding shapes were then computed by evaluating the exponential map for each of them. Overall, we sampled approximately $|S| = 4000$ points for each of the considered examples. The dataset was split randomly into a training (80 %) and a test (20 %) set, with the training set being used for the descent method of the loss functionals and the test set being used to evaluate the performance of the networks.

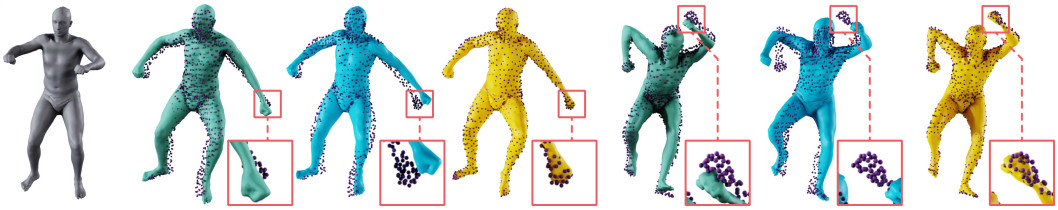

Figure 4: Two examples from the SCAPE dataset. We use the same colors as in Figure 1.

Example	m	L	m_l	Affine	Monolithic	Composite (Ours)	Sassen et al. (2020b)
SMPL+X	80	10	3 – 24	0.78	0.85	0.93	—
SCAPE	40	6	5 – 9	0.77	0.60	0.91	—
Hands	12	4	2 – 4	0.88	0.95	0.98	0.80
Faces	10	6	1 – 4	0.96	0.95	0.99	0.95

Table 1: Approximation quality R^2 on SPGA examples.

Comparison to Affine Approximation Sassen et al. (2020b) also proposed a scheme based on multilinear interpolation of precomputed exponentials for each of the factor manifolds and subsequent affine combination of the results to approximate the exponential map and parametrize $\mathcal{M}$. A natural question is how our network-based approach compares to this.

For the humanoid examples, it was not possible to precompute Riemannian exponentials on a regular grid for all factor manifolds due to their high dimensionality. Hence, instead of multilinearly interpolating precomputed exponentials we simply compute the exponentials within each factor manifold exactly before combining them affinely. We dub this method simply 'affine combination' and present its results in Figure 4 and Table 1; it is computationally heavy, but yields an upper bound on the quality of the method by Sassen et al. (2020b). The limitation does not apply to our new approach, which for example allows us to learn an efficient parametrization for the SMPL+X dataset, where the expressive movements of hand and face require a higher-dimensional data manifold.

In all examples, our composite network approach achieves higher approximation accuracy than the 'affine combination'. This shows that the network Ψ^θ is able to correct the approximation errors of the direct sum structure. For the lower-dimensional examples, this difference is not as pronounced since their sparse modes have a better support separation.

Furthermore, storing our network-based approximation requires less memory than the approach by Sassen et al. (2020b). For example, on the SCAPE dataset storing grids with approx. 20000 samples as reported by Sassen et al. (2020b) requires about 1.7 GB of storage, while our networks only require 0.6 GB (without optimizing for a small memory footprint).

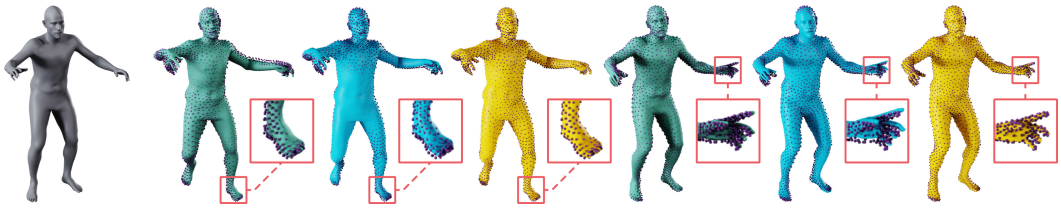

Figure 5: Two examples from the SMPL-X dataset. We use the same colors as in Figure 1.

