Computer Aided Geometric Design 71 (2019) 63–76

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Quantitative and flexible 3D shape dataset augmentation via latent space embedding and deformation learning

Jiarui Liu [a,1], Qing Xia [a,1], Shuai Li [a,b], Aimin Hao [a], Hong Qin [c,*]

[a] State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing 100191, China
[b] Beihang University Qingdao Research Institute, Qingdao 266000, China
[c] Department of Computer Science, Stony Brook University, Stony Brook 11794, USA

ARTICLE INFO

Article history:
Available online 5 April 2019

Keywords:
Gaussian process latent variable model
Data augmentation
Point clouds

ABSTRACT

Deep learning techniques for geometric processing have been gaining popularity in recent years, various deep models (i.e., deep learning methods based on neural networks) are developed with enhanced performance and functionality in conventional geometric tasks such as shape classification, segmentation, and recognition. Yet, deep models would rely on large datasets for the training and testing purpose, which are generally lacking as 3D shape geometry could not be easily acquired and/or reconstructed. In this paper, we propose a new 3D shape dataset augmentation method by learning the deformation between shapes in a highly reduced latent space while affording interactive control of shape generation. Specifically, we model each shape using a concise skeleton-based representation, and then we apply Gaussian Process Latent Variable Model (GPLVM) to embed all shape skeletons into a low-dimensional latent space, where new skeletons could be generated with diverse kinds of flexible control and/or quantitative guidance. A second network that learns the displacement between shapes can be employed to produce new 3D shape from newly-generated skeletons. Compared with popular computer vision techniques, our new generative method could overcome remaining challenges of 3D shape augmentation with new characteristics. Specifically, our new method is capable of transforming 3D shapes in a more liberal way, preserving their geometric properties at a semantic level, and creating new shape with ease and flexible control. Extensive experiments have exhibited the capability and flexibility of our new method in generating new shapes using only few samples. Our shape augmentation is an effective way to simultaneously improve the shape creation capability and the shape extrapolation accuracy, and it is also of immediate benefit to almost all deep learning tasks in geometric modeling and processing.

1. Introduction and motivation

Popular deep learning techniques have already exhibited their supremacy towards improving the state-of-the-art in speech recognition, visual object detection, image classification, and many other application domains LeCun et al. (2015). Nonetheless, deep models, for example deep neural networks, usually need extremely large training dataset to guarantee their prediction accuracy and generalization capability. In the field of computer vision, images are usually randomly ro-

* Corresponding author.
E-mail address: qin@cs.stonybrook.edu (H. Qin).
[1] These two authors contribute equally to this research.

https://doi.org/10.1016/j.cagd.2019.04.017
0167-8396/

tated, translated, and scaled to broaden the variance range of image datasets, thereby improving the performance of deep models Hernández-García and König (2018). These simple augmentations applied on images heavily rely on the pixel representation where image pixels representing colors or intensities are confined on a planar grid with canonical form. When it comes to 3D shapes, which are usually represented by point clouds or triangular meshes, simply translating or rotating shapes may not achieve the goal of increasing the variance of shapes because shapes before and after rigid translation and rotation are actually considered to be the same in terms of geometry invariance. Thus, data augmentation for 3D geometric shapes should comprise the augmentation of geometric information.

A feasible 3D shape augmentation algorithm should satisfy two conditions: (1) this augmentation should be able to discover as-much-as possible shape variances along certain directions within a relative small dataset; (2) new shapes synthesized by an augmentation algorithm should maintain semantic characteristics, and the generating process should as well be constrained under flexible and intuitive user-controlled guidance. One natural way to generate new shapes based on a few existing examples is shape interpolation Xu et al. (2006); Von-Tycowicz et al. (2015); Xia et al. (2015), which interpolate certain geometric properties of shapes and reconstruct new shapes by solving non-linear optimizations. However, these methods are usually neither versatile nor intuitive to implement and they tend to be time-consuming in general, which is not suitable for online augmentation during model training. Besides, point-to-point correspondences between shapes are necessarily enforced in most cases, which is impracticable when models are created individually wherein the order of vertices is not consistence across shapes. Another way is to synthesize shapes by recombining pre-segmented object parts. Huang et al. (2016) explore a support-induced structural organization of object parts, shapes are first segmented and represented as a structure based graph, then shapes are created by substructures combination. These methods may generate shapes in a novel way, while the quality of generated models can hardly be guaranteed. Deep learning methods, especially 3D convolutional neural networks Wu et al. (2016); Tatarchenko et al. (2017), generate reasonable results, but the simple transfer of 2D image generation framework to 3D domain will be severely restricted by memory limitation and thereby limit the resolution or equality of generated shapes. Moreover, these generating models are not easy to be controlled by user guidance in a flexible yet versatile way.

