Intrinsic Point Cloud Interpolation via Dual Latent Space Navigation

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

LIX, Ecole Polytechnique
{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.

1 Introduction

A core problem in 3D computer vision is to analyze, encode and manipulate shapes represented as point clouds. Point clouds are particularly useful compared to other representations due to their generality, simplicity and flexibility compared to more complex data-structures such as triangle meshes or dense voxel grids. For all of these reasons, and with the introduction of PointNet and its variants [37,38,42], point clouds have also gained popularity in machine learning applications, including point-based *generative models*.

Unfortunately the flexibility of point cloud representations also comes at a cost, as they do not encode any topological or intrinsic metric information of the underlying surface. 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 generative models built on point cloud data can fail to capture the space of natural shapes, e.g., [33,27], 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

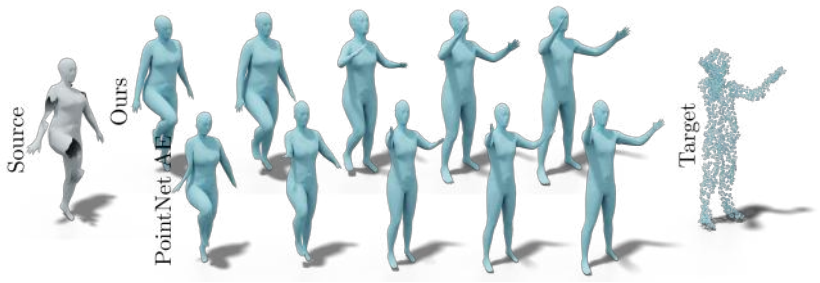

Fig. 1. Intrinsic point cloud interpolation between points from an incomplete scan with holes (left, reconstructed in first blue column) and points from a noisy mesh (right, reconstructed in last blue column). Our method both reconstructs the shape better and produces a more natural interpolation than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

by learning consistent mappings across the latent space obtained by a point cloud auto-encoder and another feature encoding that captures the intrinsic shape structure. We show that these two parts can be optimized jointly using shapes represented as triangle meshes during training. The resulting linked latent space combines the strengths of a generative latent model, and the 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 leads to 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 and smoothness of the interpolation and enables a range of novel applications.

2 Related Work

Shape interpolation, also known as morphing in certain contexts, and exploration is a vast and well-researched area of computer vision and computer graphics (see [32] for a survey of the early approaches) and its full overview is beyond the scope of this paper. Below we review works most closely related to ours, and concentrate, in particular, on either structure-preserving mesh interpolation techniques, or recent learning-based methods that focus 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 [2,28,50] and various *representations* of shape deformation that facilitate specific transformation types, e.g. [46,26,34,15,45] among many others.

A somewhat more principled framework is provided by the notion of *shape spaces* [29,36] 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 [30] and then extended in numerous follow-up works, including [48,16,25,23,24] among many 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* [6,43,10] 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 [18,27] or by exploring various *local* shape spaces obtained by analyzing a shape collection [19,51,39]. These approaches 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. [22,7,8].

Most closely related to ours are recent generative models that operate directly on point clouds [1,33,35]. These methods are largely inspired by the seminal work of PointNet and its variants [37,38] 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. [27]) 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 [21], using adversarial losses that aim to penalize unrealistic instances [33,5] or modifying 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 [12,41] or using feature-based metrics at test time [31,17]. 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 this challenge by building a *dual latent space* that combines a learned shape encoding in a point-based generative model with another parallel encoding that aims to capture the intrinsic shape metric given by the lengths of edges of triangle meshes only required during training. This second encoding exploits the insights of mesh-based interpolation techniques [30,24,40] 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 at test time without relying on correspondences or expensive optimization problems.

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 [30,25,23,24] 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 shapes by linearly interpolating their corresponding latent vectors [1,49]. 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 [31,12,17]. However, these methods again typically involve expensive computations such as the Jacobian of the decoder network, and expensive *optimization at test time*.

Within this context, our main goal is to combine the formalism and shape metrics proposed by geometric methods [30,24] 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 [30]. Specifically 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 these 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 [30] 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 [30] an additional weak regularization is employed, 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 *isometric 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 [30,24]. 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*, by first training an auto-encoder (AE) architecture and then shape manipulation (e.g. interpolation) in the learned latent space. 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 done by first computing their latent vectors, l_M, l_N and then constructing an interpolating sequence via $S_t = \text{dec}(tl_N + (1-t)l_M)$ [1,49].

Unfortunately, basic *linear interpolation* in the latent space can produce significant artefacts in the resulting reconstructed shapes as we can see in 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 significantly more efficient than directly optimizing Eq. (2) through the coordinates of the vertices, as the dimensionality of the latent space is typically much smaller. Intuitively, this procedure locally 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 To design our approach we propose to build two auto-encoder networks: one that intuitively creates a parametrization of the set of realistic shapes, and the other that captures intrinsic distortion, and thus distances between shapes in shape space. This second network builds a latent space that encodes lengths of edges of underlying meshes (available at training) so that Euclidean distances in the latent space correspond to distances between lists of ordered edges. 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 additional integrability conditions must hold to ensure that edges can be assembled into a consistent mesh [47] and because interpolated shapes might not be realistic from the point of view of the training data. Therefore, we also build two "translation" or mapping networks that allow us to go between the edge length and shape latent spaces. Finally, after training these networks, at test time, we linearly interpolate in the edge length latent space, but recover each shape by projecting onto the shape space and reconstructing using the shape decoder. As we show below, this results in both 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.