Comparison to Monolithic Network Another obvious question is whether our composite approach shows any benefit over training a simple, single network. For evaluation we also trained one fully connected network $\widetilde{\Phi}^\theta \colon \mathbb{R}^m \to \mathcal{M}$ to approximate the parametrization at once, dubbed 'monolithic' approach. The corresponding approximation qualities are reported in Table 1. One sees that for the lower-dimensional examples this monolithic approach achieves an approximation quality close to the one of our composite network. However, for the higher-dimensional, humanoid examples the approximation quality of the monolithic approach is noticeably lower.

6 Conclusion

Our results suggest that the most fundamental geometric operation on Riemannian data manifolds, the parametrization of the manifold via the Riemannian exponential map, can in principle be learned. We illustrated this on shape manifolds of triangular meshes, for which the exponential map is computationally expensive so that an approximation is attractive. However, while naive implementations via deep neural networks proved ineffective, we achieved consistently satisfying results by matching our training and network architecture to a typical structure of shape manifolds: that they can be approximated by an affine sum of submanifolds. We thus learned both, the lower-dimensional Riemannian exponential map on each submanifold as well as the (close to affine) composition of the different submanifolds. We furthermore illustrated in our examples that such manifold structures arise from basic principles like support or scale separation of different shape variations.

While we implemented the above concept of composite networks only for shape manifolds, it should also be applicable to image manifolds. However, first the corresponding tools to identify approximate product manifold structures (such as the SPGA) would have to be developed for images. It is also conceivable to replace the SPGA by learning approaches akin to disentanglement learning. Furthermore, we did not touch upon further optimization for the specific setting of our shape manifolds: The single factor manifolds typically describe localized shape variations so that a sparsity regularization of the corresponding networks would make sense and could substantially reduce the parameter size. Furthermore, one could add a regularization favoring those NRIC that correspond to an immersed mesh, thereby reducing postprocessing.

REFERENCES

Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. SCAPE: shape completion and animation of people. *ACM Trans. Graph.*, 24(3):408–416, 2005.

Karthik Bharath, Sebastian Kurtek, Arvind Rao, and Veerabhadran Baladandayuthapani. Radiologic image-based statistical shape analysis of brain tumours. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 67(5):1357–1378, 2018.

Mario Botsch, Stephan Steinberg, Stephan Bischoff, and Leif Kobbelt. Openmesh - a generic and efficient polygon mesh data structure. *1st OpenSG Symposium*, 2002.

Christopher Brandt, Christoph von Tycowicz, and Klaus Hildebrandt. Geometric flows of curves in shape space for processing motion of deformable objects. *Computer Graphics Forum*, 35(2), 2016.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations*, 2016.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

P. Thomas Fletcher, Conglin Lu, Stephen M. Pizer, and Sarang Joshi. Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE Trans. Med. Imaging*, 23(8):995–1005, 2004.

Stefan Fröhlich and Mario Botsch. Example-driven deformations based on discrete shells. *Comput. Graph. Forum*, 30(8):2246–2257, 2011.

Marco Fumero, Luca Cosmo, Simone Melzi, and Emanuele Rodolà. Learning disentangled representations via product manifold projection. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 3530–3540, 2021.

Gaël Guennebaud, Benoît Jacob, and Others. Eigen v3. http://eigen.tuxfamily.org, 2010.

Behrend Heeren, Martin Rumpf, Max Wardetzky, and Benedikt Wirth. Time-discrete geodesics in the space of shells. *Comput. Graph. Forum*, 31(5):1755–1764, 2012.

Behrend Heeren, Martin Rumpf, Peter Schröder, Max Wardetzky, and Benedikt Wirth. Exploring the geometry of the space of shells. *Comput. Graph. Forum*, 33(5):247–256, 2014.

Behrend Heeren, Chao Zhang, Martin Rumpf, and William Smith. Principal geodesic analysis in the space of discrete shells. *Comput. Graph. Forum*, 37(5), 2018.

Zhichao Huang, Junfeng Yao, Zichun Zhong, Yang Liu, and Xiaohu Guo. Sparse localized decomposition of deformation gradients. *Comput. Graph. Forum*, 33(7):239–248, 2014.