In this paper we devise a novel 3D data augmentation framework by learning the deformation between shapes in a highly reduced latent space while affording interactive and intuitive control during shape generation. Given a few shapes, a shape space spanned by these shapes can be parameterized to a low-dimensional manifold Campbell and Kautz (2014). Thus, we first adopt concise skeleton-based graphs to represent shapes and then use Gaussian Process Latent Variable Model (GPLVM) Lawrence (2005) to map skeleton-represented shapes to a shared low-dimensional embedding space. This latent space embedding in a lower dimension has two major advantages: (1) the skeleton-based representation affords input shapes without the need of point-to-point correspondences, and could significantly reduce the shape space dimension which allow users to explore with intuitive visual feedbacks; (2) GPLVM maps shapes from high-dimensional shape space into a much reduced latent space and confine reasonable shapes in a low space, where flexible and interactive controls could be applied in a most natural and intuitive manner. Then, we propose different kinds of navigation methods in the low-dimensional (it may be noted that oftentimes 2D is enough) embedding space to synthesize new graphs by simply choosing positions with low variance. Flexible user-guided navigation is enabled by moving the shape in latent space along any user-preferred direction which can be accurately characterized by a local principal component analysis and possible projection operator via an attraction force. Finally, a generative network is proposed to learn the deformation process between every two-model pair, and generate a new 3D shape from a newly produced skeleton graph. In particular, the salient contributions of this paper can be summarized as follows:

- We pioneer a novel 3D shape dataset augmentation scheme by learning the deformation process in a dimensionality reduced latent space, which gives rise to interactive and flexible shape generation control.
- We propose different kinds of shape navigation strategies in dimensionality reduced latent space, which affords flexible and quantitative user guidance for generation of new shapes in the original high-dimensional shape space.
- We propose a novel deformation-learning network which transforms a source 3D shape to a target shape under the guidance of a skeleton, which can produce excellent results even with small datasets.
- The 3D data augmentation framework is also of immediate benefit to other deep learning tasks in geometric modeling and processing.

2. Related works

Closely relevant to the central theme, we now briefly review previous approaches and their related applications in two categories: manifold learning and 3D shape generation, and geometric deep learning.

2.1. Manifold learning and 3D shape generation

The core of manifold learning techniques is to model data as low-dimensional manifolds. The Gaussian Process Latent Variable Model proposed by Lawrence (2005) has proven to be powerful in manifold learning and data prediction. Campbell and Kautz (2014) build a generative manifold of standard fonts by representing typefaces as closed curves. They take a collection of existing font files and create a low dimensional space such that every location in this space generates a

novel typeface through the interpolation and extrapolation of these fonts. Turmukhambetov et al. (2015) use the learned manifold to provide interactive visual feedbacks. They learn a low-dimensional manifold from the data that models the joint configuration of masses and the contour shape of objects, which are presented by Stokes parameters and elliptical Fourier coefficients. Manifold learning methods excel in modeling moving styles, too. Bailey et al. (2016) map high-dimensional rig control parameters to a three-dimensional latent space. They use a particle model to move within one of these latent spaces to automatically animate interactive characters, as well as bridges to link each pose in one latent space that is similar to a pose in another space. Levine et al. (2012) present a technique that animates characters performing user-specified tasks. A low-dimensional space learned from the example motions is used to continuously control the character's pose to accomplish the desired task. Similarly, Yin et al. (2018b) generate closely interacting 3D pose-pairs from a set of video frames by sampling over the space of close interactions. The sampling process starts with one or more manually designed seed 3D skeletal pose-pairs, and the seed set is augmented via a Markov Chain Monte Carlo (MCMC). Other existing shape generation methods either interpolate between existing shapes or exchange their parts. Alhashim et al. (2014) define blend operations on a spatio-structural graph, fundamental topological operations including split and merge are realized by allowing one-to-many correspondences between the source and the target. Han et al. (2015) create new shapes by recombining existent styles and contents. They perform style and content separation to analyze shapes by clustering their multi-scale corresponding patches and create novel shapes by style transfer. Huang et al. (2016) generate new shapes by applying the derived high-level substructures to part-based shape reshuffling between models. A bottom-up approach is presented to identify a set of basic support substructures and combine them to form complicated substructures. Ranaweera et al. (2017) enable novice users to work together to generate creative 3D shapes by allocating distinct parts of a shape to multiple players who model the assigned parts in a sequence. Inspired by GP-LVM's powerful ability of non-linear dimension reduction, we apply it in our framework to embed all shapes into a latent space where flexible user control of shape generation can be easily defined.

