Learning Geometry of Pose Image Manifolds in Latent Spaces Using Geometry-Preserving GANs

Shenyuan Liang[1], Benjamin Beaudett[1], Pavan Turaga[2], Saket Anand[3], and Anuj Srivastava[1(✉)]

[1] Florida State University, Tallahassee, FL 32309, USA
{sl20fu,bbeaudett,asrivastava}@fsu.edu
[2] Arizona State University, Tempe, AZ 85281, USA
pturaga@asu.edu
[3] IIIT-Delhi, Delhi, India
anands@iiitd.ac.in

Abstract. The goal of this paper is to learn the differential geometry of pose image manifolds for 3D objects. Indexed by the rotation group $SO(3)$, a pose manifold constitutes images of a 3D object from all viewing angles. Learning geometry implies computing geodesics, intrinsic statistics (means, etc), and curvatures on estimated manifolds. As these goals are unattainable in the huge image space, we perform dimension reduction that is **geometry preserving** and **invertible**. This paper introduces two distinct concepts: (1) A **Geometry-Preserving StyleGAN** (GP-StyleGAN2) that maps training images to a low-dimensional latent space with two novel geometry-preserving terms. These terms penalize changes in pairwise distances between points and pairwise angles between tangent spaces under the map. (2) Densifying the estimated manifold in latent space using **Euler's Elasticae**-based nonlinear interpolations between sparse data points. In contrast to the past findings, the latent pose manifolds are found to be distinctly nonlinear and similar in shape across objects. Incorporating these features results in superior performance in image interpolation, denoising, and computing image summaries when compared to state-of-the-art GANs and VAEs.

Keywords: Manifold Learning · Pose Image Manifold · Elasticae · Latent Space Geometry · Geodesics · Geometric GAN

1 Introduction

Image manifolds are subsets of image spaces corresponding to images of 3D objects of interest. In this paper, we focus on specific image manifolds called *rotation or pose manifolds*. A pose manifold is the set of images of an object under all 3D rotations (while fixing other imaging conditions). Even though images are high-dimensional, the

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-78398-2_4.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15327, pp. 56–72, 2025.
https://doi.org/10.1007/978-3-031-78398-2_4

pose manifolds are typically low-dimensional and are nonlinearly embedded in the huge ambient Euclidean space. Learning the differential geometries of image manifolds has been a long-standing and challenging goal in the field, especially when using limited training data. Learning, characterizing, and exploiting this geometry can help accomplish several goals in image analysis and computer vision: interpolating between images using geodesics, denoising an image using manifold projection, making statistical analysis more interpretable by adhering to the manifold structure, and creating simple yet powerful generative and discriminative models by defining probability distributions on the manifold. The goal of this paper is to *learn the differential geometry of image manifolds for individual 3D objects*, enabling computation of geodesics, tangents, curvatures, and intrinsic statistics (means, etc.) on the estimated manifolds.

We will follow the notation in [13] to develop a mathematical formulation. Let α be a 3D object, such as a chair, airplane, or car, and let O^α denote its 3D geometry and reflectance model. Let $s \in SO(3)$ represent the 3D pose of O^α relative to the camera and $\mathbb{P}$ be the orthographic projection of sO^α into the focal plane of the camera, resulting in an $n \times n$ image $\mathbb{P}(sO^\alpha)$. As mentioned earlier, all other imaging variables, *e.g.* illumination, are fixed for this discussion.

Definition 1 *Under these conditions, the set* $\mathcal{I}^\alpha = \{\mathbb{P}(sO^\alpha) \in \mathbb{R}^{n\times n} | s \in SO(3)\}$*, a subset of* $\mathbb{R}^{n\times n}$*, is called the rotation or pose manifold of* α*.*

Using additional assumptions on the smoothness of $\mathbb{P}$ and non-symmetry of O^α with respect to the rotation group, the set $\mathcal{I}^\alpha$ shares the closed, boundary-free manifold topology of $SO(3)$. (This statement deserves additional consideration to be precise but we leave the details for a future paper.) It is thus three-dimensional and its nonlinear embedding is small (almost singular) in the ambient space $\mathbb{R}^{n\times n}$.

Problem Specification: Given a training set of rotation-image pairs $\mathcal{R}_{train} = \{(s_i, I_i = \mathbb{P}(s_i O^\alpha)) \in SO(3) \times \mathcal{I}^\alpha\}_{i=1}^m$, our goal is to estimate the manifold $\mathcal{I}^\alpha$. We want not only to learn the topological set $\mathcal{I}^\alpha$, but also to characterize its local and global geometry. This characterization will enable us to compute quantities on the manifold such as geodesics, geodesic distances, and intrinsic statistical summaries. The task of learning $\mathcal{I}^\alpha$ from $\mathcal{R}_{train}$ is challenging because (i) Learning nonlinear manifolds typically requires large sample sizes that are even greater in high-dimensional (n^2) spaces, and (ii) The underlying geometry of $\mathcal{I}^\alpha$ is often complex and does not follow known simplifications such as spheres, ellipsoids, or hyperbolic spaces.

The difficulties associated with high dimensions can be mitigated by mapping to a smaller Euclidean space $\mathbb{R}^d, d \ll n^2$. If we have a map $\Phi : \mathbb{R}^{n^2} \to \mathbb{R}^d$ which is approximately isometric and invertible on the manifold, we can analyze the geometry on the simpler latent manifold $\mathcal{M}^\alpha = \Phi(\mathcal{I}^\alpha) \subset \mathbb{R}^d$ using $\{(s_i, \Phi(I_i))\}_{i=1}^m$ and map the results back to the image manifold using Φ^{-1}. The next question is: What is a good choice of Φ? Existing methods for manifold learning (such as LLE, Isomap, t-SNE, etc.), dimension reduction and latent space representation either distort the geometry of $\mathcal{I}^\alpha$ or are not invertible (see section 2). We require a new technique.

Our Approach: There are two main elements to our approach. Firstly, we seek a mapping Φ that preserves the geometry of the pose manifold. Secondly, rather than assuming

a flat geometry in the latent space as is often the case in the current literature, we use nonlinear interpolations between mapped points in $\mathbb{R}^d$ to discover the unknown manifold. We introduce these items here and elaborate on the details in section 3.

(1) Geometry-Preserving and Invertible Dimension Reduction: How can one design a Φ that maximally preserves the geometry? Preserving the geometry implies that distances, angles, and curvatures remain similar from the domain to the range of Φ. We create Φ by training with a neural network that combines elements of GANs and autoencoders, specifically a modified version of AE-StyleGAN2. The decoder trained as part of this model provides an inverse map as well. We build on this architecture by introducing two geometry-preserving loss terms, and call our model *Geometry-Preserving* StyleGAN2 or GP-StyleGAN2. The new terms help preserve (i) pairwise Euclidean distances between point locations, and (ii) pairwise dissimilarity (a measure of local curvature) between tangent space orientations. Here we use all training points to compute pairwise distances, and not just the neighbors, and treat Euclidean distances as *extrinsic* distances to help learn the global geometry. Estimation of tangent planes $\{T_i\}$ at points $\{I_i\}$ used to compute orientation dissimilarity is described later.