Martin Kilian, Niloy J. Mitra, and Helmut Pottmann. Geometric modeling in shape space. *ACM Trans. Graph.*, 26(64):1–8, 2007.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Sebastian Kurtek, Eric Klassen, Zhaohua Ding, Sandra Jacobson, Joseph B. Jacobson, Malcolm Avison, and Anuj Srivastava. Parameterization-invariant shape comparisons of anatomical surfaces. *IEEE Trans. Medical Imaging*, 30(3):849–858, 2011a.

Sebastian Kurtek, Eric Klassen, John C. Gore, Zhaohua Ding, and Anuj Srivastava. Classification of mathematics deficiency using shape and scale analysis of 3d brain structures. In Benoit M. Dawant and David R. Haynor (eds.), *Medical Imaging 2011: Image Processing*, volume 7962 of *SPIE Proceedings*, pp. 796244. SPIE, 2011b.

Sebastian Kurtek, Qian Xie, Chafik Samir, and Michel Canis. Statistical model for simulation of deformable elastic endometrial tissue shapes. *Neurocomputing*, 173:36–41, 2016. ISSN 0925-2312.

Hamid Laga, Sebastian Kurtek, Anuj Srivastava, and Stanley Miklavcic. Landmark-free statistical analysis of the shape of plant leaves. *Journal of theoretical biology*, 363:41–52, 08 2014.

Michael I. Miller, Alain Trouvé, and Laurent Younes. Geodesic shooting for computational anatomy. *J. Math. Imaging Vis.*, 24(2):209–228, 2006.

Thomas Neumann, Kiran Varanasi, Stephan Wenger, Markus Wacker, Marcus Magnor, and Christian Theobalt. Sparse localized deformation components. *ACM Trans. Graph.*, 32(6):1–10, 2013.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary 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 H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, 2019.

Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 10975–10985, 2019.

Xavier Pennec. Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *J. Math. Imaging Vis.*, 25(1):127–154, 2006.

Xavier Pennec. *Statistical Computing on Manifolds: From Riemannian Geometry to Computational Anatomy*, pp. 347–386. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

David Pfau, Irina Higgins, Aleksandar Botev, and Sébastien Racanière. Disentangling by subspace diffusion. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Martin Rumpf and Benedikt Wirth. Variational time discretization of geodesic calculus. *IMA J. Numer. Anal.*, 35(3):1011–1046, 2015.

Chafik Samir, Sebastian Kurtek, Anuj Srivastava, and Michel Canis. Elastic shape analysis of cylindrical surfaces for 3d/2d registration in endometrial tissue characterization. *IEEE Trans. Medical Imaging*, 33(5):1035–1043, 2014.

Josua Sassen, Behrend Heeren, Klaus Hildebrandt, and Martin Rumpf. Geometric optimization using nonlinear rotation-invariant coordinates. *Computer Aided Geometric Design*, 77, 2020a.

Josua Sassen, Klaus Hildebrandt, and Martin Rumpf. Nonlinear Deformation Synthesis via Sparse Principal Geodesic Analysis. *Comput. Graph. Forum*, 39(5), 2020b. ISSN 1467-8659.

Philipp von Radziewsky, Elmar Eisemann, Hans-Peter Seidel, and Klaus Hildebrandt. Optimized subspaces for deformation-based shape editing and interpolation. *Computers & Graphics*, 58: 128–138, 2016.

Guan Wang, Hamid Laga, Ning Xie, Jinyuan Jia, and Hedi Tabia. The shape space of 3d botanical tree models. *ACM Trans. Graph.*, 37(1):7, 2018.

Yuanzhen Wang, Beibei Liu, and Yiying Tong. Linear surface reconstruction from discrete fundamental forms on triangle meshes. *Comput. Graph. Forum*, 31(8):2277–2287, 2012.

Yupan Wang, Guiqing Li, Zhichao Zeng, and Huayun He. Articulated-motion-aware sparse localized decomposition. *Comput. Graph. Forum*, 36(8):247–259, 2017.

