Intrinsic Point Cloud Interpolation via Dual Latent Space Navigation

Marie-Julie Rakotosaona[1] and Maks Ovsjanikov[1]

LIX, Ecole Polytechnique, IP Paris
{mrakotos,maks}@lix.polytechnique.fr

Abstract. We present a learning-based method for interpolating and manipulating 3D shapes represented as point clouds, that is explicitly designed to preserve intrinsic shape properties. Our approach is based on constructing a dual encoding space that enables shape synthesis and, at the same time, provides links to the intrinsic shape information, which is typically not available on point cloud data. Our method works in a single pass and avoids expensive optimization, employed by existing techniques. Furthermore, the strong regularization provided by our dual latent space approach also helps to improve shape recovery in challenging settings from noisy point clouds across different datasets. Extensive experiments show that our method results in more realistic and smoother interpolations compared to baselines. Both the code and our pre-trained network can be found online: https://github.com/mrakotosaon/intrinsic_interpolations.

Keywords: 3D Point Clouds, 3D Reconstruction, Deep Learning: Applications, Methodology, and Theory

1 Introduction

A core problem in 3D computer vision is to manipulate and analyze shapes represented as point clouds. Compared to other representations such as triangle meshes or dense voxel grids, point clouds are distinguished by their generality, simplicity and flexibility. For these reasons, and especially with the introduction of PointNet and its variants [38,39,43], point clouds have gained popularity in machine learning applications, including point-based *generative models*.

Unfortunately the flexibility of the point cloud representation also comes at a cost, as it does not encode any *topological* or intrinsic metric information of the underlying object. Thus, methods trained on point cloud data can, by their nature, be insensitive to distortion that might appear on generated shapes. This problem is particularly prominent in 3D shape interpolation, where a common approach is to generate intermediate shapes by interpolating the learned latent vectors. In this case, even if the end-shapes are realistic, the intermediate ones can have severe distortions that are very difficult to detect and correct using only point-based information. More generally, several works have observed that point

Fig. 1. Intrinsic point cloud interpolation between points from an incomplete scan with holes (left, reconstructed in first blue column) and points from a noisy mesh (right, reconstructed in last blue column). Our method both reconstructs the shape better and produces a more natural interpolation than a PointNet-based auto-encoder.

cloud-based generative models can fail to capture the space of natural shapes [34,28], making it difficult to navigate them while maintaining realism.

In this paper, we introduce a novel architecture aimed specifically at injecting intrinsic information into a generative point-based network. Our method works by learning consistent mappings across the latent space obtained by a point cloud auto-encoder and a feature encoding that captures the *intrinsic* shape structure. We show that these two components can be optimized using shapes represented as triangle meshes during training. The resulting linked latent space combines the strengths of a generative latent model and of intrinsic surface information. Finally, we use the learned networks at test time on raw 3D point clouds that are neither in correspondence with the training shapes, nor contain any connectivity information.

Our approach is general and not only enables smooth interpolations, while avoiding expensive iterative optimization, but also, as we show bellow, leads to more accurate shape reconstruction from noisy point clouds across different datasets. We demonstrate on a wide range of experiments that our approach can significantly improve upon recent baselines in terms of the accuracy of shape recovery as well as realism and smoothness of shape interpolation.

2 Related Work

Shape interpolation, also known as morphing in certain contexts, is a vast and well-researched area of computer vision and computer graphics (see [33] for a survey of the early approaches). Below we review only most relevant works and focus on structure-preserving mesh interpolation, and on recent learning-based methods that operate on point clouds.

Classical methods for 3D shape interpolation have primarily focused on designing well-founded geometric metrics, and associated optimization methods that enable smooth structure-preserving interpolations. Early works in this direction include variants of as-rigid-as-possible interpolation and modeling [2,29,51]

and various *representations* of shape deformation that facilitate specific transformation types, e.g. [47,27,35,16,46] among many others.

A somewhat more principled framework is provided by the notion of *shape spaces* [30,37] in which interpolation can be phrased as computing a shortest path (geodesic). In the case of surface meshes, this approach was studied in detail in [31] and then extended in numerous follow-up works, including [49,17,26,24,25] among others. These approaches enjoy a rich theoretical foundation, but are typically restricted to shapes having a fixed connectivity and can lead to difficult optimization problems at test time.

We also note a recent set of methods based on the formalism of *optimal transport* [6,44,10] which have also been used for shape interpolation. These approaches treat the input shapes as probability measures that are interpolated via efficient optimization techniques.

Somewhat more closely related to ours are data-driven and feature-based interpolation methods. These include interpolation based on hand-crafted features [19,28] or on exploring various *local* shape spaces obtained by analyzing a shape collection [20,52,40]. Such techniques work well if the input shapes are sufficiently similar, but require triangle meshes and dense point-wise correspondences, or a single template that is fitted to all input data to build a statistical model, e.g. [23,7,8].

Most closely related to ours are recent generative models that operate directly on unorganized point clouds [1,34,36]. These methods are often inspired by the seminal work of PointNet and its variants [38,39] and are typically based on autoencoder architectures that allow shape exploration by manipulation in the latent space. Despite significant progress in this area, however, the structure of learned latent spaces is typically not easy to control or analyze. For example, it is well-known (see e.g. [28]) that commonly-used *linear interpolation* in latent space can give rise to unrealistic shapes that are difficult to detect and rectify.

Common approaches to address these issues include extensive data augmentation [22], adversarial losses that penalize unrealistic instances [34,5] or explicit modeling of the metric in the latent space. The latter can be done by computing the Jacobian of the decoder from the latent to the embedding space [13,42] or using feature-based metrics at test time [32,18]. Unfortunately, as we show below, such techniques either lead to difficult optimization problems at test time, or can still result in significant shape distortion.