(2) Discovering the Manifold using Nonlinear Elasticae: We use the trained Φ to map points from another sparse set $\mathcal{R}_{test} \subset \mathbb{R}^{n^2}$ (disjoint from $\mathcal{R}_{train}$) to the latent space, resulting in $\{\Phi(I_i) \in \mathbb{R}^d\}$. These points lie on the latent manifold $\mathcal{M}^\alpha$ and we can use them to uncover it in more detail. We do this by interpolating between neighboring mapped points using *free elasticae* [25,29]. Elasticae use curved interpolations between pairs $(\Phi(I_i), \Phi(I_j))$, with curvatures dictated by their distance and the misalignment of tangent planes $(d\Phi(T_i), d\Phi(T_j))$. Repeatedly applying this tool between neighboring points, we 'fill in' the manifold with arbitrarily dense point sets and produce an estimated latent manifold $\widehat{\mathcal{M}}^\alpha$ and the corresponding image manifold $\widehat{\mathcal{I}}^\alpha = \Phi^{-1}(\widehat{\mathcal{M}}^\alpha)$.

Knowing the geometry of $\mathcal{M}^\alpha$ allows us to improve performance in some crucial vision tasks including: (1) Image Interpolation Using Geodesics: Given any two images $I_i \equiv \mathbb{P}(s_i O^\alpha)$ and $I_j \equiv \mathbb{P}(s_j O^\alpha)$ in the test data, the task is to estimate the image path $t \mapsto \mathbb{P}(x(t)O^\alpha)$, where $x : [0,1] \to SO(3)$ is a geodesic between s_i, s_j in $SO(3)$. The image path should resemble the video of a rotating object. (2) Intrinsic Image Statistics and Modeling: Given a set of images $\{I_i\}$, one would like to compute their summary statistics (mean, covariance) and develop statistical models as elements of $\mathcal{I}^\alpha$ rather than $\mathbb{R}^{n^2}$. (3) Image Denoising Using Manifold Projection: Given a noisy or corrupted image $J \in \mathbb{R}^{n^2}$ known to be associated with an $I \in \mathcal{I}^\alpha$, we seek a tool to denoise it.

2 Related Works

In recent years, deep neural networks (DNNs) have provided powerful tools for encoding of images by mapping them to low-dimensional latent spaces. In the following we summarize some past ideas that are most relevant to our method.

Manifold Learning Techniques: From the pre-deep network era, there is a long list of learning methods that sought nonlinear dimension-reduction while preserving some

geometric properties for image data, including LLE [33], Isomap [40], LTSA [44], Laplacian eigenmaps [3], Hessian eigenmaps [10], diffusion maps [7], vector diffusion maps [37], Riemannian relaxation [27], and t-SNE [26]. These were successful in mapping image data into smaller Euclidean spaces while preserving pairwise distances or other local properties. However, these mostly only go as far as embedding the observed training points in a low-dimensional space. They fall far short of our goal of creating an invertible map that can read out-of-sample points in both the input and latent spaces. Several recent papers such as FM-VAE [6], IRVAE [42], GRAE [11], structure-preserving AE [38], GGAE [24], and DIMAL [31] seek geometry-aware manifold learning using different DNN architectures. Our work constitutes further exploration of this area.

Differential Geometry of Latent Spaces: Bengio et al. [4] stressed the importance of understanding the geometry of latent space representations. Several papers [23,35,36] have investigated this geometry, mainly utilizing existing architectures geared towards image synthesis, and reported them to be (surprisingly) flat. Other papers [2,9,16] demonstrated that Jacobian-based Riemannian metrics on the latent space produce better inference results than using Euclidean distance. Sáez et al. [30] fitted local, constant-curvature patches to data using Gromov-Hausdorff distance and Bayesian optimization. Zhang and Jiang [43] presented a method for geometric space selection in representation learning, allowing data points to select optimal geometric spaces.

GANs and VAEs: Goodfellow et al. [12] introduced the basic framework and training procedure for generative adversarial networks (GANs). Radford et al. [32] improved their stability and efficiency, while Karras et al. [18] proposed Progressive Growing GAN and Style-Based GAN that incorporated regularizations. Han et al. [14] designed AE-StyleGAN2 for more disentangled latent space and improved efficiency. Kingma and Welling [22] introduced variational autoencoder (VAE) to map input data to a low-dimensional latent space (encoder) and back (decoder). VAEs have been extended in various directions. Davidson et al. [8] proposed the *Hyperspherical VAE* (SVAE) that samples latent vectors on a unit sphere. Chadebec and Allassonière [5] proposed the geometry-based *Riemannian Hamiltonian VAE* (RHVAE), which models the latent space as a Riemannian manifold, combining Riemannian metric learning and geodesic shooting. Huh et al. [17] proposed Quotient VAEs.

3 Proposed Framework

In this section, we present the design of *Geometry Preserving StyleGAN2* (GP-StyleGAN2), which facilitates learning by preserving the geometry of the image manifold. We start with a brief introduction to StyleGANs and AE-StyleGAN2. Consider a set of images $\{I_i \in \mathcal{I}\}_{i=1}^{b}$ and random latent vectors $\{v_k \in \mathcal{V} \equiv \mathbb{R}^d\}_{k=1}^{K}$ sampled from a probability distribution P_v. StyleGANs [20] use a *Multilayer perceptron* (MLP) $F : \mathcal{V} \rightarrow \mathcal{W}$ that maps v_k to an intermediate latent space point w_k which is then fed to a generator $G : \mathcal{W} \rightarrow \mathcal{I}$ that synthesizes an image $G(F(v_k))$. One trains the generator by pitting it against a discriminator network Q that distinguishes

between real and generated images. The disentangled latent space $\mathcal{W}$ used in StyleGANs gives improved image generation compared to the basic GAN architecture. AE-StyleGAN2 [14] borrows ideas from a VAE, attaching an encoder E to the model which maps from the image to latent space (as our application requires) and giving additional training to G as a decoder. E and G are trained using autoencoder reconstruction loss $\min_{E,G} \left\|I - G(E(I))\right\| + \left\|\eta(I) - \eta(G(E(I)))\right\|$ where η is a pre-trained VGG16 network, and adversarial loss $\min_{E,F,G} \max_Q \mathbb{E}_I[\log Q(I)] + \mathbb{E}_v[\log(1 - Q(G(F(v))))] + \mathbb{E}_I[\log(1 - Q(G(E(I))))]$ which adds an autoencoder term to previous StyleGAN objectives. While AE-StyleGAN2 accomplishes its aims, it does not consider the geometry of the image manifold. *For preserving geometry, we propose GP-StyleGAN2.*

3.1 Learning Latent Map Using GP-StyleGAN2

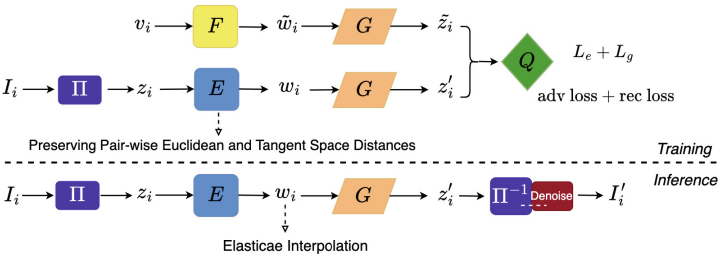

Fig. 1. Training and inference procedure. We preserve the geometry of image space by constraining the encoder (E) with loss functions L_e and L_g defined in Algorithms 1, 2.