4.2 Architecture

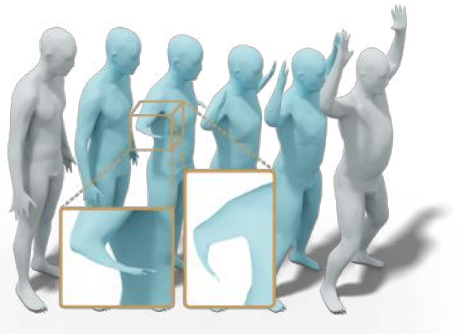

Fig. 2. Linear interpolation in the latent space of the shape AE produces artefacts, as the interpolation is close to linear interpolation of the coordinates.

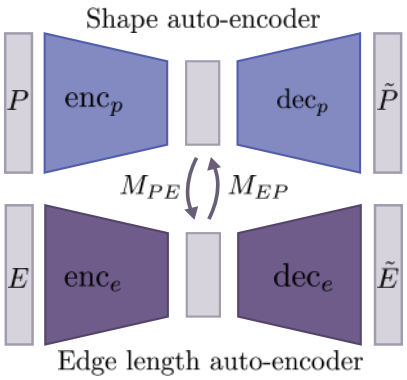

Fig. 3. Our overall architecture. We build two auto-encoders that capture the shape and edge length structure respectively, as well as two mapping networks M_{PE} and M_{EP} that "translate" across the two latent spaces.

Shape auto-encoder. Our first building block (Figure 3 top) consists of a shape auto-encoder, based on the PointNet architecture [37]. We denote the encoder and decoder networks as enc_p and dec_p respectively (we provide the exact implementation details F). 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 [37], 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 bellow, 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 in shape interpolation applications. We address this issue by constructing a separate auto-encoder that aims to capture 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:

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

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

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 second loss L_{lin} is to explicitly encourage linear structure, which promotes smoothness of interpolated edge lengths and thus, as we demonstrate bellow, 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 $l_A = \text{enc}_p(A)$ denote the latent vector associated with A by the shape encoder. Recall that when training the shape AE we compare A with $\text{dec}_p(l_A)$. To train our mapping networks M_{PE} and M_{EP} we instead compare A with $\text{dec}_p\left(M_{EP}(M_{PE}(l_A)\right)$. 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 $\tilde{A} = \text{dec}_p(M_{EP}(M_{PE}(\text{enc}_p(A))))$. We compare $\tilde{A}$ to the original shape A, which leads to the following loss:

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

Here d^{rot} is a *rotation invariant* shape distance comparing the the original and reconstructed shape. We use it since the list of edge lengths can only encode a shape up to rigid motion [20]. 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 [27] 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:

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

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 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.

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. (13). Then we train the edge length auto-encoder using the sum of the losses in Eq. (16) 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.

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 based auto-encoder and the mapping networks M_{PE}, M_{EP} to better preserve intrinsic shape properties.

Our main observation is that the latent space associated with the shape auto-encoder provides a way to recover realistic point clouds, while the latent space of the edge length auto-encoder helps to impose a better distance structure in that space. Note that our approach is related to methods for reconstructing a shape from its edge lengths, which while possible theoretically [20], is computationally challenging and error prone in practice [47,11,14,13]. By using a learned shape space, however, our reconstruction is both efficient and leads to realistic shapes.

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 [30], 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 bellow, 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).

4.4 Unsupervised training

Our method can be adapted to an 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 [21]. We initialize the weights by pre-training on a selected mesh using the reconstruction loss L_{rec} described in (13) 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 E.

5 Results

Datasets We train our networks on two different datasets: humans and animals. For humans, we use the dataset proposed in [27]. The dataset contains 17440 shapes subsampled to 1k points from DFAUST [9] and SURREAL [44]. 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 [52]. 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 provided 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 [37] and PointNet++ [38] architectures as well as the pre-trained auto-encoder proposed in the state-of-the-art non-rigid shape matching method 3D-CODED [21].

We also compare to three optimization-based geometric methods, by building on the ideas from [30,41,12]. 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 L2 variance over the interpolated shape coordinates as described in [41]. Finally we compare to a method simplified from [30] (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	**0.2311**	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. We highlight that among the direct inference methods our method achieves lowest variance across all intrinsic features. We highlight the best numerical results per category. GD coord. leads to interpolation with low distortion, as it optimizes the coordinates directly however the shapes are not realistic (see Figure 5)

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 4.2. We provide an ablation study on the choice of losses in D.2.

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, overall surface area 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. To be fair when comparing with PointNet++ since it was trained on normalized bounding boxes and not area, we normalize the total area of each output. The large volume variance of this baseline is primarily due to bad reconstruction quality of the source and target 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 5). Furthermore, similarly to [30] 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 provide qualitative comparison of the linear interpolations in the basic shape (PointNet) AE latent space and the interpolation using our method. Our method preserves body type better (row 2) and interpolates well between a pair of shapes where the end results differs highly from the linear interpolation of the coordinates (row 4).

In Figure 5 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 the provided video.

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 6 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 B.2.