Contribution In this paper, we propose to address the challenges mentioned above by building a *dual latent space* that combines a learned point-based autoencoder with another parallel encoding that captures the intrinsic shape metric given by the lengths of edges of triangle meshes, required only during training. This second encoding exploits the insights of mesh-based interpolation techniques [31,25,41] that highlight the importance of *interpolating the intrinsic surface information* rather than the point coordinates. We combine these two encodings by constructing dense networks that "translate" between the two latent spaces, and enable smooth and accurate interpolation without relying on correspondences or solving expensive optimization problems at test time.

3 Motivation & Background

Our main goal is to design a method capable of *efficiently and accurately* interpolating shapes represented as point clouds. This problem is challenging for several key reasons. First, most existing theoretically well-founded axiomatic 3D shape interpolation methods [31,26,24,25] assume the input shapes to be represented as triangle meshes with fixed connectivity in 1-1 correspondence, and furthermore typically require extensive optimization at test time. On the other hand, learning-based approaches typically embed the shapes in a compact latent space, and interpolate them by linearly interpolating the corresponding latent vectors [1,50]. Although this approach is efficient, the metric in the latent space is typically not well-understood and therefore *linear interpolation* in this space may result in unrealistic and heavily distorted shapes. Classical methods such as Variational Auto-Encoders (VAEs) help introduce regularity into the latent space, and enable more accurate generative models, but offer little control on the distances and thus interpolation in the latent space. To address this challenge, several recent approaches have proposed ways to endow the latent space with a metric and help recover geodesic distances [32,13,18]. However, these methods again typically involve expensive computations such as the Jacobian of the decoder network, and *optimization at test time*.

Within this context, our main goal is to combine the formalism and shape metrics proposed by geometric methods [31,25] with the accuracy and flexibility of data-driven techniques while maintaining efficiency and scalability.

Shape Interpolation Energy We first recall the intrinsic shape interpolation energy introduced in [31]. Suppose we are given a pair of shapes M, N represented as triangle meshes with fixed connectivity, so that $M = (\mathcal{V}_M, \mathcal{E})$, and $N = (\mathcal{V}_N, \mathcal{E})$, where $\mathcal{V}, \mathcal{E}$ represent the coordinates of the points and the fixed set of edges respectively. An interpolating sequence is defined by a one parameter family $S_t = (\mathcal{V}_t, \mathcal{E})$, such that $\mathcal{V}_0 = \mathcal{V}_M$, and $\mathcal{V}_1 = \mathcal{V}_N$. Denoting by $v_i(t)$ the trajectory of vertex i in S_t, the basic time-continuous intrinsic interpolation energy of S_t is defined as:

$$E_{\text{cont}}(S_t) = \int_{t=0}^{1} \sum_{(i,j)\in\mathcal{E}} \left(\frac{\partial \|v_i(t) - v_j(t)\|_2}{\partial t} \right)^2 dt. \tag{1}$$

This energy measures the integral of the change of all the edge lengths in the interpolation sequence. It can be discretized in time by sampling the interval $[0 \dots 1]$ with samples t_k, where $k = 1 \dots n_k$. When the time samples are uniform, resulting in a discrete set of shapes $\{S_k\}$, this leads to the discrete energy:

$$E_{\text{disc}}(\{S_k\}) = \sum_{k=2}^{n_k} \sum_{ij\in\mathcal{E}} \left(\|v_i(t_k) - v_j(t_k)\|_2 - \|v_i(t_{k-1}) - v_j(t_{k-1})\|_2 \right)^2. \tag{2}$$

This discrete energy simply measures the sum of the squared differences between lengths of edges across consecutive shapes in the sequence. The authors of [31]

argue that computing a shape sequence between M and N that minimizes such a distortion energy results in an accurate interpolation of the two shapes (more precisely in [31] use squared edge lengths and employ an additional weak regularization, which we omit for simplicity and as we have found it to be unnecessary in our case). Note that both the continuous and discrete versions of the energy promote *as-isometric-as-possible* shape interpolations. Specifically they aim to minimize the *intrinsic distortion* by promoting intermediate meshes whose edge lengths interpolate as well as possible the edge lengths of M, N, without requiring the two input shapes to be isometric themselves.

Despite the simplicity and elegance of the intrinsic interpolation energy, minimizing it directly is challenging as it leads to large non-convex optimization problems over vertex coordinates. Indeed, additional regularization is typically required to achieve realistic interpolation across large motions [31,25]. Perhaps even more importantly, the assumption of input shapes having a fixed triangle mesh and being in 1-1 correspondence is very restrictive in practice.

Latent space optimization In the context of data-driven techniques the standard way to manipulate shapes is through operations in the *latent space*. This is done by first training an auto-encoder (AE) architecture and then using the learned latent space for shape manipulation. Specifically, an encoder is trained to associate a *latent vector* l_S to each 3D shape S in a training set via $l_S = \text{enc}(S)$, while the decoder is trained so that $\text{dec}(l_S) \approx S$. Given two shapes M, N, the interpolation is performed by first computing their latent vectors, l_M, l_N and then constructing an interpolating sequence via $S_t = \text{dec}(tl_N + (1-t)l_M)$ [1,50].

Unfortunately, basic *linear interpolation* in the latent space can produce significant artefacts in the resulting reconstructed shapes (see, e.g., Figure 2). More broadly, the metric (distance) structure of the latent space is not easy to control, as the encoder-decoder architecture is typically trained only to be able to *reconstruct* the shapes, and does not capture any information about distances in the latent space.