2.2. Geometric deep learning

Researches in deep neural networks have advanced significantly in recent years. The majority of extant works, namely Wu et al. (2016), generate 3D objects via 3D Generative-Adversarial network by leveraging volumetric convolution. To obtain shapes preferred by users, a joint embedding space can be learned by autoencoder approaches. Girdhar et al. (2016) achieve this by building an architecture that has two major components, an autoencoder that ensures the representation is generative, a convolutional network to predict shapes from images. Chen et al. (2018) learn implicit cross-modal connections between shapes and texts by using association and metric learning approaches, and produce a joint representation for texts and the properties of 3D shapes. Hu et al. (2018) expand the volumetric convolutional method to a framework, which recognize the functionality of a single voxelized 3D object and synthesizes segmented surrounds. Dai et al. (2016) generate coarse 3D geometries and add details for those low-resolution predictions by finding neighbors in high resolution 3D geometric shape database. These methods are limited in voxel presentation and suffers from huge calculations. Li et al. (2017) generate new structures by introducing a novel network architecture which recursively maps a flat, unlabeled, arbitrary part layout to a compact code using autoencoders, and maps a code back to a full hierarchy using an associated decoder. This method yields generative models of plausible structures, but the curve information of each part is lost, which caused the ambiguousness of results. Given a picture, point clouds can be generated by estimating the depth for visible parts and hallucinating the rest as Fan et al. (2016) does. Qi et al. (2016) come up with a novel type of neural network that directly consumes point clouds by using symmetric functions, and Qi et al. (2017) extend the network to a hierarchical neural network that applies PointNet recursively by partitioning the input point set into local small points. More plausible results have come up after the propose of PointNet++. Sung et al. (2017) synthetic novel models by jointly training embedding and retrieval networks, where the first maps parts to a low-dimensional space and the second maps input parts to appropriate complements. Lastly, a placement network is trained to put the complements on right places. Yin et al. (2018a) build a bidirectional net work to learn the shape transformation process between two domains, while Achlioptas et al. (2017) perform a thorough study of different generative networks operating on point clouds, including GANs and AEs, especially GANs trained in the latent space learned by AEs. The limitation of those methods is that they can neither fully control the generation process nor generate shapes in a continues way. In this paper, we will combine traditional manifold learning method with neural network to achieve flexible and controllable 3D shape generation or namely augmentation.

3. Latent space embedding and navigation

Fig. 1 shows the entire pipeline of our 3D shape augmentation framework. We firstly extract the skeleton for each shape represented by point cloud to produce a consistent graph-based representation wherein shapes are modeled with the same topology. Then we embed these graphs into a low-dimensional latent space using a latent variable model, where one can explore to generate new graphs by simply choosing positions with low variance. In this section, we will introduce the details of skeleton-based shape representation, latent space embedding and flexible navigation.