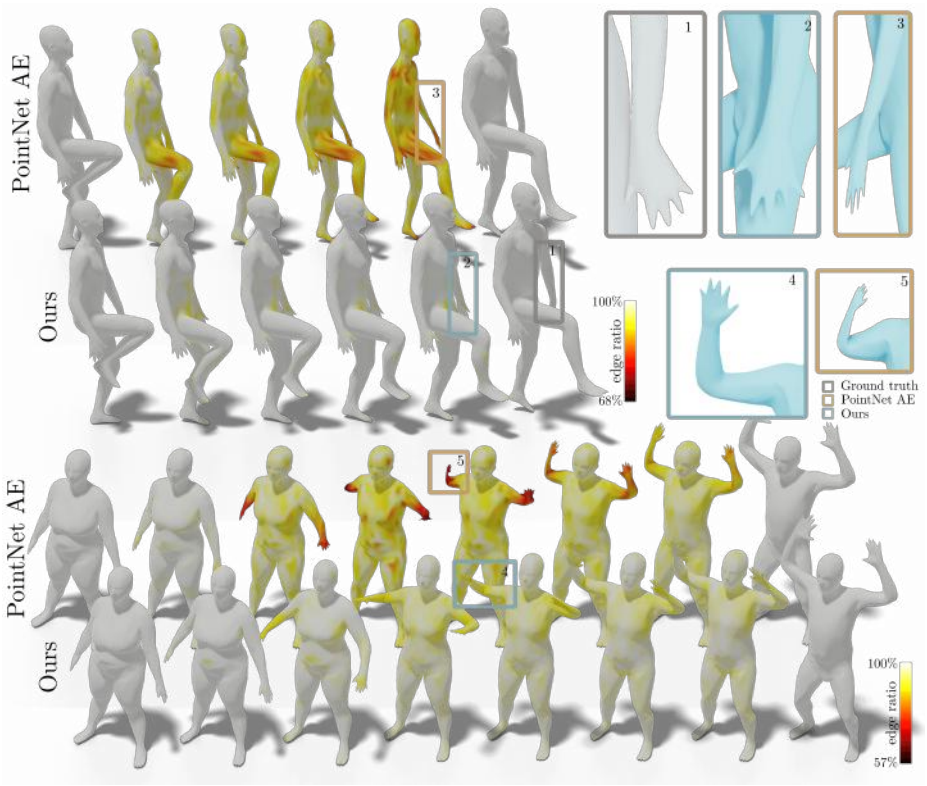

Fig. 4. We compare linear interpolations in PointNet AE latent space and interpolation using our approach. We visualize the ratio between the linear interpolation of edge lengths and edge lengths of the computed interpolations, to help highlight problematic areas.

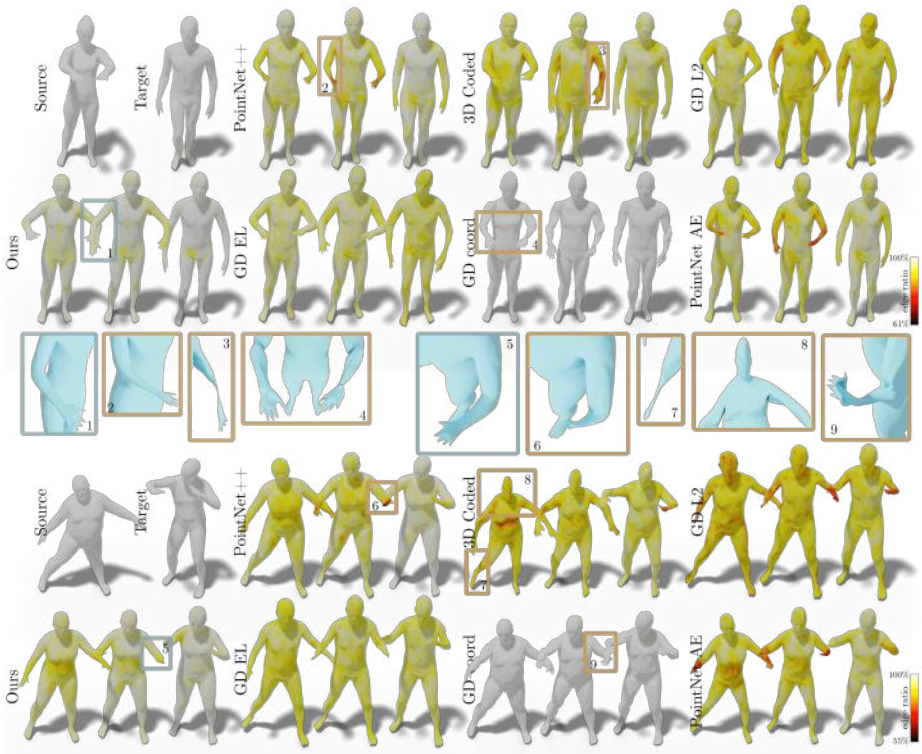

Fig. 5. Qualitative comparison of interpolation on DFAUST testset. We display the edge ratio between the linear interpolation of the target and source edges and the produced interpolation.

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 E.

5.2 Shape reconstruction

For our method, given an unordered point cloud P, we reconstruct the shapes by using the following combination of our trained networks $\text{dec}_p(M_{EP}(M_{PE}(\text{enc}_p(P))))$, which differs from the standard auto-encoder approach $\text{dec}_p(\text{enc}_p(P))$. Therefore, in this section we show that the additional regularization provided by our mapping networks M_{EP}, M_{PE} results in better shape reconstruction.