3.1 Metric interpolation in a learned space

To overcome this limitation, perhaps the simplest approach is to use a learned latent space, but to compute an interpolating sequence while minimizing the intrinsic distortion energy of the decoded shapes explicitly. Namely, after training an auto-encoder, given the source and target shapes with latent vectors l_M, l_N, one can construct a set of samples l_k in the latent space and *at test time* optimize:

$$\begin{aligned} \min_{l_1, l_2, \ldots, l_k} E_{\text{disc}}(\{S_k\}), \text{s.t. } S_i = \text{dec}(l_i), i = 1 \ldots k, \\ S_0 = \text{dec}(l_M), S_{k+1} = \text{dec}(l_N). \end{aligned} \tag{3}$$

This operation employs the fact that a decoder can be trained to always produce shapes that are in 1-1 correspondence, thus making it possible to compare the decoded shapes $\{S_k\}$.

To solve this problem, the samples l_k can be initialized through linear interpolation of l_M, l_N, and Eq. (3) can optimized via gradient descent using the pre-trained decoder network. This is more efficient than directly optimizing Eq. (2) through the coordinates of the vertices, as the latent space typically has a much smaller dimensionality. Intuitively, this procedure adjusts the latent vectors to correct the distortion induced by using the Euclidean metric in the latent space. In addition, the use of a pre-trained decoder acts as the regularization (required by purely geometric methods) to produce realistic shapes.

Despite leading to significant improvement compared to the basic linear interpolation in the latent space, this approach has two key limitations 1) it requires potentially expensive optimization at test time, and 2) its accuracy is limited by the initial linear interpolation in the latent space. The latter issue is particularly prominent since the latent space is not related to the intrinsic distortion energy and therefore linear interpolation can be a suboptimal initialization for the problem in Eq. (3).

Intuition Our main intuition is that in the absence of any constraints, the intrinsic distortion energy E_{disc} is minimized by the family of shapes that linearly interpolates the edge lengths between the source and the target. This, however is not guaranteed to lead to actual 3D shapes, both because integrability conditions must hold to ensure that edges can be assembled into a consistent mesh [48] and because interpolated shapes might not be realistic from the point of view of the training data. Therefore, we build *two* auto-encoder networks that capture, respectively, point coordinates and lengths of edges of underlying meshes (available at training) so that Euclidean distances in the latent space depend linearly on distances between lists of ordered edges. We then build two "translation" or mapping networks across the two latent spaces. Finally, after training these networks, at test time, we linearly interpolate in the edge length latent space, but recover each shape by mapping onto the shape space and reconstructing using the shape decoder. As we show below, this results in smooth and realistic shape interpolation without relying on correspondences or optimization at test time.

4 Method

4.1 Overview

Figure 3 gives an overview of our network. As mentioned above, it consists of three main building blocks and training steps: a shape auto-encoder, an auto-encoder of the edge lengths of the underlying mesh, and two "translation" networks that enable communication between the two latent spaces. These networks are used at test time to endow given point clouds with intrinsic information which is then used, in particular, for more accurate point cloud interpolation. We assume that the training data is given in the form of triangle meshes with fixed connectivity, while the input at test time consists of unorganized point clouds. In the following section we describe our architecture and the associated losses, while the implementation and experimental details are given in Section 5.

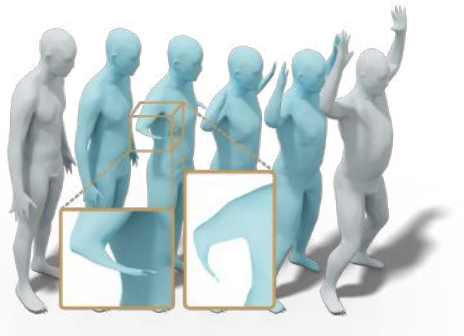

Fig. 2. Linear interpolation in the latent space of the shape AE produces artefacts, as the interpolation is close to linear interpolation of the coordinates.

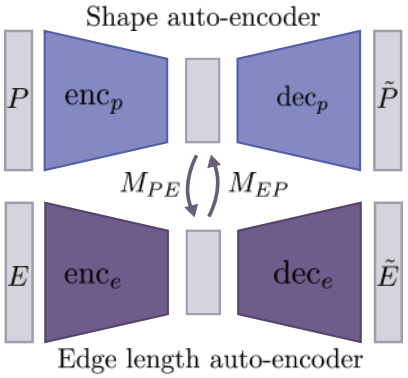

Fig. 3. Our overall architecture. We build two auto-encoders that capture the shape and edge length structure respectively, as well as two mapping networks M_{PE} and M_{EP} that "translate" across the two latent spaces.

4.2 Architecture

Shape auto-encoder. Our first building block (Fig. 3 top) consists of a shape auto-encoder, based on the PointNet architecture [38]. We denote the encoder and decoder networks as enc_p and dec_p respectively (we provide the exact implementation details and compare to a VAE in the supplementary). To train this network we use the basic L_2 reconstruction loss, since we assume that the input shapes are in 1-1 correspondence. This leads to the following training loss:

$$L_{rec}(P) = \frac{1}{n}\sum_{i=1}^{n} \|P_i - \tilde{P}_i\|^2, \text{ where } \tilde{P} = \text{dec}_p\left(\text{enc}_p(P)\right). \tag{4}$$

Here P is a training shape, the summation is done over all points in the point cloud, and P_i represents the 3D coordinates of point i.

Importantly, our point-based encoder enc_p inherits the permutation invariance of PointNet [38], which is crucial in real applications. Specifically, this allows us to encode arbitrary point clouds at test time even if they have significantly different sampling and are not in correspondence with the training data.

Edge length auto-encoder As observed in previous works and as we confirm below, the shape AE can capture the structure of individual shapes, but often fails to reflect the overall structure of *shape space*, which is particularly evident during shape interpolation. We address this by constructing a separate auto-encoder whose latent space captures the *intrinsic* shape information, and by learning mappings across the two latent spaces.