GP-StyleGAN2 Architecture: We seek a **significant** dimension reduction (from $n^2 = 2^{14}$ to $d = 5$) which is (1) **invertible** and (2) **geometry preserving** ($d = 5$ is the smallest embedding dimension of $SO(3)$). The computational demands of learning a nonlinear map for such a drastic reduction suggest a two-step approach. We first move the problem to points $\{z_i\}$ in an intermediate dimension $c^2 = 2^{10}$ using PCA. This linear projection Φ suffices for a modest reduction while being approximately invertible and norm-preserving (Parseval's theorem), though it will fail if we use it for the full reduction $2^{14} \to 5$. From here, we train the nonlinear encoder $E : \mathbb{R}^{c^2} \to \mathbb{R}^d$ generator $G : \mathbb{R}^d \to \mathbb{R}^{c^2}$ between $\{z_i\}$ in the PCA space and points $\{w_i\}$ in the latent space. After this training we build the finalized maps $\Phi : \mathbb{R}^{n^2} \to \mathbb{R}^d$ and $\Phi^{-1} : \mathbb{R}^d \to \mathbb{R}^{n^2}$ defined by the compositions $\Phi = E \circ \Phi$ and $\Phi^{-1} = \Phi^{-1} \circ G$. We note that Φ^{-1} and Φ^{-1} are approximate inverses and that the PCA reconstruction Φ^{-1} is fine-tuned by a denoising neural network. The architecture for training E and G begins with the AE-StyleGAN2 autoencoder and adversarial objectives, but augments the training of E with geometry-preserving loss terms based on pairwise point and tangent space distances. Fig. 1 shows a schematic of the training and inference procedure.

Algorithm 1. Optimizing encoder E with geometry-preserving term L_e (points)

1: Given (1): E; (2): A batch of images $I^B \in \mathbb{R}^{b \times n^2}$.
2: Map images to PCA space: $Z^B = \Phi(I^B) \in \mathbb{R}^{b \times c^2}$.
3: Map Z^B to latent space: $W^B = E(Z^B) \in \mathbb{R}^{b \times d}$.
4: Compute pair-wise distance matrices: $D_{ij}^{e,z} = \|z_i - z_j\|$ and $D_{ij}^{e,w} = \|w_i - w_j\|$, where $z_i, z_j \in Z^B$, $w_i, w_j \in W^B$, and $\|\cdot\|$ denotes the Euclidean norm.
5: Compute the loss: $L_e = L(D^{e,z}, D^{e,w})$, where L is defined in Eqn. 1.
6: Update the weights: $\theta_E \leftarrow \text{ADAM}(\nabla_{\theta_E} L_e, \theta_E)$, where θ_E denotes the internal parameters of E, and ADAM refers to adaptive moment optimization [21].

Geometry-preserving Terms: The central feature of our method is a loss function designed to make the Euclidean distances between the latent space encodings of images correspond to their Euclidean distances in the image space. The loss is computed as a dissimilarity between pairs of distance matrices. We use three types of metrics to compute these matrices. The first is the standard Euclidean distance $\|\cdot\|$. The second is a metric defined on $SO(3)$: Using the rotation matrix representations $s_i, s_j \in SO(3)$ of two poses of an object, the Riemannian distance between them is $d_s(s_i, s_j) = \cos^{-1}\left(\frac{\text{trace}(s_i s_j^T)-1}{2}\right)$. The third is a metric on sets of linear subspaces with common dimensions: Let $T_i, T_j \in \mathbb{R}^{d \times r}$, $r \leq d$, denote arbitrary orthogonal bases of any two r-dimensional subspaces in $\mathbb{R}^d$. Then, define $d_g(T_i, T_j) = \|T_i T_i^T - T_j T_j^T\|_F$. We use this extrinsic distance on the Grassmannian manifold to simplify computations.

In practice, we approximate tangent spaces in the PCA space and the latent space using the training data as follows. First, we find $N > 3$, $SO(3)$-neighbors for each training image (s_i) using the lowest values of $d_s(s_i, s')$. For PCA points, we then approximate N tangent vectors at z_i using finite differences as $\{V_{ij} = \frac{z_j - z_i}{d_s(s_i, s_j)} \in \mathbb{R}^{c^2}\}_{j=1}^N$ and set $T_i^z \in \mathbb{R}^{c^2 \times 3}$ to be the three dominant singular vectors of the set $\{V_{i\cdot}\}$. Similarly, we can approximate tangent spaces $T_i^w \in \mathbb{R}^{d \times 3}$ in the latent space (see Algorithm 2). A very similar method for tangent plane estimation is used in [37], where a proof is given for convergence to the true tangent plane.

Given a batch of b images, we compute distance matrices $D \in \mathbb{R}^{b \times b}$ between mapped points and tangent planes in the PCA and latent spaces. Then a measure of discrepancy between computed matrices D^1 (PCA space) and D^2 (latent space) is calculated using:

$$L(D^1, D^2) = \sum_{j=1}^{b} \left[1 - \frac{(D_{\cdot j}^1 - \mu_j^1 \mathbf{1})^T (D_{\cdot j}^2 - \mu_j^2 \mathbf{1})}{\|D_{\cdot j}^1 - \mu_j^1 \mathbf{1}\| \, \|D_{\cdot j}^2 - \mu_j^2 \mathbf{1}\|}\right], \tag{1}$$

where μ_j^1 and μ_j^2 are the mean values for columns $D_{\cdot j}^1$ and $D_{\cdot j}^2$, and $\mathbf{1}$ is a vector of ones. We center and scale columns (or rows) of D matrices into unit vectors and compute the cosines of angles between them. In an implicit manner, each entry of D^1 is compared with the corresponding entry of D^2. Since we are forming a loss function, we subtract this quantity from one, and sum over all points in the batch. We use the resulting loss L to define novel geometric terms for modifying AE-StyleGAN2 as follows:

1. **Term 1: Distance Preserving**: Here $D^1_{ij} = \|z_i - z_j\|$, the Euclidean distances between PCA scores of training images, and $D^2_{ij} = \|w_i - w_j\|$, the Euclidean distances between corresponding latent vectors. We use Euclidean distances between all pairs, not just the neighbors. These distances play the role of extrinsic or embedding distances between points on the (unknown) manifold and help learn its global geometry. Later on in the paper, once the manifold is estimated, we use geodesics and geodesic (intrinsic) distances to perform statistical analysis. In other words, we use the extrinsic Euclidean distance for learning and intrinsic geodesic distance for analysis. We will call the loss $L = L_e$ in this case. Algorithm 1 lists the steps for computing L_e.
2. **Term 2: Curvature Preserving**: Here $D^1_{ij} = d_g(T^z_i, T^z_j)$, the tangent space distance, and $D^2_{ij} = d_g(T^w_i, T^w_j)$ the tangent distances in the latent space. We will call the loss $L = L_g$ in this case. Algorithm 2 lists the steps for computing L_g.