Yupan Wang, Guiqing Li, Huiqian Zhang, Xinyi Zou, Yuxin Liu, and Yongwei Nie. Panoman: Sparse localized components–based model for full human motions. *ACM Transactions on Graphics (TOG)*, 40(2):1–17, 2021.

Qian Xie, Ian H. Jermyn, Sebastian Kurtek, and Anuj Srivastava. Numerical inversion of SRNFs for efficient elastic shape analysis of star-shaped objects. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars (eds.), *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pp. 485–499. Springer, 2014.

I-Cheng Yeh, Chao-Hung Lin, Olga Sorkine, and Tong-Yee Lee. Template-based 3d model fitting using dual-domain relaxation. *IEEE Trans. Vis. Comput. Graph.*, 17(8):1178–1190, 2011.

Laurent Younes. *Shapes and Diffeomorphisms*. Springer, 2010.

Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. Spacetime faces: High resolution capture for modeling and animation. *ACM Trans. Graph.*, 23(3):548–558, 2004. ISSN 0730-0301.

Sharon Zhang, Amit Moscovich, and Amit Singer. Product manifold learning. In Arindam Banerjee and Kenji Fukumizu (eds.), *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pp. 3241–3249. PMLR, 2021.

Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006.

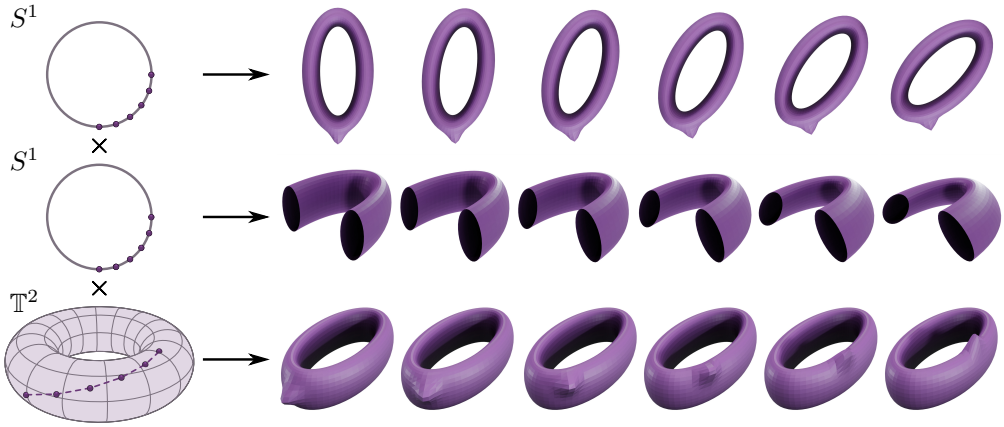

Figure 6: Factors of the *Freaky Torus*. We visualize our synthetic shape space by demonstrating the effect of moving along the individual factors to the final shape. In the first row, we see how the first factor, an S^1, controls the deformation of the latitudinal cross-section. We show the deformed torus from the top. In the second row, we see how the next factor, another S^1, controls the deformation of the longitudinal cross-section. Here, we show the torus cut in half to better highlight the cross-section's shape. In the last row, we show how the third factor, a two-dimensional flat torus $\mathbb{T}^2$, controls the position of a bump on the deformed torus.

A Freaky Torus

In this appendix, we provide more details on the construction of our shape space *Freaky Torus* of deformed tori, a synthetic shape space with factors $S^1 \times S^1 \times \mathbb{T}^2$, whose action is also summarized in Figure 6. We will first derive its continuous version leading to a family of parametrization and then arrive at the discrete version via a simple spatial discretization. To begin, we recall the parametrization $f\colon [0, 2\pi)^2 \to \mathbb{R}^3$ of a standard torus of revolution with radii R and r of the latitudinal and longitudinal circular cross-sections respectively, which is given by

$$f(u, v) \coloneqq R \begin{pmatrix} \cos(u) \\ \sin(u) \\ 0 \end{pmatrix} + r \begin{pmatrix} \cos(u)\,\cos(v) \\ \sin(u)\,\cos(v) \\ \sin(v) \end{pmatrix}. \tag{6}$$