For this, we first build an auto-encoder (enc_e, dec_e) with dense layers that aims to reconstruct a list of edge lengths. Note that since we assume 1-1 correspondence at training time, the list of lengths of edges can be given in canonical

(e.g., lexicographic with respect to vertex ids) order. We therefore build an auto-encoder that encodes this list into a compact vector and decodes it back from the latent representation. Our training loss for this part consists of two components: an L_2 error on the predicted edge lengths and an additional term that promotes linearity in the learned latent space:

$$L_e(E_A) = \|\mathrm{dec}_e(\mathrm{enc}_e(E_A)) - E_A\|^2 \tag{5}$$

$$L_{lin}(E_A, E_B) = \left\| \frac{\mathrm{dec}_e(\mathrm{enc}_e(E_A)) + \mathrm{dec}_e(\mathrm{enc}_e(E_B))}{2} - \mathrm{dec}_e\left(\frac{\mathrm{enc}_e(E_A) + \mathrm{enc}_e(E_B)}{2}\right)\right\|^2. \tag{6}$$

Here E_A, E_B are the lists of edge lengths corresponding to the triangle meshes A, B given during training. Our motivation for the loss L_{lin} is to explicitly encourage linear structure, which promotes smoothness of interpolated edge lengths and thus, as we show below, minimizes intrinsic distortion.

Mapping networks Given two pretrained auto-encoders described above, we train two dense mapping networks that translate elements between the two latent spaces. We use M_{PE} and M_{EP} to denote the networks that translate an element from the shape (resp. edge) latent space to the edge (resp. shape) latent space.

To define the losses we use to train these two networks, for a training mesh A we let $l_A = \mathrm{enc}_p(A)$ denote the latent vector associated with A by the shape encoder. Recall that when training the shape AE we compare A with $\mathrm{dec}_p(l_A)$. To train our mapping networks M_{PE} and M_{EP} we instead compare A with $\mathrm{dec}_p\left(M_{EP}(M_{PE}(l_A)\right)$. In other words, rather than decoding directly from l_A we first map it to the edge length latent space (via M_{PE}). We then map the result back to the shape latent space (via M_{EP}) and finally decode the 3D shape. We denote the shape reconstructed this way by $\tilde{A} = \mathrm{dec}_p(M_{EP}(M_{PE}(\mathrm{enc}_p(A))))$. We compare $\tilde{A}$ to the original shape A, which leads to the following loss:

$$L_{map1}(A) = d^{\mathrm{rot}}(\tilde{A}, A). \tag{7}$$

Here d^{rot} is a *rotation invariant* shape distance comparing the original and reconstructed shape. We use it since the list of edge lengths can only encode a shape up to rigid motion [21]. Specifically, we first compute the optimal rigid transformation between the input shape A and the predicted point cloud $\tilde{A}$ using Kabsh algorithm [4]. We then compute the mean square error between the coordinates after alignment. As shown in [28] this loss is differentiable using the derivative of the Singular Value Decomposition.

Our second loss compares the edge lengths of the reconstructed shape $\tilde{A}$ to the edge lengths of A. For this we use the standard L_2 norm squared:

$$L_{map2}(A) = \|E_A - E_{\tilde{A}}\|_2^2, \tag{8}$$

where E_A denotes the list of edge lengths of shape A.

Our last loss considers a similar difference but starting in the edge length latent space, rather than the shape one. Specifically, given a shape A with the list of edge lengths E_A, we first encode it to the edge length latent space via $\text{enc}_e(E_A)$. We then translate the resulting latent vector to the shape latent space (via M_{EP}) and back to the edge length latent space (via M_{PE}), and finally decode the result using dec_e. This leads to the following loss:

$$L_{map3}(A) = \|\text{dec}_e(M_{PE}(M_{EP}(\text{enc}_e(E_A)))) - E_A\|_2^2, \tag{9}$$

Our overall loss is then simply a weighted sum of three terms $\alpha L_{map1} + \beta L_{map2} + \gamma L_{map3}$ for shapes given at training where γ is non-zero. We evaluate other possible losses in the supplementary materials.

Network Training To summarize, we train our overall network architecture described in Figure 3 in three separate steps. First we train the shape-based auto-encoder using the loss given in Eq. (4). Then we train the edge length auto-encoder using the sum of the losses in Eq. (5) and Eq. (6). Finally we train the dense networks M_{EP} and M_{PE} using the sum of the three losses in Eq. (7), Eq. (8), Eq. (9). We also experimented with training the different components jointly but have observed that the problem is both more difficult and the relative properties of the computed latent spaces become less pronounced when trained together, leading to less realistic reconstructions (Sec. 4.1 of the supplementary).

4.3 Navigating the restricted latent space

After training the networks as described above, we use them at test time for shape reconstruction and interpolation. We stress that at test time we do not use the edge encoder and decoder networks enc_e, dec_e, as they require canonical edge ordering. Instead we use the permutation invariant shape auto-encoder and the mapping networks M_{PE}, M_{EP} to better preserve intrinsic shape properties.

Interpolation Given two possibly noisy unorganized point clouds P_A and P_B we first compute their associated edge-based latent codes: $m_A = M_{PE}(\text{enc}_p(P_A))$ and $m_B = M_{PE}(\text{enc}_p(P_B))$. Here we use the permutation-invariance of our encoder enc_p allowing to encode unordered point sets. We then linearly interpolate between m_A and m_B but use the *shape decoder* dec_p for reconstruction. Thus, we compute a family of intermediate point clouds as follows:

$$P_\alpha = \text{dec}_p\left(M_{EP}\left((1-\alpha)m_A + \alpha m_B\right)\right),\ \alpha \in [0 \dots 1] \tag{10}$$

In other words, we interpolate the latent codes in the edge-based latent space, but perform the reconstruction via the shape decoder dec_p. This allows us to make sure that the reconstructed shapes are both realistic and their intrinsic metric is interpolated smoothly. Note that unlike the purely geometric methods, such as [31], our approach does not rely on the given mesh structure at test time. Instead, we employ the learned edge-based latent space as a proxy for recovering

the intrinsic shape structure, which as we show below, is sufficient to obtain accurate and smooth interpolations.