We evaluate the reconstruction accuracy of our model on the DFAUST/SURREAL testset. In Table 5, we compare the reconstruction accuracy to the base models. We measure intrinsic features: edge length and area per triangle $L2$ recon-

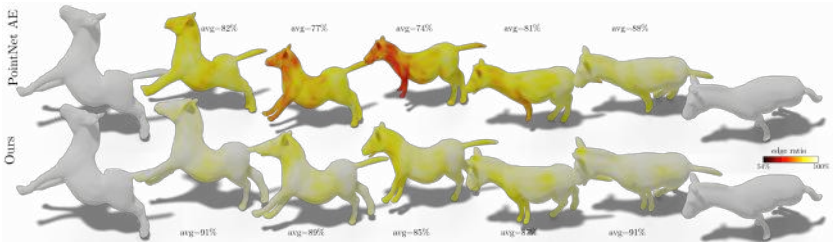

Fig. 6. Interpolation of two horses from SMAL dataset

	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 SCAPE dataset. We measure the Chamfer distance (CD), mean square total volume difference and MS total area difference

struction loss, and extrinsic features the $L2$ 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 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 7 on meshes from the DFAUST testset. To be fair to 3D-CODED, we normalize the total area of the output shapes. We evaluate this method before (3D-CODED) and after (3D-CODED*) their additional step of Chamfer Distance minimization. Note that in the case of 3D-CODED* additional optimization *at test time* is required to recompute the latent code that best approximates the input. Our method, on the other hand, performs the reconstruction in one shot. 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 C.

Fig. 7. Reconstruction of meshes from point clouds containing 1000 points, sampled from the underlying shape.

6 Conclusion, Limitations & Future Work

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

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. Stockholmsmssan, 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. 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)
13. Chern, A., Knöppel, F., Pinkall, U., Schröder, P.: Shape from metric. ACM Transactions on Graphics (TOG) **37**(4), 63 (2018)
14. 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)
15. Crane, K., Pinkall, U., Schröder, P.: Spin transformations of discrete surfaces. ACM Transactions on Graphics (TOG) **30**(4), 104 (2011)
16. 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)
17. 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)
18. 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)
19. 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)
20. Gluck, H.: Almost all simply connected closed surfaces are rigid. In: Geometric topology, pp. 225–239. Springer (1975)
21. 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)
22. 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)
23. 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)

24. Heeren, B., Rumpf, M., Schröder, P., Wardetzky, M., Wirth, B.: Splines in the space of shells. Computer Graphics Forum **35**(5), 111–120 (2016)
25. Heeren, B., Rumpf, M., Wardetzky, M., Wirth, B.: Time-discrete geodesics in the space of shells. Computer Graphics Forum **31**(5), 1755–1764 (2012)
26. 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)
27. Huang, R., Rakotosaona, M.J., Achlioptas, P., Guibas, L., Ovsjanikov, M.: Operatornet: Recovering 3d shapes from difference operators. arXiv preprint arXiv:1904.10754 (2019)
28. Igarashi, T., Moscovich, T., Hughes, J.F.: As-rigid-as-possible shape manipulation. ACM transactions on Graphics (TOG) **24**(3), 1134–1141 (2005)
29. Kendall, D.G.: Shape manifolds, procrustean metrics, and complex projective spaces. Bulletin of the London Mathematical Society **16**(2), 81–121 (1984)
30. Kilian, M., Mitra, N.J., Pottmann, H.: Geometric modeling in shape space. ACM Transactions on Graphics (TOG) **26**(3), 64 (2007)
31. Laine, S.: Feature-based metrics for exploring the latent space of generative models (2018)
32. Lazarus, F., Verroust, A.: Three-dimensional metamorphosis: a survey. The Visual Computer **14**(8), 373–389 (1998)
33. Li, C.L., Zaheer, M., Zhang, Y., Poczos, B., Salakhutdinov, R.: Point cloud GAN. arXiv preprint arXiv:1810.05795 (2018)
34. 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)
35. 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)
36. Michor, P.W., Mumford, D.B.: Riemannian geometries on spaces of plane curves. Journal of the European Mathematical Society (2006)
37. 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)
38. 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)
39. 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)
40. 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)
41. 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)
42. 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)

43. 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)
44. Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M.J., Laptev, I., Schmid, C.: Learning from synthetic humans. In: CVPR (2017)
45. Vaxman, A., Müller, C., Weber, O.: Conformal mesh deformations with möbius transformations. ACM Transactions on Graphics (TOG) **34**(4), 55 (2015)
46. Von Funck, W., Theisel, H., Seidel, H.P.: Vector field based shape deformations. ACM Transactions on Graphics (TOG) **25**(3), 1118–1125 (2006)
47. 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)
48. 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)
49. 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)
50. Xu, D., Zhang, H., Wang, Q., Bao, H.: Poisson shape interpolation. Graphical models **68**(3), 268–281 (2006)
51. 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)
52. 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)

Supplementary

A Overview

In Section B we provide additional illustrations of our shape interpolation method. In Section C we demonstrate the performance of our approach for *shape reconstruction* highlighting the utility our dual network for strong regularization of recovering high-quality shapes from noisy point clouds, as mentioned in the main manuscript. In Section D we provide an in-depth ablation study of our network design. In Section E we demonstrate the performance of our approach in the unsupervised case (when the training data is not in correspondence). Finally, in Section F we provide details of our architecture.