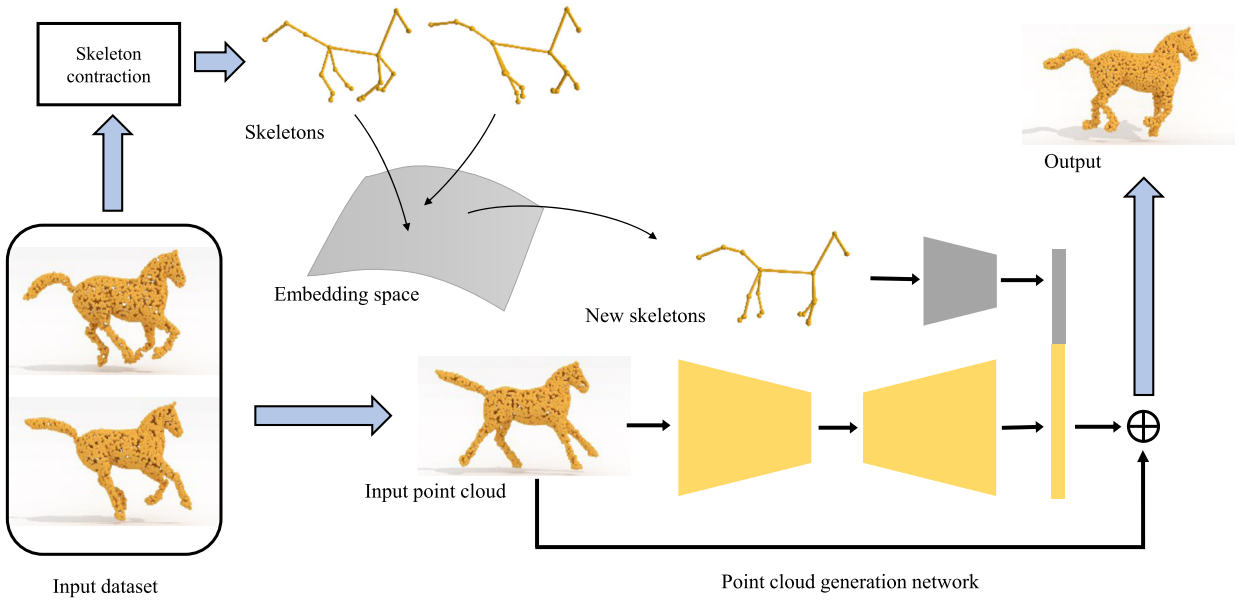


Fig. 1. Our shape augmentation framework. We firstly contract the skeleton for each point cloud in our dataset, then project all the skeletons onto a shared embedding space and explore new skeletons in this space. Finally, we use a deformation network to transform a point cloud we have to a new shape whose pose is represented by the new generated skeleton. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

3.1. Skeleton-based shape representation

As point clouds can hardly be aligned to each other, projecting high dimensional point clouds into a low-dimensional embedding space seems intractable. Our key observation is that humans have the ability to distinguish between shapes according to their skeletons, because skeletons contain the majority information about the differences between shapes of the same category. Based on this assumption, we consider the skeleton as key information of the shape and extract the skeleton for each point cloud as a concise representation of its interior architectural feature. For each shape, we compute its skeleton via local Delaunay triangulation and topological thinning with Laplace matrix, as is stated by Cao et al. (2010). Solving contraction problems in a global way for models with uneven thickness may cause strange results, it struggles between contracting skeletons of thick parts and preserving curves of thin parts. We expand the method to a dynamic local Principal Component Analysis (PCA) version. In every step, we extract its k nearest neighbors and perform a local PCA on them. Then we determine that a point is already lying on a curve if their projection on one direction covers the majority of information. Given the sorted local eigenvalue written as U, we keep the points that already satisfy our demand stable by updating their Laplace matrix as

$$M_{ij} = \begin{cases} 0 & \frac{\max_{u_k \in U} u_k}{\sum_{i=k}^{3} u_k} > \alpha \\ v_{ij} - w_{ij} & \text{otherwise} \end{cases}, \tag{1}$$

where v_{ij} is the value of the row i and column j of the degree matrix, and w_{ij} is the corresponding value of adjacency matrix, α is an manually defined threshold. As shown in the left side of Fig. 2, the improved method can handle with our model in a better way, the details in the dotted circle show how the octopus' legs are better represented by curves contracted by the improved method. After contraction, a skeleton graph is constructed by farthest-point sampling method as Cao et al. (2010) does.

The skeletons obtained above still need to be further normalized, for skeletons should be aligned into same length to be projected onto a low-dimensional latent space, and redundant points will affect the calculation of the distribution. To this end, we cut the graph into branches by identifying nodes whose degree is greater than 2 as bifurcations. Then we cluster these branches according to their positions as well as their lengths and directions. For each branch with one one-degree end point, the position is defined as the position of the one-degree point, and for branches with two two-degree end points, the position is defined as the middle position of the branch. The direction is determined by the first component of local PCA. After that, branches of the same class are uniformly sampled with the same number of points, where the number is chosen empirically in our experiments. We show the representation computing process in Fig. 2. For parts with stick bones, we can further reduce the number of points by fitting bones to each part and optimizing their joints. After that, we reorganize the