The first two factors $S^1 \times S^1$ of our shape space control the deformation of these cross-sections into ellipses and hence we want to replace the parametrizations of the cross-sectional circles by ones of appropriate ellipses. To this end, a circle deformed into an ellipse with semi-axes' lengths a and b rotated by $\eta/2$ against the coordinate axis is parametrized by

$$\tau^{\eta,a,b}(t) \coloneqq \begin{pmatrix} a\cos(\frac{\eta}{2})\cos(t - \frac{\eta}{2}) - b\sin(\frac{\eta}{2})\,\sin(t - \frac{\eta}{2}) \\ a\sin(\frac{\eta}{2})\,\cos(t - \frac{\eta}{2}) + b\cos(\frac{\eta}{2})\sin(t - \frac{\eta}{2}) \end{pmatrix}. \tag{7}$$

The phase-shift in the parametrization comes from the fact that the rotation of the semi-axes is also achieved by a warping of the circle instead of rotating it. We introduced this parametrization with the half-scaling of η for cosmetic reasons so that later all deformations are parametrized over the same interval $[0, 2\pi)$.

Now, we use this parametrization as replacement for the circular cross-section in the parametrization f of the torus. We fix the semi-axes' lengths a_R, b_R and a_r, b_r of the latitudinal and longitudinal cross-sections respectively and introduce their rotation as parameters $\alpha, \beta \in [0, 2\pi)$. This leads to the parametrization

$$f^{\alpha,\beta}(u, v) \coloneqq R \begin{pmatrix} \tau_1^{\alpha,a_R,b_R}(u) \\ \tau_2^{\alpha,a_R,b_R}(u) \\ 0 \end{pmatrix} + r \begin{pmatrix} \cos(u)\,\tau_1^{\beta,a_r,b_r}(v) \\ \sin(u)\,\tau_1^{\beta,a_r,b_r}(v) \\ \tau_2^{\beta,a_r,b_r}(v) \end{pmatrix}. \tag{8}$$

The last factor $\mathbb{T}^2$ controls the position on a bump on such a torus. To describe the corresponding deformation, we first need the normal of the surface in which direction the bump will point. It is

given by the usual formula for smooth surfaces namely

$$n^{\alpha,\beta}(u,v) \coloneqq \frac{\partial_u f^{\alpha,\beta} \times \partial_v f^{\alpha,\beta}}{\|\partial_u f^{\alpha,\beta} \times \partial_v f^{\alpha,\beta}\|}(u,v). \tag{9}$$

Then the bump is a deformation in the direction of this normal around the position determined by $(\gamma,\zeta) \in [0,2\pi)^2$ with maximal height h. To limit the support of the deformation, we use a simple Gaussian with parameter ε on the distance to the center point.

We finally arrive at the parametrization of the continuous version of our synthetic shape space

$$\begin{aligned} \mathcal{F}\colon [0,2\pi) \times [0,2\pi) \times [0,2\pi)^2 &\to \left([0,2\pi)^2 \to \mathbb{R}^3\right) \\ (\alpha,\beta,\gamma,\zeta) &\mapsto \left((u,v) \mapsto f^{\alpha,\beta}(u,v) + h\, e^{-\frac{\|f^{\alpha,\beta}(u,v) - f^{\alpha,\beta}(\gamma,\zeta)\|^2}{\varepsilon^2}} n^{\alpha,\beta}(\gamma,\zeta)\right), \end{aligned} \tag{10}$$

where the images are parametrizations of embedded surfaces. To obtain discrete surfaces, we apply these to a fixed triangulation of the torus, which yields the discrete version of the shape space.

For the dataset we used in our experiments, we chose $a_R = a_r = 1, b_R = b_r = \frac{1}{2}, R = 0.375, r = 0.125, h = 0.075$, and $\varepsilon = 0.05$. It is available along with the code at [to be added in final publication].