B Shape interpolation

B.1 Video and Comparison to Optimization-based Approaches

We provide a video which contains qualitative comparisons of interpolations on DFAUST and SMAL test sets with our main baselines. Note that our approach produces visually smoother interpolations with significantly lower distortions than all baselines across all shape pairs.

In the video we also provide comparisons with optimization-based approaches that achieve low distortion in Table 1 of the main manuscript. Specifically note that methods such as GD Coord. 1) require the input shapes to be in 1-1 correspondence 2) rely on expensive optimization at test time (for this reason, we compute these interpolations at half of the frame-rate), and most importantly 3), as shown in the accompanying video, as they are not learning-based, lead to non-realistic intermediate shapes.

B.2 Additional Evaluation

We further compare our method to the PointNet AE on the SMAL animals dataset. Table 4 reports the mean-squared variance of several shape features during interpolation of 100 pairs among 50 shapes obtained by farthest points sampling on this dataset. Note that our method produces significantly better quantitative results across all shape features

	edge length	area (10^{-3})	volume (10^{-2})
PointNet	2.068	3.742	2.754
Ours	**1.538**	**2.975**	**1.728**

Table 4. MS variance of various shape features obtained from interpolating 100 pairs among 50 shapes obtained by farthest points sampling on animals dataset (SMAL)

C Shape reconstruction

As mentioned in the main manuscript, our approach not only enables better interpolation, but also results in more accurate reconstructions from noisy input. Here we provide additional qualitative and quantitative evaluation of the reconstruction performance and comparison to different baseline methods.

In all of the experiments the training data is the combination of DFAUST and SURREAL datasets, and the test data is the DFAUST test shapes, both with and without noise.

Table 5 shows reconstruction results for several baselines on the 800 DFAUST test shapes. We report the edge length accuracy (EL), rotation-invariant point cloud reconstruction accuracy (PC) and per triangle area reconstruction accuracy (area). Note that our approach achieves the best overall reconstruction accuracy, especially on the intrinsic quantities and gives slightly worse reconstruction extrinsic loss (PC) compared to PointNet AE. We provide qualitative examples in Figure 7. Note that our method leads to both preservation of the overall shape structure and significantly less intrinsic distortion compared to all baselines.

Table 6 (left) shows reconstruction performance on noisy point clouds. Note that we test using our model which was trained on clean data. Each noisy point

	EL (10^{-5})	PC (10^{-4})	area (10^{-8})
PointNet AE	3.023	**2.120**	2.454
Edge Length AE	3.127	-	-
Ours $L_{1,2,3}$	**1.641**	2.572	**1.562**
3D-CODED	6.323	5.803	5.485
3D-CODED*	6.284	4.260	5.409
PointNet++	2.835	3.224	2.835

Table 5. Mean squared reconstruction losses on DFAUST testset. Edge length reconstruction loss (EL), Point cloud coordinates reconstruction loss (PC) and per triangle area difference

cloud is obtained by adding Gaussian noise magnitude 5% of the scale of the mesh to each vertex coordinate. We observe that our method outperforms the other baselines for all the features. Figure 8 shows reconstructed meshes from the noisy point clouds. Figure 9 shows reconstructed meshes from point clouds under-sampled to only 500 points. Notice that our method performs better at recovering the original pose and body type than the different baselines.

	Noisy dataset			Undersampled dataset		
	EL (10^{-5})	PC (10^{-4})	area (10^{-8})	EL (10^{-5})	PC (10^{-4})	area (10^{-8})
PointNet AE	5.663	8.538	5.650	3.847	**3.313**	2.810
Ours	**3.016**	**7.329**	**2.812**	**1.854**	3.587	**1.685**
3D-CODED	8.553	10.463	7.058	6.219	6.898	5.341
PointNet++	26.837	81.379	18.23	36.223	117.824	27.541

Table 6. Mean squared reconstruction losses on the DFAUST testset with noise (left) or undersampled (right). We use 5% of the shape bounding box gaussian noise on the testset. We randomly sample 500 points from the test shapes surfaces. We recall that the network was trained on 1000 point clouds. We show the edge length reconstruction loss (EL), the rotation invariant reconstruction loss (PC) and the per triangle area difference

Table 6 (right) shows reconstruction results on simplified point clouds. We randomly sample 500 points from the test shapes surfaces. We recall that the network was trained on 1000 point clouds. We observe that our method is more robust to under-sampling. In particular, and contrary to other methods, the intrinsic properties remain competitive with the performance from Table 5.

We also demonstrate the generalization power across different datasets by showing in Figure 10 examples of reconstructions from SCAPE dataset [3]. While the simple PointNet AE, is still able to reconstruct the overall position of the tested human, the output has distortions near the hands (left) and the legs (right). Our method generates more natural meshes even though the dataset is completely unknown with an entirely different underlying mesh, different body