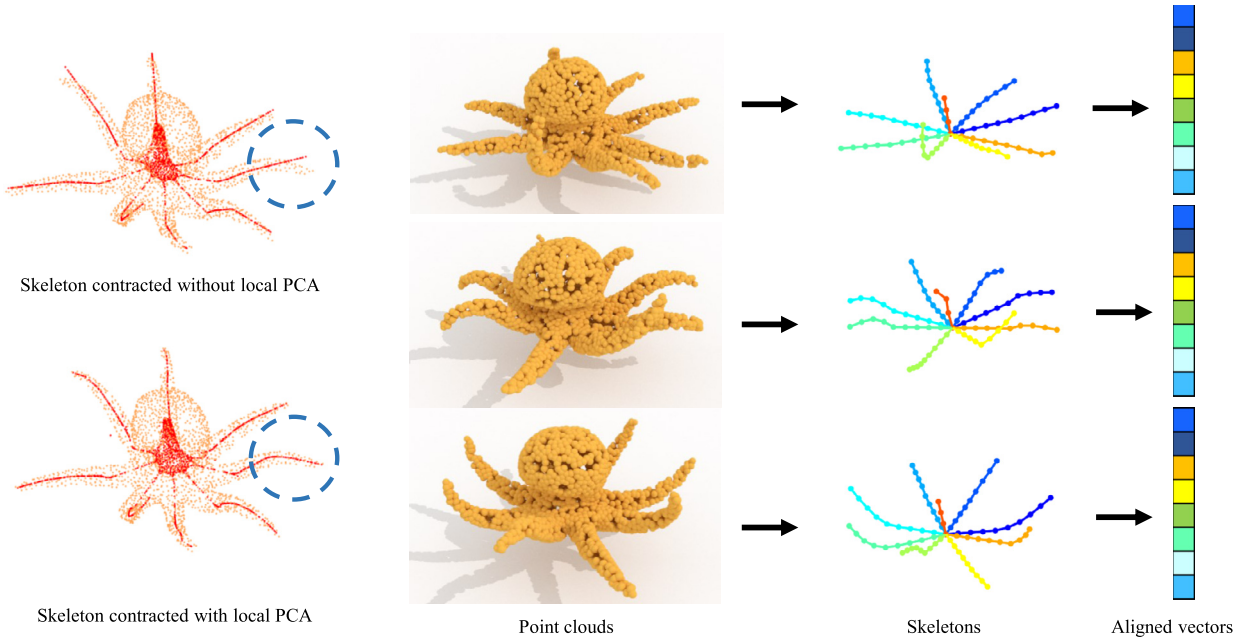


Fig. 2. The left two octopus show skeleton contraction results with and without local PCA. The skeleton contracted with PCA shown in second row preserves the curves of legs better. The right part shows our vector alignment procedure which maps point clouds to consistent vectors with same length. Segmentation of skeletons and the organization of parts are indicated by different colors.

skeletons into vectors of the same length. Finally, we got 19 key points for each horse, 84 key points for each octopus, and 13 key points for each human shape.

3.2. Latent space embedding

For now, we have modeled the shapes with vectors of same length, then we will utilize the leverage of Gaussian Process Latent Variable Model to learn a continuous, generative latent space. As stated by Rasmussen and Williams (2005), GP-LVM assumes that the mapping procedure from embedding low dimensional space to high dimensional space is a Gaussian process, which means, inputs and outputs appear in a continuous domain, and any discretely extracted set under the function is normally distributed. Given aligned D dimensional vectors organized in matrix $Y \in R^{N\times D}$, our goal is to find the associated latent variables $X \in R^{N\times Q}$, where N is the number of observations and Q is the dimensionality of latent variables, especially, $Q \ll D$. The GP-LVM defines a generative mapping from the latent pace to observation space that is governed by Gaussian processes. The vectors of different outputs are drawn independently from the same Gaussian process prior which is evaluated at the inputs X, the likelihood function is written as

$$p(Y \mid X) = \prod_{n=1}^{N} p(\mathbf{y}_n \mid X), \tag{2}$$

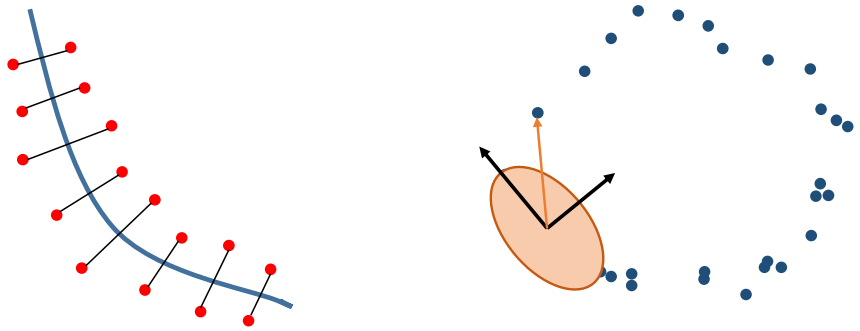
where $\mathbf{y}_n$ is the n-th vector in Y, and the probability for each $\mathbf{y}_n$ is written as

$$p(\mathbf{y}_n \mid X) = \mathcal{N}(\mathbf{y}_n \mid 0, C(X, X \mid \theta)), \tag{3}$$

where $C(X, X \mid \theta)$ is the covariance function that maps X to Y and θ is the hyper-parameter. We can assign every $\mathbf{x}_n \in X$ a prior density given by the standard normal distribution, $\mathcal{N}(0, 1)$. The covariance function is organized in a matrix where the value of row i and column j defines the covariance between two input vectors, which is calculated by a kernel function. We choose the Radial Basis Function (RBF) kernel as the kernel function. For every two vectors in embedding space namely $\mathbf{x}_i$ and $\mathbf{x}_j$, the i-th row and j-th column of the covariance matrix is given by

$$c(\mathbf{x}_i, \mathbf{x}_j \mid \theta) = \sigma_{rbf}^2 exp(-\frac{1}{2}\phi \left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2), \tag{4}$$

where σ_{rbf} and ϕ are hyper-parameters, written as $\theta = [\sigma_{rbf}, \phi]$. Substituting Eq. (3) and Eq. (4) into the likelihood function of Eq. (2), we can obtain a complex, non-linear objective function to be maximized. Then we solve the optimization with methods introduced in Titsias and Lawrence (2010) and initialize the values with a linear PCA reduction.