Algorithm 2. Optimizing encoder E with geometry-preserving term L_g (tangent distances)

1: Given (1): A set of images $I^B \in \mathbb{R}^{b\times n^2}$. (2): The corresponding rotation set $S^B \in \mathbb{R}^{b\times 3\times 3}$. (3): Corresponding neighborhoods $\mathcal{N}^I_i = [I_{\ell_1(i)}, \cdots, I_{\ell_N(i)}] \in \mathbb{R}^{n^2\times N}$, of the N closest points to $I_i \in I^B$ according to d_s. (4): The the index functions $\{\ell_k\}_{k=1}^N$, which take an argument i and return the index of the kth nearest neighbor of I_i.
2: Map images and their corresponding neighbors to PCA space:
$Z^B = \Phi(I^B) \in \mathbb{R}^{b\times c^2}, \mathcal{N}^z_i = \Phi(\mathcal{N}^I_i) = [z_{\ell_1(i)}, \cdots, z_{\ell_N(i)}]$, where $z_i \in Z^B$.
3: Map Z^B and $\{\mathcal{N}^z_i\}_{i=1}^b$ to latent space:
$W^B = E(Z^B) \in \mathbb{R}^{b\times d}, \mathcal{N}^w_i = E(\mathcal{N}^z_i) = [w_{\ell_1(i)}, \cdots, w_{\ell_N(i)}]$, where $w_i \in W^B$.
4: Compute neighborhood $SO(3)$ distances:
$\Delta^s_i = \left[d_s(s_i, s_{\ell_1(i)}) \cdots d_s(s_i, s_{\ell_N(i)})\right] \in \mathbb{R}^N$, where $s_i, s_j \in S^B$.
5: Compute over-dimensional tangent planes in *Principal Component Analysis* (PCA) space and latent space:
$\tilde{T}^z_i = \left(\mathcal{N}^z_i - z_i \mathbf{1}^T\right) \cdot \text{diag}\left(\Delta^s_i\right)^{-1} \in \mathbb{R}^{c^2\times N}$, $\tilde{T}^w_i = \left(\mathcal{N}^w_i - w_i \mathbf{1}^T\right) \cdot \text{diag}\left(\Delta^s_i\right)^{-1} \in \mathbb{R}^{d\times N}$.
6: Compute tangent planes T^z_i, T^w_i by taking the three dominant singular vectors of the corresponding $\tilde{T}^z_i, \tilde{T}^w_i$.
7: Compute pair-wise distance matrices: $D^{g,z}_{ij} = d_g(T^z_i, T^z_j)$ and $D^{g,w}_{ij} = d_g(T^w_i, T^w_j)$.
8: Compute the loss: $L_g = L(D^{g,z}, D^{g,w})$.
9: Update the weights: $\theta_E \leftarrow \text{ADAM}(\nabla_{\theta_E} L_g, \theta_E)$.

3.2 Elasticae Interpolation

We can use the trained map Φ to project test data into the latent space. However, this data may be sparse, and we wish to discover the projected manifold $\mathcal{M}^\alpha$ at a higher resolution than the test data permits. One way to find intermediate points is through interpolation. Straight line interpolations would be reasonable if we had just the points $\{w_i \in \mathbb{R}^d\}$. However, our access to tangent planes $\{(w_i, T^w_i) \in \mathbb{R}^d \times \mathbb{R}^{d\times 3}\}$ at each

point allows us to account for the nonlinearity of the underlying manifold by utilizing nonlinear interpolations based on elasticae.

Elasticae are smooth curves that can be used to interpolate between directed points, *i.e.* Euclidean points with attached tangent vectors. Consider the set $\mathcal{B}$ of smooth curves in $\mathbb{R}^d$ parameterized on $[0,1]$. For a curve $\beta \in \mathcal{B}$, let $\dot{\beta}$ and κ_β denote its velocity and scalar curvature functions, $Len[\beta]$ its length, and define its elastic energy $En[\beta] = \frac{1}{2}\int_0^1 \kappa_\beta^2(s)\, ds$. Then the *free elastica* from a given directed point (w_1, u_1) to another (w_2, u_2) is the minimizing curve $\hat{\beta} = \arg\min_{\beta\in\mathcal{B}}(En[\beta] + \lambda Len[\beta])$ such that $\beta(0) = w_1$, $\beta(1) = w_2$, $\dot{\beta}(0) = u_1$, and $\dot{\beta}(1) = u_2$. The tuning parameter $\lambda > 0$ balances the focus on length versus curvature. Mumford [29] advocated the use of free elasticae as the most likely solutions to fill in the missing or obscured curves in images, *e.g.*, in the famous Kanizsa triangles.

Our implementation follows Mio et al. ([28], Algorithm 4.2). We interpolate between point pairs w_i, w_j in the latent space $\mathbb{R}^d$. To find the corresponding tangent vectors, we take the difference vector $\tilde{u}_{ij} = w_j - w_i$, project it separately into each point's tangent space, and scale to form the unit vectors $u_i \in \text{span}(T_i^w)$ and $u_j \in \text{span}(T_j^w)$. These maximally-aligned vectors are used to direct a free elastica β that interpolates from (w_i, u_i) to (w_j, u_j). This interpolation is mapped to a path in image space as $\Phi^{-1}(\beta(t))$. The top portion of Fig. 3 in Section 4.1 illustrates the image space elasticae as sequences of images.

4 Experiments

Before detailing the design and results of our experiments, we begin by laying the groundwork of key features which are used throughout.

Experimental Data: Creating an image set to represent an object $\mathcal{I}^\alpha$ for training Φ and Φ^{-1} requires a structured sampling over rotations in $SO(3)$. We express rotations using a Hopf coordinate system similar to Yershova et al. [41]. Seeking a partition of $SO(3)$ with regular equivolumetric cells, we first generate (θ, ϕ) values on $\mathbb{S}^2$ using the Fibonacci grids of Swinbank and Purser [39] (also studied in Hardin et al. [15]). We attach a circle of ψ values to each pair, and the circle is uniformly segmented following a ratio shown in [41] to produce $SO(3)$ cells analogous to cubes. The experimental results presented here are restricted to a patch of $SO(3)$ to simplify computations. This patch is $\mathcal{P} \triangleq \{(\theta, \phi, \psi) \in [\frac{2}{5}\pi, \frac{3}{5}\pi] \times [\frac{4}{5}\pi, \frac{6}{5}\pi]^2\}$. To create an image set, we begin with a three-dimensional object O^α set at a default position. We use CAD objects $\alpha \in$ {chair, sports car, zebra} from clara.io [1] processed using meshio [34]. We apply 4000 rotations from $\mathcal{P}$ to the default orientation and use $\mathbb{P}$ to generate images $\mathbb{P}(s_i O^\alpha)$. We will call this training set $\mathcal{R}_{train} \triangleq \{(s_i, I_i) \in (SO(3) \times \mathcal{I}^\alpha)\}_{i=1}^{4000}$. Fig. 2 shows a representation of these points. We also create a separate data set $\mathcal{R}_{test}$ of 3696 indexed images for testing and evaluation. The points in $\mathcal{R}_{test}$ also lie in $\mathcal{P}$ but are defined by a uniform rectangular grid in (θ, ϕ, ψ) coordinates.