Since the edge length auto-encoder is fully rotation invariant, it is necessary to align the output shapes at test time. We can do so easily by using the same optimal rigid transformation as used to compute Eq. (9).

Reconstruction Given a point cloud P_A we also use our trained architecture for shape recovery via $S = \text{dec}_p(M_{EP}(M_{PE}(\text{enc}_p(P_A))))$. Here we use the fact that the edge-length latent space helps to regularize the shape space avoiding noisy or distorted output.

4.4 Interpretation

Our approach can be interpreted both in terms of capturing the structure of individual 3D shapes and of the entire *shape space*. For the former, our shape and edge-length auto-encoders help to capture, respectively, the *extrinsic* and *intrinsic* information of the underlying surface. Jointly, they enable more accurate shape recovery and comparison. In this context, our approach is related to methods for reconstructing a shape from its intrinsic metric. This problem, while possible theoretically [21], is computationally challenging and error prone in practice [48,11,15,14]. By using a *learned* latent space our reconstruction is both efficient and leads to realistic results.

In terms of the shape space, the latent vectors of the shape auto-encoder provide a way to parametrize the space of realistic 3D shapes while the edge-length latent space helps to impose a *distance structure* on that space. This is similar to the standard approach in Riemannian geometry [12] where the manifold structure of a space and the *metric* on it are encoded separately. We highlight this interpretation in the supplementary, and leave its complete exploration as exciting future work.

4.5 Unsupervised training

Our method can be adapted to the unsupervised context where the 1-1 correspondences are not provided during training. Contrary to our main pipeline, we cannot compute the edge lengths directly from the training data. However, we can encourage the model to produce a consistent mesh as described in [22]. We initialize the weights by pre-training on a selected mesh using the reconstruction loss L_{rec} described in (4) and train the model using Chamfer distance and regularization losses to keep the triangulation consistent. Finally, we can train the edge-length auto-encoder by using the output of the shape auto-encoder as training data. We describe this process in detail in the supplementary materials.

5 Results

Datasets We train our networks on two different datasets: humans and animals. For humans, we use the dataset proposed in [28]. The dataset contains 17440

shapes subsampled to 1k points from DFAUST [9] and SURREAL [45]. The test set contains 10 sub-collections (character + action sequence, each consisting of 80 shapes) that are isolated from the training set of DFAUST and 2000 shapes from SURREAL dataset. During training the area of each shape is normalized to a common value. For animals we sample 12000 shapes from the SMAL dataset [53]. We sample an equal number of shapes from the 5 categories (big cats, horses, cows, hippos, dogs) to build a training set of 10000 shapes and a testset of 2000 shapes. We simplify the shapes from SMAL to 2002 points per mesh. The animal dataset provides challenging shape pairs that are far from being isometric, some of which we highlight in the supplementary video.

5.1 Shape interpolation

We evaluate our method on our core application of shape interpolation and compare against six different recent baselines. Namely, we compare to three data-driven methods, by performing linear interpolations in the latent spaces of auto-encoders using PointNet [38] and PointNet++ [39] architectures as well as the pre-trained auto-encoder proposed in the state-of-the-art non-rigid shape matching method 3D-CODED [22].

We also compare to three optimization-based geometric methods, by building on the ideas from [31,42,13]. We produce our first two baselines by initializing a linear path in latent space of our shape auto-encoder and optimizing each sample via 1000 steps of gradient descent. We use GD EL to denote the method that optimizes E_{disc} as described in Eq.(3), and G2 L2 to denote the method that minimizes the L^2 variance over the interpolated shape coordinates as described in [42]. Finally we compare to a method simplified from [31] (GD Coord.), in which we first initialize a path by linearly interpolating the coordinates of source and target shapes. Similarly to GD EL, we minimize the discrete interpolation energy E_{disc} using gradient descent on the point coordinates directly.

Remark that GD Coord., GD L2 and GD EL methods all rely on gradient descent to compute each interpolation *at test time*. In other words, these approaches all require to solve a highly non-trivial optimization problem during interpolation, leading to additional computational cost and parameters (learning rate, number of iterations). In contrast, our method outputs a smooth interpolation in a single pass.

	Direct inference				Optimization based		
	Ours	PointNet	3D-Coded	PointNet++	GD L2	GD EL	GD Coord.
EL	**0.231**	0.3510	0.6130	0.2993	0.3631	0.2985	**0.0345**
Area (10^{-4})	**1.261**	1.773	3.137	1.586	1.838	1.714	**0.248**
Volume (10^{-4})	**0.342**	1.613	1.243	335.2	1.483	1.703	**0.152**

Table 1. We report the mean squared variance of the edge length (EL), per surface area and total shape volume over the interpolations of 100 shape pairs. Our method achieves lowest variance across all intrinsic features among direct inference methods. Note that GD coord. leads to interpolation with low distortion, as it optimizes the coordinates directly but produces unrealistic shapes (see Figure 4).

To evaluate the interpolations we sample 50 shapes from the DFAUST testset using farthest point sampling. We then test on 100 random pairs from those 50 shapes. We use our pipeline trained with $\alpha = 30$, $\beta = 1200$ and $\gamma = 800$ in the mapping networks loss described in Section 4.2. We provide an ablation study on the choice of losses in supplementary materials.

Table 1 shows quantitative comparisons. Given an interpolation path (S_n) obtained by each method, we compute the mean squared variance of various shape features f on the path. We consider three features: lengths of edges, area of faces and overall volume enclosed by the shape (computed from the mesh embedding). For each of these, we compute the sum of the squared differences across all instances in the interpolating sequence:

$$Var_f(S_n) = \frac{1}{n-1} \sum_{i=2}^{n} \|f(S_i) - f(S_{i-1})\|^2. \tag{11}$$

Intuitively, we expect a good interpolation method to result in smooth interpolations which would have low variance across all of the intrinsic shape properties. When comparing with PointNet++ as it inputs normalized bounding boxes, we normalize the total area of each output. The large volume variance of this baseline is primarily due to bad reconstruction quality of the input shapes.