(a) Principal curve (b) Local direction

Fig. 3. Schematic diagram of the principal curve fitted in reduced latent space (left) and the local PCA guided exploring directions (right). The red and blue dots in (a) represent our embedding data, while black lines represent the self-consistent characteristic of principal curve. Arrows in (b) are local coordinates (black) and the direction a user may actually want to go(orange).

3.3. Automatic exploration based on principle curve

In order to discover all possible shapes, we need ways to automatically explore and generate new shapes. We believe the features obtained from models lie on a shared manifold, so discovering the embedding manifold means finding all the shapes possible. Principle curves (Hastie and Stuetzle (1989)) and principle surfaces are believed to be one-dimensional and two-dimensional form of manifold. We perform a principal curve extraction to obtain the underlying manifold, here we use 2D latent space for illustration. Given embedding data $X \in R^{N\times Q}$, principal curves are curves that self-consistent and pass the middle of the data in a smooth way. That is to say, the positions of data are distributed on both sides of the curve symmetrically. Given a one-dimensional smooth curve parameterized over R^1 written as $f(\lambda)$, we seek for the curve that minimizes the cost

$$E_c = \sum_{i=1}^{N} \|\mathbf{x_i} - f(\lambda_i)\|^2, \tag{5}$$

where λ_i is the index of x_i, which is sorted by the distance from start point of the curve to the projection point of x_i, which we call the projection index. Since one point might be projected on several positions on the curve, we define λ_i as the largest index of the values of the smallest projection distance in all possible projection value τ. The definition is given as

$$\lambda_i = argmax\{\lambda : \|\mathbf{x} - f(\lambda)\| = inf_\tau \|\mathbf{x} - f(\tau)\|\}. \tag{6}$$

Theoretically, as shown in Fig. 3(a), a self-consistent curve means that for every projection index λ_i, the curve lies exactly on the median of points whose projection index is λ_i. However, there is usually only one point for a projection index, so we find the curve that lies on the middle of the points of their projection index neighbors. In practice, we represent $f(x)$ as a collection of line segments, and initialize it with the first linear principal component of all data. The curve is calculated in an iterative way, in which we calculate the projection index and update $f(\lambda_i)$ as the middle position of the points whose projection index are close to λ_i and repeat this step until the algorithm converges.

3.4. User-guided navigation based on local PCA

In addition to the automatic exploration, we also provide user-guided navigation method to augment the dataset in an interactive way. Assuming there is a point who always tends to discover new shapes, a main direction is specified for user to decide whether to find shapes novel or generate shapes that is similar to shapes in dataset. The local coordinate system is defined by preforming local principal component analysis and the default direction is obtained by defining forces around them. As shown in the right side of Fig. 3, given the current position in latent space, we find the point with its nearest neighbors and perform a principal component analysis on them. The main direction is defined by its first principal component, while the second principal component defines another. Always going through the main direction will produce shapes similar to point clouds in dataset, while going through the direction vertical to the main direction will go away from the manifold and generate shapes with novel features. Users can also easily define the advance direction simply by defining the degree of deviation from the main direction.

We provide default advancing directions by defining forces on the current point in an embedding space. When current point is moving too far from the manifold, it should be advised to go towards original points, while when the point is too close to the path it has been explored, it should be advised to leave the path and find something novel. Forces acting on the point includes attractive forces from original points and repulsive forces from points on the paths. The resultant force for a point of position $\hat{\mathbf{x}}$ is defined as

$$F_{attr} = -\sum_{\mathbf{x_i} \in N_{nei}(\hat{\mathbf{x}}, X)} \frac{1}{\sqrt{2\pi}\gamma_1} exp(-\frac{\left\|\hat{\mathbf{x}} - \mathbf{x_i}\right\|_2^2}{2\beta_1^2}) + \sum_{p_i \in N_{nei}(\hat{\mathbf{x}}, P)} \frac{1}{\sqrt{2\pi}\gamma_2} exp(-\frac{\left\|\hat{\mathbf{x}} - p_i\right\|_2^2}{2\beta_2^2}), \tag{7}$$

where P is the path consisting of positions it has passed. $N_{nei}(\hat{\mathbf{x}}, X)$ and $N_{nei}(\hat{\mathbf{x}}, P)$ are neighbors in original points and neighbor points on the path, γ_1, γ_2, β_1 and β_2 are hyper-parameters. Assuming a point is subjected to a constant force F_{main} along the main direction, the recommended direction is

$$F_{dir} = F_{main} + \epsilon F_{attr}, \tag{8}$$

where ϵ is an empirical parameter used to define how much a point is affected by its neighbors.