Evaluation Metrics: Our goals include performing several tasks that can be evaluated quantitatively. In our quantifiable experiments, we compare model outcomes against

ground truth values using the Euclidean norm and tabulate errors. In our most-used scenario, we compare a ground truth path in image space $\{I(t) \in \mathbb{R}^{n^2}\}_{t=0}^{T}$ with an estimated $\{\hat{I}(t) = \Phi^{-1}(\hat{w}(t)) \in \mathbb{R}^{n^2}\}_{t=0}^{T}$ decoded from a computation in latent space. Our evaluations use the Squared Errors (SE) $\|\hat{I}(t) - I(t)\|^2$ indexed by t. For elasticae evaluations, we also compare the velocities along the path to the ground truth using $\|(\hat{I}(t+1) - \hat{I}(t)) - (I(t+1) - I(t))\|^2$.

Model Comparisons: We compare our method with three recent deep-learning generative models described in Section 2: RHVAE [5], SVAE [8], and AE-StyleGAN2 [14].

Implementation details: Experiments are conducted on a Linux workstation with Nvidia RTX A6000 (48GB) GPU and Intel Core i9-13900K CPU @ 5.8GHz with 128GB RAM. The hyperparameters of generator G and MLP F are chosen to be identical to those in [19]. The hyperparameters of encoder E are the same as in [14]. The size of image space is $n^2 = 128^2$. The size of PCA space is $c^2 = 32^2$. The dimension of latent space is $d = 5$. The batch size for training is $b = 64$. The tangent spaces are built using $N = 16$ neighbors. The balance parameter for elasticae is $\lambda = 1$.

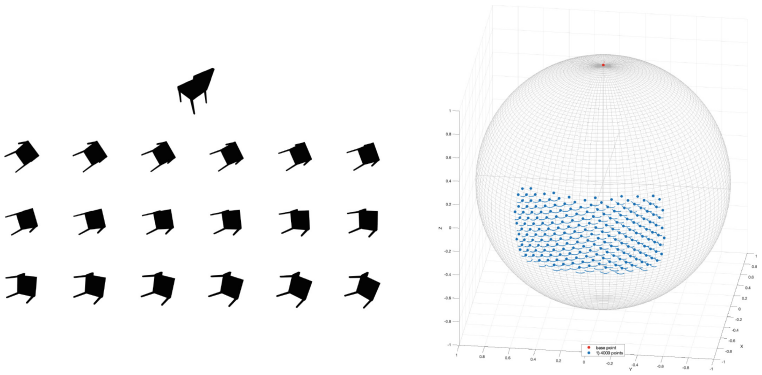

Fig. 2. Right: A visual representation of the 4000 $SO(3)$ training rotations. Each rotation is represented by its heading (θ, ϕ) on $\mathbb{S}^2$ and its roll about that heading drawn as a curve ψ around it. Left: Top image is the chair at default orientation. Others are $\mathbb{P}(sI^{chair})$ for some $s \in SO(3)$ with the same heading and varying roll.

4.1 Results: Elasticae Interpolations and Manifold Estimation

Having used $\mathcal{R}_{train}$ to train the maps Φ and Φ^{-1}, we map elements of $\mathcal{R}_{test}$ to points in latent space, use elasticae to interpolate between them, and then map these elasticae paths to image space. We then compare results from our framework to using various SOTA DNN models. Fig. 3 (top) shows results for $\alpha =$ chair. These interpolations are computed in the latent space but visualized in the image space. Each row shows an interpolation between two fixed points in $\mathcal{R}_{test}$ at the left and right. Different rows correspond to different techniques. The bottom row shows the ground truth geodesic $t \mapsto \mathbb{P}(x(t)O^{\alpha})$, where $x(t)$ is a geodesic in $SO(3)$. We observe that the path obtained

by our model is consistently closest to the ground truth. This outcome is representative of the results we obtained for several experiments.

To quantify performance, we perform extensive experiments on three 3D objects: $\alpha \in \{\text{chair, sports car, zebra}\}$. We compute 100 different interpolation paths using randomly selected pose pairs in $\mathcal{R}_{test}$ (3696 points). For each time index t, we calculate the mean values of the errors (point values and tangent values) and plot them in Fig. 3. The errors are naturally close to zero at the start and the end, and are highest at the center. As these plots exhibit, the interpolation errors are the smallest using our method when compared to RHVAE, SVAE, and AE-StyleGAN2.

Ablation Studies: To evaluate the key components of GP-StyleGAN2, we perform ablation studies that add them sequentially to an AE-StyleGAN2 baseline model. We study six models which differ in: (1) the type of interpolation: linear or elastica, (2) the loss function for training E: inclusion of geometry-preserving terms L_e and L_g (defined in Section 3.1) or not, and (3) PCA for image pre-processing: PCA or no PCA. For each of the six models, the experimental setup is an analysis of interpolation path accuracy as above. The results of these experiments performed on the chair object are summarized in Table 1. The table lists the means over 100 experiments of the average interpolation error summed over all time indices t in the path. Comparing Models 1, 2 versus the others shows dramatic gains due to the introduction of the geometry-preserving terms. Comparisons of Models 3, 4, and 5 versus Model 6 show individual benefits of the use of both L_e and L_g over L_e alone, elasticae over linear interpolation, and PCA reduction.

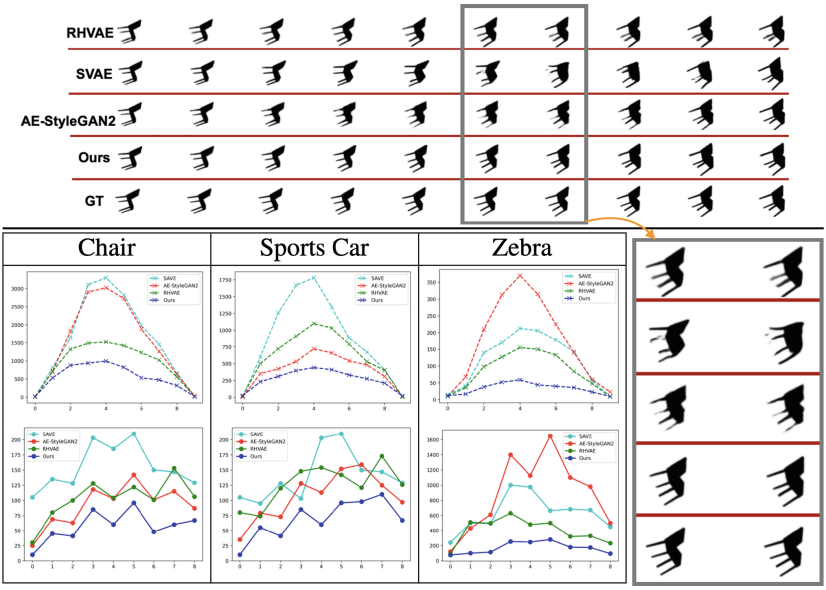

Fig. 3. Top: Interpolated paths between the original (left) and rotated poses (right) using various methods. Bottom: Average squared errors over 100 interpolated paths for each object, and zoom from top. First row: Average SEs for interpolated points. Second row: Average SEs for velocities.