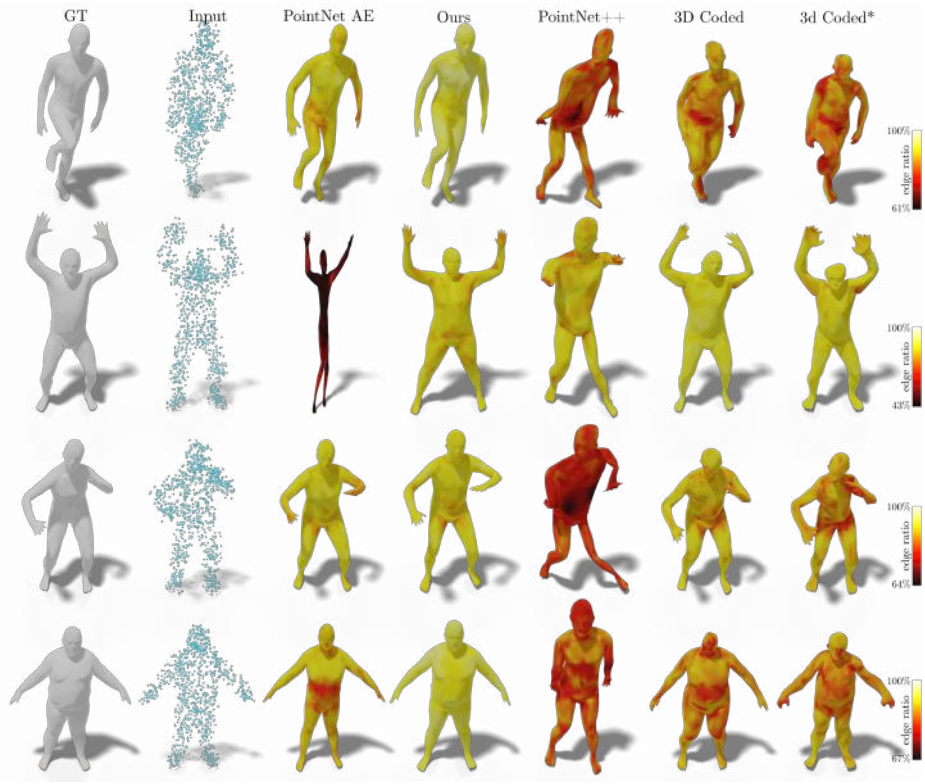

Fig. 8. Reconstructions from point clouds with 5% of the shape scale gaussian noise.

type and poses that are different to those seen at training. Note that we do not display the color coding as we do not have access to ground truth edge lengths.

D Ablation study

D.1 Architecture design

Importance of multiple separate networks We first test the utility of having separate networks, rather than training a single network with a combined loss. Specifically, in our study, we have observed that introducing intrinsic information directly during the training of the shape auto-encoder produces unrealistic results with significant artefacts. (Fig. 11) We train two point-cloud AE (auto-encoders) using: a combination of edge (L_e) and point coordinate (L_{rec}) losses and edge (L_e), point coordinate (L_{rec}) and linearity losses (L_{lin})

Effect of separate networks training In our experiments, we fix the weights of the shape AE and edge auto-encoder during the training of the mapping

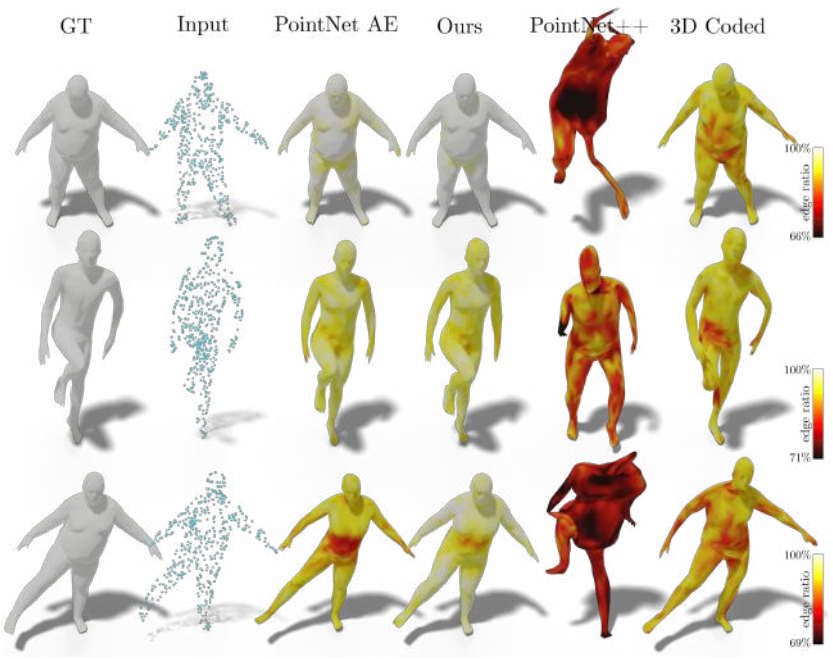

Fig. 9. We reconstruct a mesh from 500 points sub-sampled randomly from the ground truth mesh. We use a network pre-trained on inputs of size 1000 points.

networks. By doing so, we fix the latent space and generating capabilities of each network. We believe that if this constraint is not respected, the shape AE and edge auto-encoder can be indirectly trained for different losses and generate distortions in the generated shapes. Here, we train the mapping networks, edge auto-encoder and shape AE at the same time. To make the training easier, we use a pretrained shape AE and edge auto-encoder. As seen in Table 7, the reconstruction losses are better than before. However, the shape AE can produce non natural reconstructions during interpolations as shown in Figure 12. We believe that if the shape AE and edge auto-encoder network were not pretrained, the resulting reconstructed shapes would present even more distortions since the pretrained shape AE can already generate decent natural looking shapes on parts of the dataset.