4. Geometry displacement learning for shape synthesis

For now, we have embedded the shapes into a low-dimensional latent space and provided automatic and manually controlled navigation methods. In this section, we will detail how to generate new shapes using the trained GP-LVM and a shape generating network that learns the displacement between any two point clouds during transformation.

4.1. Skeleton construction

Generating new skeletons from latent space with our trained GP-LVM model is straightforward. We denote trained embedding vectors as X^* and super parameters as θ^*. Consider $\hat{\mathbf{x}}$ as our new point in embedding space, we get the model with $\hat{\mathbf{x}}$ as

$$\begin{bmatrix} Y \\ \hat{\mathbf{y}} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} C(X^*, X^*) & C(X^*, \hat{\mathbf{x}}) \\ C(\hat{\mathbf{x}}, X^*) & C(\hat{\mathbf{x}}, \hat{\mathbf{x}}) \end{bmatrix}\right). \tag{9}$$

Then we got the function of output $\hat{\mathbf{y}}$ as

$$p(\hat{\mathbf{y}} \mid X^*, \theta^*) = \mathcal{N}(C(\hat{\mathbf{x}}, X^* \mid \theta^*) C(X^*, X^* \mid \theta^*)^{-1} Y, \Sigma), \tag{10}$$

with the covariance Σ as

$$\Sigma = C(\hat{\mathbf{x}}, \hat{\mathbf{x}}) - C(\hat{\mathbf{x}}, X^* \mid \theta^*) C(X^*, X^* \mid \theta^*)^{-1} C(\hat{\mathbf{x}}, X^* \mid \theta^*). \tag{11}$$

We treat the mean value as our predicted result and the covariance Σ as a confidence measurement for the prediction, a lower value usually means a more reasonable result, which is shown in Section 5.

4.2. 3D point clouds generation for new shapes

Now we introduce our generation network used to generate new point clouds from skeletons. While deep learning methods struggle learning proper weights with few training data, in traditional shape generation methods, new meshes are usually generated by combining several parts extracted from neighbors under certain metrics, wherein the point-to-point correspondences are built and source parts are deformed and transformed according to features of the target shapes. Is there any way that combines the advantages of both, to generate shapes from a small dataset without part-to-part, or point-to-point correspondence? We address this problem by changing the goal from learning the mapping between skeletons and point clouds to learning the deformation process between every two models with a skeleton as guidance. The change of learning target expands the training data. Given N point clouds and their corresponding skeletons, traditional one-to-one learning networks have N pairs of training data, while, by learning the deformation process between any two model, we use each data N time in one epoch and obtain $N \times N$ pairs of data, which is usually sufficient for a network to learn a meaningful model.

The whole network structure is shown in Fig. 4. Our network consists of three components, feature extraction, posture recognition and displacement learning. We utilize the strong power of PointNet Qi et al. (2016) and PointNet++ Qi et al. (2017) in feature extraction part and posture recognition part. In feature extraction part, the network analyzes what it is for each part and learns a feature for each point. This part takes whole point cloud models as input and passes them into a multi-layer network. At each layer l, we sample and group the inputs locally, and pass the features of same group into a shared multi-layer perceptron. The learned features in each group is passed into a symmetric function that maps a set of inputs into a vector in layer $l+1$. By interpolating feature vectors according to their coordinates in layer $l-1$, global features are propagated back to each point. We believe that the learned features contain the local structural information of the geometry. In the posture recognition part, key points are passed into another network which transforms the skeletons into features that a network can understand. In displacement learning part, skeleton features and point cloud features are concatenated and passed into three full-connected layers. Those layers analyze and integrate the information of two parts, and end up with a vector for each point. The deformed point clouds maintain geometric details of input point clouds and have the same posture with input skeletons.