As shown in Table 1 our method produces the best results among the direct data-driven methods and the best results over all the baselines except from GD Coord. This latter method is not data-driven and optimizes edge lengths directly on the coordinates without any constraints. As such, it produces shapes with low distortion but that are not realistic (see Figure 4). Furthermore, similarly to [31] it requires the input shapes to be represented as meshes in 1-1 correspondence.

In all qualitative figures, we visualize the minimum ratio between the linear interpolation of the ground truth edge lengths and the edge lengths of the produced shapes. We color-code this ratio to highlight areas of highest intrinsic distortion (shown in red). In Figure 4 we illustrate the interpolated shapes between the input source and target, shown in grey. We observe that PointNet AE and PointNet++ methods tend to produce results that are closer to linear interpolation of the coordinates. As highlighted above, we notice that while GD Coord. has low variance in the interpolated intrinsic features, the reconstructed shapes do not look natural. Overall, our method presents less distortions and more smooth interpolations compared to all baselines. We present more comparisons and evaluations in a video and in the supplementary.

We further evaluate our model on the SMAL dataset. To build the interpolation pairs from the test set, we sample 10 shapes per category by farthest points sampling. We then choose 100 random pairs from that dataset. In Figure 5 we show results of interpolating between two horses. We observe that linear interpolation in the shape latent space leads to shape distortions such as shorter legs (middle) and wrong shape size estimation (top left). The Shape AE (resp. Ours) produces a edge variance of 2.068 (resp. 1.548). Similarly to above, our method shows improvement at interpolating intrinsic information. We provide detailed numerical evaluation of interpolations on SMAL in supplementary materials.

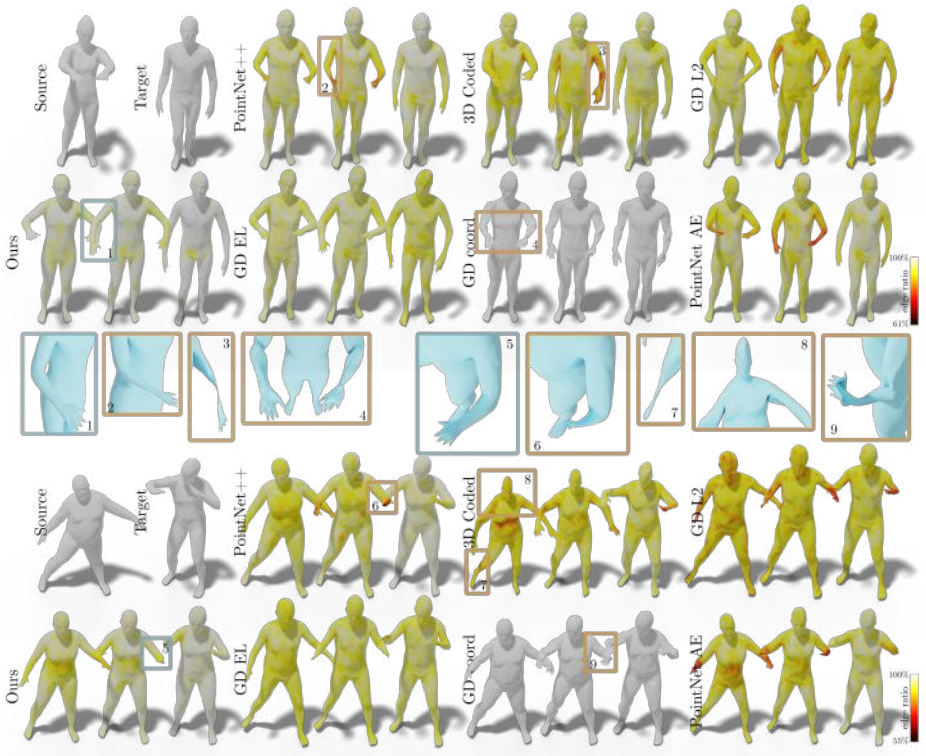

Fig. 4. Qualitative comparison of interpolation on DFAUST testset. We display the edge ratio between the linear interpolation of the target and source edges and the produced interpolation.

Interpolation in the unsupervised case. The unsupervised Shape AE (resp. Ours) produces a edge variance of 0.599 (resp. 0.394). While we observe better results in the supervised setting, our method nevertheless produces quantitative and qualitative improvement over the linear interpolation in latent space. We provide further numerical and qualitative results in the supplementary materials.

5.2 Shape reconstruction

We also evaluate the accuracy of our model for shape reconstruction on the DFAUST/SURREAL testset. In Table 2, we compare the reconstruction accuracy to the base models. We measure intrinsic features: edge length and per triangle area reconstruction loss, and extrinsic L^2 coordinates reconstruction loss. Our method reconstructs the input shape intrinsic features better that the PointNet AE while producing comparable extrinsic reconstruction loss.

We further evaluate the generalization capacity of our network by evaluating on the SCAPE [3] dataset. For testing we sample 1000 random points from the

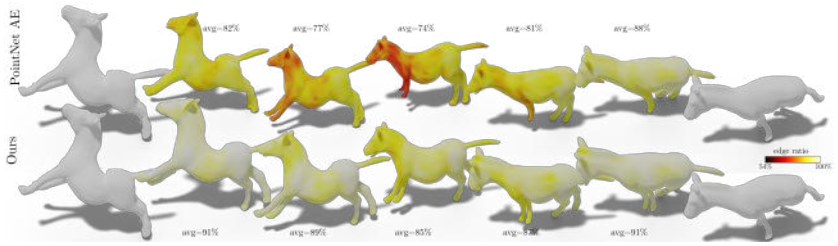

Fig. 5. Interpolation of two horses from the SMAL dataset.