The complete Model 6 (GP-StyleGAN2) including all of the components substantially outperforms the others.

Computational Cost: The computational cost of training for the four methods are as follows: AE-StyleGAN2 - 20.64 $min/500$ epochs, RHVAE - 12.78 $min/500$ epochs, SVAE - 8.78 $min/500$ epochs, and GP-StyleGAN2 - 38.41 $min/500$ epochs. The computational cost for interpolating a path between two test points is: AE-StyleGAN2 - 0.02 sec, RHVAE - 62.36 sec, SVAE - 0.01 sec, and GP-StyleGAN2 - 0.13 sec.

Manifold Estimation: To estimate the latent pose manifold $\widehat{\mathcal{M}}^{\alpha}$, we randomly select 800 sparse points from $\mathcal{R}_{test}$ and map them using the trained Φ. For each point in this latent space, we identify its five nearest-neighbors using $SO(3)$ distance d_s and interpolate between these neighbors using elasticae with eight intermediate points. This results in an $\widehat{\mathcal{M}}^{\alpha}$ with 32,800 total points.

Table 1. (**Ablation studies**): Average total squared errors over 100 interpolated paths (for chair) under different models. AE-loss denotes the standard loss function of AE-StyleGAN2.

Model Features	**Model 1**	**Model 2**	**Model 3**	**Model 4**	**Model 5**	**Model 6**
AE-StyleGAN2	yes	yes	yes	yes	yes	yes
Interpolation	linear	elasticae	elasticae	linear	elasticae	elasticae
Loss function	AE-loss	AE-loss	AE-loss + L_e	AE-loss + L_e + L_g	AE-loss + L_e + L_g	AE-loss + L_e + L_g
PCA	no	no	yes	yes	no	yes
Summary	**Model 1**	**Model 2**	**Model 3**	**Model 4**	**Model 5**	**Model 6 (Ours)**
Mean error	1638.97	1654.32	698.14	687.02	561.70	**436.11**

4.2 Results: Analyzing Latent Map Φ Using Test Data

We investigate the geometry-preserving properties of the mapping Φ by applying it to the complete test set $\mathcal{R}_{test}$. As described above, $\mathcal{R}_{test}$ is a set of 3696 points in the same patch $\mathcal{P}$ as $\mathcal{R}_{train}$ but disjoint from it. We exploit the grid structure of $\mathcal{R}_{test}$ to interpret the range space of Φ visually, create paths that bridge distant points, and verify the goal of distance preservation on a large scale.

Visualizing $\Phi(\mathcal{R}_{test})$: First we visualize the mapping of $\mathcal{R}_{test}$ into $\mathbb{R}^d$ using the learnt Φ. Here we investigate the geometry of the underlying manifold $\mathcal{M}^{\alpha}$ using this set's grid structure rather than densifying interpolations. The latent space plots shown here use the first three PCA axes of the mapped points in $\mathbb{R}^5$. We note that the subset of $SO(3)$ used in these experiments is topologically a box and that the first three singular values accounted for most of the variation in these examples. Figure 4 displays two viewing angles of the GP-StyleGAN2 latent space embeddings of the chair, car, and zebra objects. The three clouds look remarkably similar, all resembling shell-like segments of a thickened sphere, despite the vastly different shapes of the original objects.

Traversing Distant Points in $\Phi(\mathcal{R}_{test})$: In this experiment, we first endow the set $\Phi(\mathcal{R}_{test})$ with a graph structure determined by $SO(3)$ neighbors. Each point in $SO(3)$ has 26 neighbors; see the supplement for details. We then arbitrarily select two distant points in this set and create three paths between them: (1) the ground truth path

derived from the geodesic in $SO(3)$, (2) the shortest-length path through the graph found using Dijkstra's algorithm, and (3) a simple straight-line interpolation in latent space. Finally, we map the paths to image space. Figure 5 compares the results obtained using AE-StyleGAN2 and GP-StyleGAN2. We can derive multiple conclusions from these results. Firstly, the linear interpolations perform poorly under both models, highlighting the nonlinearity of the pose manifold. Notice how the linear interpolation loses its chair structure as it passes through the hollow space in the point cloud on the right. Secondly, GP-StyleGAN2 performs significantly better than AE-StyleGAN2 both visually and by error quantification: the Dijkstra path for GP-StyleGAN is nearly as good as the ground truth, while for AE-StyleGAN it is hardly any better than the straight line.

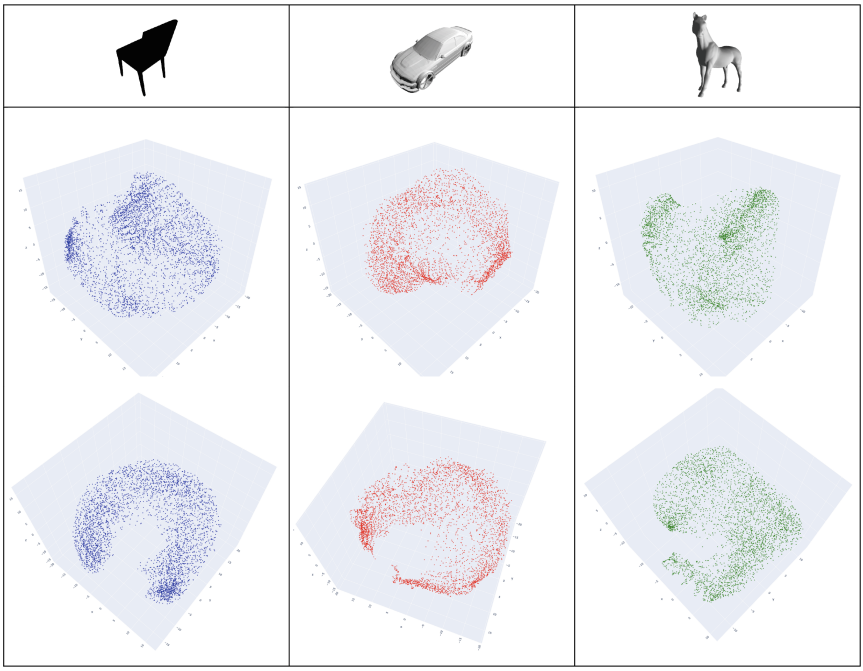

Fig. 4. Latent pose manifolds of chair, car, and zebra objects. Two views of each. We emphasize that these are not just points but are graphs with geometries.

4.3 Results: Exploiting Manifold Geometry

Once we have the estimated manifold $\widehat{\mathcal{M}}^{\alpha}$, we study its geometry in two different ways: Computing intrinsic image means and performing image denoising. (Fig. 6).