Auto-encoder vs Variational auto-encoder During our study we compared the performances of our pipeline using either a PointNet AE or a PointNet VAE. The type of network did not result in significant differences. By instance the mean squared variance of the edge length for our architecture trained with a VAE is 0.2301 and 0.2311 when trained with a AE (respectively 0.3760 and 0.3510 for the simple VAE and AE without using our pipeline).

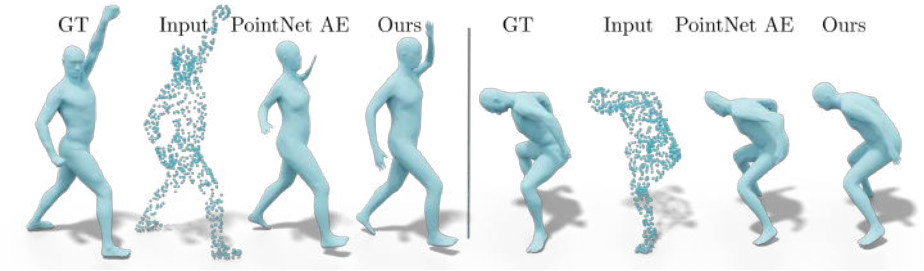

Fig. 10. Shape reconstruction from SCAPE. We reconstruct from 1k random points on the surface.

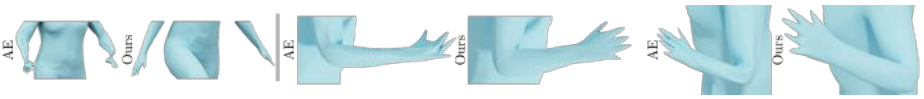

Fig. 11. Simple AE trained with L_e and L_{rec} (left) or L_e, L_{rec} and L_{lin} (right) produces artifacts during interpolation.

D.2 Choice of losses

Importance of cycle consistency loss. We train the mapping networks with direct reconstruction losses instead of cycle consistency losses as described in section 4.2 with L_{map1}, L_{map2}, L_{map3} :

$$\begin{aligned} L_{direct}(P, E_P) = \alpha \|\text{dec}_p(M_{EP}(\text{enc}_e(E_P))) - P\|^2 \\ + \beta \|el(\text{dec}_p(M_{EP}(\text{enc}_e(E_P)))) - E_P\|^2 \\ + \|\text{dec}_e(M_{PE}(\text{enc}_p(P))) - E_P\|^2 \end{aligned} \tag{12}$$

In Table 8, we observe that the quality of the map and the quality of the reconstructions are worse. In Figure 13 we show the cumulative distribution function of the edge length reconstruction loss on the testset. While most shapes seem to have reasonable edge reconstruction quality, outlier points make the

	EL (10^{-5})	PC (10^{-4})	area (10^{-8})
Ours	1.666	2.611	1.554
Ours sim. train.	1.027	1.464	1.027

Table 7. Mean squared reconstruction losses on the DFAUST testset. We present our main network and an alternative model where all three components are trained simultaneously. Edge length reconstruction loss (EL), Point cloud rotation invariant reconstruction loss (PC) and per triangle area difference (area).

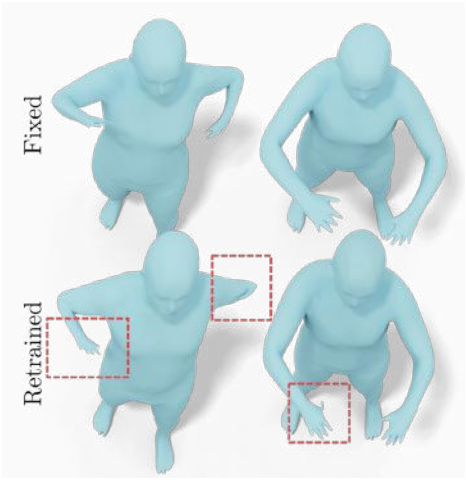

Fig. 12. Shape distortions are appearing during interpolation if the shape AE, edge auto-encoder and mapping networks are trained at the same time.

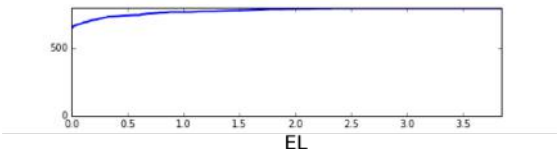

Fig. 13. Cumulative distribution function of edge reconstruction loss on the DFAUST testset for our network trained without cycle consistency with L_{direct}.

reconstruction loss explode. Since cycle consistency is not enforced, the network can map shapes onto outliers in the shape space that do not correspond to reasonable natural shapes.

Mapping losses In Table 9 we show an ablation study of the different losses combinations (described in section 4.2 of the main manuscript) used for training the mapping networks. The subscripts $1, 2, 3$ denote the use of L_{map1}, L_{map2}, L_{map3} respectively. We observe that when trained with L_{map2}, L_{map3}, so only intrinsic features, the model produces better intrinsic reconstruction performances to the expense of the extrinsic reconstruction loss. On the contrary, when trained with only L_{map1} and L_{map3} the network produces good point coordinate reconstruction but worse intrinsic reconstruction performances. To combine the benefits of the different losses, we choose to experiment with a model trained with the 3 losses.