	EL (10^{-5})	PC (10^{-4})	area (10^{-8})
PointNet AE	3.023	**2.120**	2.454
Edge Length AE	3.127	-	-
Ours	1.641	2.572	1.562

Table 2. Mean squared reconstruction losses on the humans testset. Edge length reconstruction loss (EL), Point cloud coordinates reconstruction loss (PC) and per triangle area difference.

	CD (10^{-3})	volume (10^{-5})	area
Shape AE	4.703	30.851	0.1382
Ours	4.135	9.47	0.047

Table 3. Reconstruction accuracy on the SCAPE dataset. Chamfer distance (CD), mean squared total volume difference and total area difference.

surface of each mesh. Table 3 shows an improvement in the reconstruction for our method. We observe even higher relative performance when comparing the total volume and total area of the reconstructed shapes which give a sense of the perceived quality of the shapes. Shape distortions are often related to shrunk or disproportional body parts. We show qualitative results on reconstruction in the supplementary materials. Overall, our method produces more precise and natural reconstructions. Finally, as shown in Figure 1, our method is robust to high levels of noise (left), holes, and missing parts (right). We provide further reconstruction examples in the supplementary materials.

6 Conclusion, Limitations & Future Work

We presented a method for interpolating unorganized point clouds. Key to our approach is a dual latent space that both captures the extrinsic and intrinsic shape information, given by edge lengths provided during training. We demonstrate that our approach leads to significant improvement over existing methods, both in terms of interpolation smoothness and quality of the generated results. In the future, we plan to extend our method to incorporate other features such as semantic classes or segmentations. It would also be interesting to explore our dual encoding space in other applications on images or graphs.

Acknowledgements Parts of this work were supported by the KAUST CRG-2017-3426 Award and the ERC Starting Grant No. 758800 (EXPROTEA).

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3D point clouds. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 40–49. Stockholmsmässan, Stockholm Sweden (10–15 Jul 2018)
2. Alexa, M., Cohen-Or, D., Levin, D.: As-rigid-as-possible shape interpolation. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques. pp. 157–164. ACM Press/Addison-Wesley Publishing Co. (2000)
3. Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., Davis, J.: Scape: shape completion and animation of people. In: ACM transactions on graphics (TOG). vol. 24, pp. 408–416. ACM (2005)
4. Arun, K.S., Huang, T.S., Blostein, S.D.: Least-squares fitting of two 3-d point sets. IEEE Transactions on pattern analysis and machine intelligence **1**(5), 698–700 (1987)
5. Ben-Hamu, H., Maron, H., Kezurer, I., Avineri, G., Lipman, Y.: Multi-chart generative surface modeling. In: SIGGRAPH Asia 2018 Technical Papers. p. 215. ACM (2018)
6. Benamou, J.D., Brenier, Y.: A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. Numerische Mathematik **84**(3), 375–393 (2000)
7. Bogo, F., Romero, J., Loper, M., Black, M.J.: Faust: Dataset and evaluation for 3d mesh registration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3794–3801 (2014)
8. Bogo, F., Romero, J., Pons-Moll, G., Black, M.J.: Dynamic faust: Registering human bodies in motion. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6233–6242 (2017)
9. Bogo, F., Romero, J., Pons-Moll, G., Black, M.J.: Dynamic FAUST: Registering human bodies in motion. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (Jul 2017)
10. Bonneel, N., Rabin, J., Peyré, G., Pfister, H.: Sliced and radon wasserstein barycenters of measures. Journal of Mathematical Imaging and Vision **51**(1), 22–45 (2015)
11. Boscaini, D., Eynard, D., Kourounis, D., Bronstein, M.M.: Shape-from-operator: Recovering shapes from intrinsic operators. In: Computer Graphics Forum. vol. 34, pp. 265–274. Wiley Online Library (2015)
12. Carmo, M.P.d.: Riemannian geometry. Birkhäuser (1992)
13. Chen, N., Klushyn, A., Kurle, R., Jiang, X., Bayer, J., van der Smagt, P.: Metrics for deep generative models. arXiv preprint arXiv:1711.01204 (2017)
14. Chern, A., Knöppel, F., Pinkall, U., Schröder, P.: Shape from metric. ACM Transactions on Graphics (TOG) **37**(4), 63 (2018)
15. Corman, E., Solomon, J., Ben-Chen, M., Guibas, L., Ovsjanikov, M.: Functional characterization of intrinsic and extrinsic geometry. ACM Transactions on Graphics (TOG) **36**(2), 1–17 (2017)
16. Crane, K., Pinkall, U., Schröder, P.: Spin transformations of discrete surfaces. ACM Transactions on Graphics (TOG) **30**(4), 104 (2011)
17. Freifeld, O., Black, M.J.: Lie bodies: A manifold representation of 3d human shape. In: European Conference on Computer Vision. pp. 1–14. Springer (2012)
18. Frenzel, M.F., Teleaga, B., Ushio, A.: Latent space cartography: Generalised metric-inspired measures and measure-based transformations for generative models. arXiv preprint arXiv:1902.02113 (2019)