Mean Computations on $\widehat{\mathcal{M}}^{\alpha}$: We assess the utility of finding mean images in the latent space by comparing the decoded means to the ground truth, defined by images associated with $SO(3)$ Karcher means of sample rotations. Selecting 10 random images

$\{I_j \in \mathcal{R}_{test}\}$, we first compute a naive mean in $\mathbb{R}^{n^2}$, denoted as μ_I. Then we compute the Euclidean mean of $\{\Phi(I_j)\}$ in $\mathbb{R}^d$, project it to the nearest point in $\widehat{\mathcal{M}}^\alpha$, and map it back to image space to define μ_w. For comparison, we show the result $\tilde{\mu}_w$ obtained by performing these steps but skipping projection. The corresponding quantities under AE-StyleGAN2 are labeled as μ_w^A and $\tilde{\mu}_w^A$ respectively. Fig. 6 (top part) compares these results with the ground truth means μ_{gt}. The means estimated using GP-StyleGAN2 (green boxes) display realistic structure of the chair and better resemble the ground truth (orange boxes) than AE-StyleGAN2.

Image Denoising using $\widehat{\mathcal{M}}^\alpha$: The manifold geometry can also be used for denoising or cleaning corrupted images. A noisy image J can be mapped into latent space a $\Phi(J)$, projected to the nearest point $w \in \widehat{\mathcal{M}}^\alpha$, and mapped back as a cleaned image $\Phi^{-1}(w)$. Fig. 6 (bottom) shows images of the chair corrupted by adding noise and clutter, and compares results of cleaning using GP-StyleGAN2 and AE-StyleGAN2. The visual results and a histogram of reconstruction errors both show better outcomes using GP-StyleGAN2.

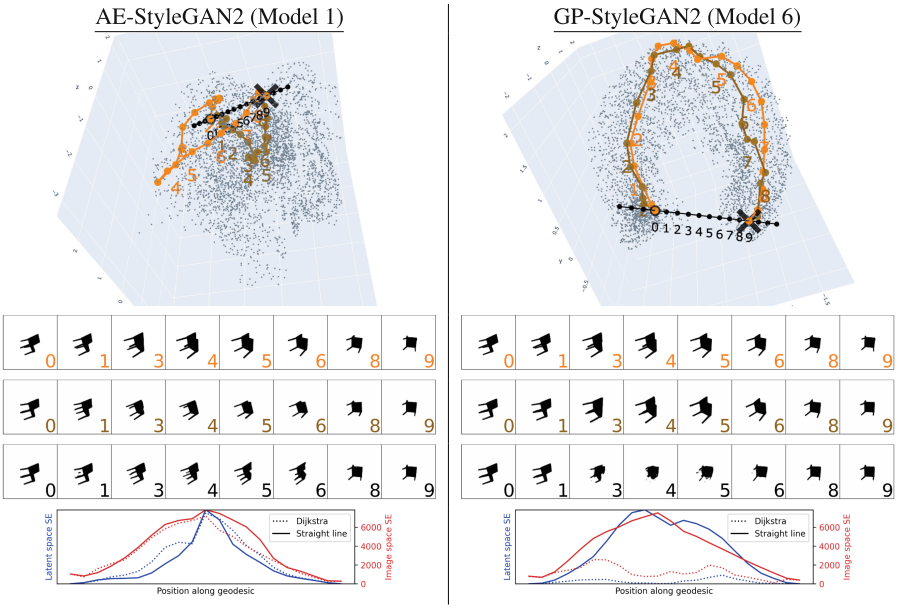

Fig. 5. Traversing distant points. Top and middle rows: Paths in latent and image space - geodesic GT (orange), Dijkstra on graph (brown), and straight line (black). Bottom: Euclidean squared errors along paths. Dijkstra paths under GP-StyleGAN2 perform best.

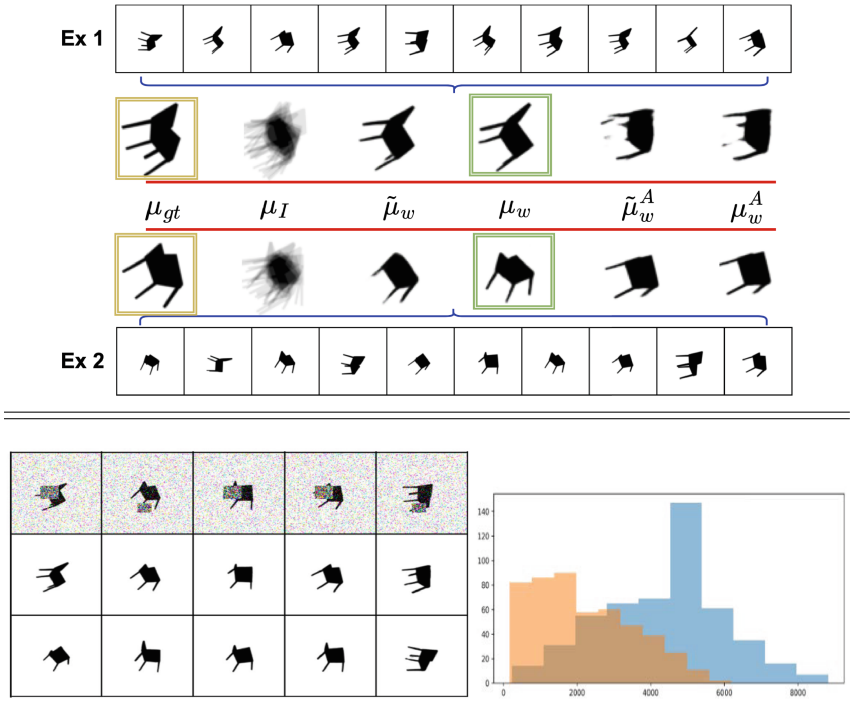

Fig. 6. Top Part: Manifold averaging on $\widehat{\mathcal{M}}^{\alpha}$. Ground truth μ_{gt} is compared with image space mean μ_I and different latent space means: $\mu_w, \tilde{\mu}_w$ computed with/without projection on $\widehat{\mathcal{M}}^{\alpha}$ using GP-StyleGAN2, and likewise $\mu_w^A, \tilde{\mu}_w^A$ using AE-StyleGAN2. Bottom Part: Left: Top row displays noisy images J, middle and bottom rows show corresponding denoised images using GP-StyleGAN2 and AE-StyleGAN2, respectively. Right: Histogram of squared errors for 500 noisy images using AE-StyleGAN2 (blue) and GP-StyleGAN2 (orange).

5 Conclusions

We introduced a new approach, GP-StyleGAN2, for characterizing pose manifolds of 3D objects. This approach preserves geometry when mapping to a low-dimensional latent space and creates dense manifold representations that account for nonlinearity using Euler's free elasticae. Comparisons of interpolations using GP-StyleGAN2 and various other methods (Fig. 3) showed superior results for our model visually and quantitatively. Ablation studies (Table 1) gave more detailed quantitative results that demonstrated improvements from including our two novel geometry-preserving terms and using elasticae rather than linear interpolation. Graph-based geodesic approximations pointed to regular but nonlinear geometry of pose manifolds (Fig. 5), in stark contrast to past conclusions of linear geometry for latent space image data. We also found that the use of manifold geometry improved mean computations and image denoising (Fig. 6).