	EL	PC	area
PointNet AE	$3.023 * 10^{-5}$	$2.120 * 10^{-4}$	$2.454 * 10^{-8}$
Ours	$1.641 * 10^{-5}$	$2.572 * 10^{-4}$	$1.562 * 10^{-8}$
Ours L_{direct}	0.1019	0.6289	$1.338 * 10^{-2}$

Table 8. Mean squared reconstruction losses on the DFAUST testset.

	EL (10^{-5})	PC (10^{-4})	area (10^{-8})
Ours $L_{2,3}$	1.595	14.816	1.490
Ours $L_{1,3}$	2.301	2.245	2.113
Ours $L_{1,2,3}$	1.641	2.572	1.562

Table 9. Ablation study on different mapping network losses. The subscripts $1, 2, 3$ refer to L_{map1}, L_{map2}, L_{map3} respectively. We show the mean squared reconstruction losses on DFAUST testset. Edge length reconstruction loss (EL), Point cloud coordinates reconstruction loss (PC) and per triangle area difference

Linearity regularization term in edge auto-encoder. We train a version of our network without the linearity regularization term L_{lin} described in Eq. (6) of the main manuscript for training the edge auto-encoder. As seen in Table 10, the interpolations in the latent space of the edge auto-encoder are smoother when the network is trained with the linearity term. In Table 11, we observe that this term is also related to smoother interpolations of shapes.

	EL
Edge AE	0.199
Edge AE no lin. reg.	1.777

Table 10. We report the mean squared variance of the edge length (EL) over the interpolation in the edge length AE latent space of 100 shape pairs.

	EL	area (10^{-4})	volume (10^{-4})
Ours	0.230	1.220	0.385
Ours no lin. reg.	0.245	1.361	0.430

Table 11. Interpolation losses for our network where the edge auto-encoder is trained with and without linearity regularization term. 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 from the DFAUST testset.

E Interpolation in unsupervised case

Our method can be adapted to an unsupervised context where the 1-1 correspondences are not provided during training. The training process can be described in 3 steps: We first train a point cloud auto-encoder that takes unordered point clouds and outputs an ordered point clouds where the order corresponds to given template T. Then we train the edge auto-encoder by using the output of

the shape auto-encoder as training data. Finally, we train the mapping networks as described in the main manuscript.

We first initialize the weights by pre-training the shape AE network to output a chosen template mesh using a variant of the reconstruction loss L_{rec} described in Eq. 4 of the main manuscript.

$$L_{recInit}(P) = \frac{1}{n}\sum_{i=1}^{n} \|T_i - \tilde{P}_i\|^2, \text{ where } \tilde{P} = \text{dec}_p\left(\text{enc}_p(P)\right). \tag{13}$$

Then we train the model using Chamfer Distance (CD) from Eq. (14) while encouraging the network to maintain the learned triangulation from step 1 by using regularization terms similar to those used in [21] described bellow.

$$CD(\tilde{P}, P) = \frac{1}{n}\sum_{p_i \in \tilde{P}} \min_{p_j \in P} \|p_i - p_j\|_2^2 + \frac{1}{n}\sum_{p_j \in P} \min_{p_i \in \tilde{P}} \|p_j - p_i\|_2^2 \tag{14}$$

$$L_e^{reg}(E_{\tilde{P}}) = \|E_{\tilde{P}} - E_T\|_2^2, \text{where } \tilde{P} = \text{dec}_p(\text{enc}_p(P)) \tag{15}$$

$$L_{lap}^{reg}(\tilde{P}) = \|L * (\tilde{P} - T)\|_2^2, \text{where } L \text{ is the graph laplacian} \tag{16}$$

We report numerical evaluation of the interpolations in Table 12. Note, that our method leads to improved shape features. In Figure 14, we observe that our method produces more realistic shapes, in particular it produces better arms and heads than PointNet AE.

	EL	area (10^{-4})	volume (10^{-5})
PointNet AE (unsupervised)	0.597	3.508	5.251
Ours (unsupervised)	**0.398**	**2.752**	**4.718**

Table 12. 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. We highlight, while both models produce worse results than their supervised equivalents, our method leads to better interpolations.

F Architecture details

We present the detailed architecture of the shape AE, edge length AE and mapping networks in Figure 15, 16, 17.

We implemented the presented architectures using Tensorflow and the Adam optimizer for training. Our complete implementation will be released upon acceptance.

Fig. 14. Interpolation between shapes when trained with no 1-1 correspondences at train time. Our method produces more realistic shapes.

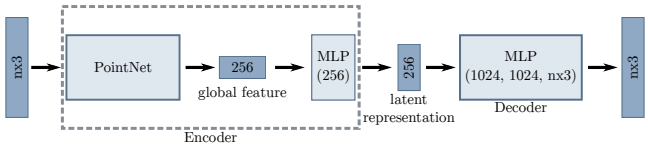

Fig. 15. Shape AE architecture.

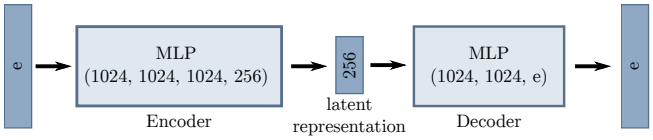

Fig. 16. Edge length AE architecture.

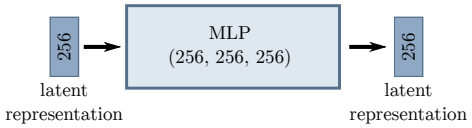

Fig. 17. Mapping networks architecture.