19. Gao, L., Chen, S.Y., Lai, Y.K., Xia, S.: Data-driven shape interpolation and morphing editing. Computer Graphics Forum **36**(8), 19–31 (2017)
20. Gao, L., Lai, Y.K., Huang, Q.X., Hu, S.M.: A data-driven approach to realistic shape morphing. Computer graphics forum **32**(2pt4), 449–457 (2013)
21. Gluck, H.: Almost all simply connected closed surfaces are rigid. In: Geometric topology, pp. 225–239. Springer (1975)
22. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: 3d-coded: 3d correspondences by deep deformation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 230–246 (2018)
23. Hasler, N., Stoll, C., Sunkel, M., Rosenhahn, B., Seidel, H.P.: A statistical model of human pose and body shape. Computer graphics forum **28**(2), 337–346 (2009)
24. Heeren, B., Rumpf, M., Schröder, P., Wardetzky, M., Wirth, B.: Exploring the geometry of the space of shells. In: Computer Graphics Forum. vol. 33, pp. 247–256. Wiley Online Library (2014)
25. Heeren, B., Rumpf, M., Schröder, P., Wardetzky, M., Wirth, B.: Splines in the space of shells. Computer Graphics Forum **35**(5), 111–120 (2016)
26. Heeren, B., Rumpf, M., Wardetzky, M., Wirth, B.: Time-discrete geodesics in the space of shells. Computer Graphics Forum **31**(5), 1755–1764 (2012)
27. Huang, J., Shi, X., Liu, X., Zhou, K., Wei, L.Y., Teng, S.H., Bao, H., Guo, B., Shum, H.Y.: Subspace gradient domain mesh deformation. ACM Transactions on Graphics (TOG) **25**(3), 1126–1134 (2006)
28. Huang, R., Rakotosaona, M.J., Achlioptas, P., Guibas, L., Ovsjanikov, M.: Operatornet: Recovering 3d shapes from difference operators. arXiv preprint arXiv:1904.10754 (2019)
29. Igarashi, T., Moscovich, T., Hughes, J.F.: As-rigid-as-possible shape manipulation. ACM transactions on Graphics (TOG) **24**(3), 1134–1141 (2005)
30. Kendall, D.G.: Shape manifolds, procrustean metrics, and complex projective spaces. Bulletin of the London Mathematical Society **16**(2), 81–121 (1984)
31. Kilian, M., Mitra, N.J., Pottmann, H.: Geometric modeling in shape space. ACM Transactions on Graphics (TOG) **26**(3), 64 (2007)
32. Laine, S.: Feature-based metrics for exploring the latent space of generative models (2018)
33. Lazarus, F., Verroust, A.: Three-dimensional metamorphosis: a survey. The Visual Computer **14**(8), 373–389 (1998)
34. Li, C.L., Zaheer, M., Zhang, Y., Poczos, B., Salakhutdinov, R.: Point cloud GAN. arXiv preprint arXiv:1810.05795 (2018)
35. Lipman, Y., Cohen-Or, D., Gal, R., Levin, D.: Volume and shape preservation via moving frame manipulation. ACM Transactions on Graphics (TOG) **26**(1), 5 (2007)
36. Liu, X., Han, Z., Wen, X., Liu, Y.S., Zwicker, M.: L2g auto-encoder: Understanding point clouds by local-to-global reconstruction with hierarchical self-attention. In: Proceedings of the 27th ACM International Conference on Multimedia. pp. 989–997. ACM (2019)
37. Michor, P.W., Mumford, D.B.: Riemannian geometries on spaces of plane curves. Journal of the European Mathematical Society (2006)
38. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proc. CVPR. pp. 652–660 (2017)
39. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems. pp. 5099–5108 (2017)

40. von Radziewsky, P., Eisemann, E., Seidel, H.P., Hildebrandt, K.: Optimized subspaces for deformation-based modeling and shape interpolation. Computers & Graphics **58**, 128–138 (2016)
41. Sassen, J., Heeren, B., Hildebrandt, K., Rumpf, M.: Solving Variational Problems Using Nonlinear Rotation-invariant Coordinates. In: Bommes, D., Huang, H. (eds.) Symposium on Geometry Processing 2019- Posters. The Eurographics Association (2019)
42. Shao, H., Kumar, A., Thomas Fletcher, P.: The riemannian geometry of deep generative models. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 315–323 (2018)
43. Shen, Y., Feng, C., Yang, Y., Tian, D.: Mining point cloud local structures by kernel correlation and graph pooling. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4548–4557 (2018)
44. Solomon, J., De Goes, F., Peyré, G., Cuturi, M., Butscher, A., Nguyen, A., Du, T., Guibas, L.: Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. ACM Transactions on Graphics (TOG) **34**(4), 66 (2015)
45. Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M.J., Laptev, I., Schmid, C.: Learning from synthetic humans. In: CVPR (2017)
46. Vaxman, A., Müller, C., Weber, O.: Conformal mesh deformations with möbius transformations. ACM Transactions on Graphics (TOG) **34**(4), 55 (2015)
47. Von Funck, W., Theisel, H., Seidel, H.P.: Vector field based shape deformations. ACM Transactions on Graphics (TOG) **25**(3), 1118–1125 (2006)
48. Wang, Y., Liu, B., Tong, Y.: Linear surface reconstruction from discrete fundamental forms on triangle meshes. In: Computer Graphics Forum. vol. 31, pp. 2277–2287. Wiley Online Library (2012)
49. Wirth, B., Bar, L., Rumpf, M., Sapiro, G.: A continuum mechanical approach to geodesics in shape space. International Journal of Computer Vision **93**(3), 293–318 (2011)
50. Wu, J., Zhang, C., Xue, T., Freeman, B., Tenenbaum, J.: Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In: Advances in neural information processing systems. pp. 82–90 (2016)
51. Xu, D., Zhang, H., Wang, Q., Bao, H.: Poisson shape interpolation. Graphical models **68**(3), 268–281 (2006)
52. Zhang, Z., Li, G., Lu, H., Ouyang, Y., Yin, M., Xian, C.: Fast as-isometric-as-possible shape interpolation. Computers & Graphics **46**, 244–256 (2015)
53. Zuffi, S., Kanazawa, A., Jacobs, D., Black, M.J.: 3D menagerie: Modeling the 3D shape and pose of animals. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (Jul 2017)