While there is still much progress to be made, the overall success of GP-StyleGAN2 shows a step in the direction of truly learning the geometry of pose manifolds.

Acknowledgements. This research was supported in part by the grants DARPA-PA-21-04-05-FP-052, NIH R01-GM135927, and NSF IIS 1955154 to AS and NSF award 2323086 to PT.

References

1. Clara.io. https://clara.io
2. Arvanitidis, G., Hansen, L.K., Hauberg, S.: Latent space oddity on the curvature of deep generative models. In: Proc. of International Conference on Learning Representations (2018)
3. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. **15**(6), 1373–1396 (2003)
4. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. **35**, 1798–1828 (2013)
5. Chadebec, C., Allassonniere, S.: A geometric perspective on variational autoencoders. In: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (eds.) Advances in Neural Information Processing Systems (2022), https://openreview.net/forum?id=PBmJC6rDnR6
6. Chen, N., Klushyn, A., Ferroni, F., Bayer, J., Van Der Smagt, P.: Learning flat latent manifolds with vaes. arXiv preprint arXiv:2002.04881 (2020)
7. Coifman, R.R., Lafon, S.: Diffusion maps. Appl. Comput. Harmon. Anal. **21**(1), 5–30 (2006)
8. Davidson, T.R., Falorsi, L., De Cao, N., Kipf, T., Tomczak, J.M.: Hyperspherical variational auto-encoders. arXiv preprint arXiv:1804.00891 (2018)
9. Detlefsen, N.S., Hauberg, S., Boomsma, W.: Learning meaningful representations of protein sequences. Nat. Commun. **13**(1), 1914 (2022)
10. Donoho, D.L., Grimes, C.: Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. Proc. of the National Academy of Sciences **100**(10), 5591–5596 (2003)
11. Duque, A.F., Morin, S., Wolf, G., Moon, K.R.: Geometry regularized autoencoders. IEEE Trans. Pattern Anal. Mach. Intell. **45**(6), 7381–7394 (2022)
12. Goodfellow, I., et al.: Generative adversarial networks. Comm. of the ACM **63**(11), 139–144 (2020)
13. Grenander, U., Srivastava, A., Miller, M.I.: Asymptotic performance analysis of Bayesian object recognition. IEEE Trans. Inf. Theory **46**(4), 1658–66 (2000)
14. Han, L., Musunuri, S.H., Min, M.R., Gao, R., Tian, Y., Metaxas, D.: AE-StyleGAN: Improved training of style-based auto-encoders. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 3134–3143 (2022)
15. Hardin, D.P., Michaels, T., Saff, E.B.: A comparison of popular point configurations on $\mathbb{S}^2$. arXiv preprint arXiv:1607.04590 (2016)
16. Hauberg, S.: Only Bayes should learn a manifold (on the estimation of differential geometric structure from data). arXiv preprint arXiv:1806.04994 (2018)
17. Huh, I., et al.: Isometric quotient variational auto-encoders for structure-preserving representation learning. Neural Information Processing Systems **36** (2024)
18. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. In: International Conference on Learning Representations (2018)
19. Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., Aila, T.: Training generative adversarial networks with limited data. Adv. Neural. Inf. Process. Syst. **33**, 12104–12114 (2020)

20. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Computer Vision and Pattern Recognition. pp. 4401–4410 (2019)
21. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR). San Diega, CA, USA (2015)
22. Kingma, D.P., Welling, M.: Auto-encoding variational Bayes. arXiv preprint arXiv:1312.6114 (2013)
23. Kühnel, L., Fletcher, T., Joshi, S., Sommer, S.: Latent space geometric statistics. In: Pattern Recognition. ICPR International Workshops and Challenges, pp. 163–178. Springer International Publishing, Cham (2021)
24. Lim, J., Kim, J., Lee, Y., Jang, C., Park, F.C.: Graph geometry-preserving autoencoders. In: Forty-first International Conference on Machine Learning (2024)
25. Linnér, A.: Existence of free nonclosed euler-bernoulli elastica. Nonlinear Analysis: Theory, Methods & Applications **21**(8), 575–593 (1993)
26. Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. Journal of machine learning research **9**(11) (2008)
27. McQueen, J., Meila, M., Joncas, D.: Nearly isometric embedding by relaxation. Advances in Neural Information Processing Systems **29** (2016)
28. Mio, W., Srivastava, A., Klassen, E.: Interpolations with elasticae in euclidean spaces. Q. Appl. Math. **62**(2), 359–378 (2004)
29. Mumford, D.: Elastica and computer vision p. 491–506 (1994)
30. Sáez de Ocáriz Borde, H., Arroyo, A., Morales, I., Posner, I., Dong, X.: Neural latent geometry search: Product manifold inference via gromov-hausdorff-informed bayesian optimization. Advances in Neural Information Processing Systems **36** (2024)
31. Pai, G., Talmon, R., Bronstein, A., Kimmel, R.: Dimal: Deep isometric manifold learning using sparse geodesic sampling. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 819–828. IEEE (2019)
32. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
33. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)
34. Schlömer, N.: meshio: Tools for mesh files. https://doi.org/10.5281/zenodo.1173115, https://github.com/nschloe/meshio
35. Shao, H., Kumar, A., Fletcher, P.T.: The Riemannian geometry of deep generative models. arXiv **abs/1711.08014** (2017)
36. Shukla, A., Uppal, S., Bhagat, S., Anand, S., Turaga, P.: Geometry of deep generative models for disentangled representations. ICVGIP 2018, Association for Computing Machinery, New York, NY, USA (2020)
37. Singer, A., Wu, H.T.: Vector diffusion maps and the connection laplacian. Commun. Pure Appl. Math. **65**(8), 1067–1144 (2012)
38. Singh, A., Nag, K.: Structure-preserving deep autoencoder-based dimensionality reduction for data visualization. In: 2021 IEEE/ACIS 22nd International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD). pp. 43–48. IEEE (2021)
39. Swinbank, R., James Purser, R.: Fibonacci grids: A novel approach to global modelling. Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography **132**(619), 1769–1793 (2006)
40. Tenenbaum, J.B., Silva, V.d., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. science **290**(5500), 2319–2323 (2000)
41. Yershova, A., Jain, S., Lavalle, S.M., Mitchell, J.C.: Generating uniform incremental grids on SO(3) using the hopf fibration. Intl. journal of robotics research **29**(7), 801–812 (2010)

42. Yonghyeon, L., Yoon, S., Son, M., Park, F.C.: Regularized autoencoders for isometric representation learning. In: International Conference on Learning Representations (2021)
43. Zhang, S., Jiang, W.: Data-informed geometric space selection. Advances in Neural Information Processing Systems **36** (2024)
44. Zhang, Z., Zha, H.: Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. SIAM J. Sci. Comput. **26**(1), 313–338 (2004)