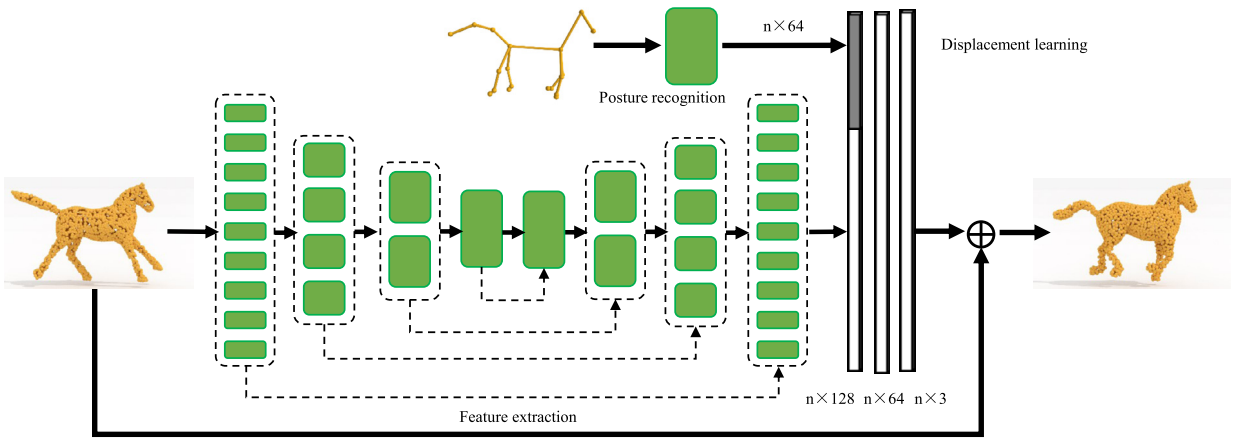


Fig. 4. Our shape generation network. Our network takes an existing point cloud and a skeleton as input, and outputs a new point cloud with the pose indicted by the input skeleton. Each green rectangle represents a PointNet Qi et al. (2016) block, and the whole feature extraction part is in the form of PointNet++ Qi et al. (2017).

We organize each training data as a tuple (I, T, S). Given input point cloud I and learned displacement T, we obtain the deformed output model as $\hat{S} = I + T$. To measure the distances between deformed shape $\hat{S}$ and target shape S, we define loss functions between two point clouds. To obtain a shape that is similar to target, we use the Chamfer's distance as our loss function

$$L_{chamfer}(\hat{S}, S) = \sum_{p\in\hat{S}} min_{q\in S} \|p - q\|_2^2 + \sum_{q\in S} min_{p\in\hat{S}} \|p - q\|_2^2 . \tag{12}$$

In order to generate point clouds with smooth surface, we use the density loss defined in Yin et al. (2018a)

$$L_{density}(\hat{S}, S) = \frac{1}{k} \sum_{p\in S} \sum_{i=1}^{k} \left| d(p, N_i[S, p]) - d(p, N_i[\hat{S}, p]) \right|, \tag{13}$$

where $N_i[S, p]$ is the nearest neighbors in point cloud S of point p, and k is the number of neighbors. In traditional deformation methods, preservation of local topology is often added as a constraint to the optimal target equations. We add it as a transformation loss which measures the change of the distance between every point and their neighbors before and after the transformation. This constraint guides the point to a more correct position during the transformation instead of simply putting it to the nearest position in the target, and therefore reduces outliers. The transformation is defined as

$$L_{trans}(\hat{S}, I) = \sum_{p\in\hat{S}, p'\in I} \left| d(p, R_r[\hat{S}, p]) - d(p', R'_r[I, p']) \right|, \tag{14}$$

where p' is the point that is transformed to p during the deformation, $R'_r[I, p']$ is the neighbors on input shape I whose distance to point p' is smaller than a searching radius r, which is usually set as 0.02, and $R_r[\hat{S}, p]$ is the corresponding points transformed from $R'_r[I, p']$ in shape $\hat{S}$. The entire loss function is as follows

$$L_{loss}(\hat{S}, S) = L_{chamfer}(\hat{S}, S) + \delta L_{density}(\hat{S}, S) + \eta L_{trans}(\hat{S}, I), \tag{15}$$

where δ and η are hyper-parameters defined for training, which are usually set as 1.0 and 2.0 respectively. Theoretically, since the deformation process is trained on every two point clouds, we can use any shape in dataset as a template model to provide details. A KNN algorithm can also be performed according to the common sense that deformations between similar shapes are likely to obtain better results. Since the dataset is small, the searching process won't be a waste of time. Considerable results are shown in Section 5.

5. Experimental results

5.1. Data and parameters

We demonstrate our experimental results with three categories point clouds. For each category, we collect 3D shapes from various websites, and transform them into point clouds by uniformly sampling on their surface using poisson disk sampling method. We have 47 horse models, 84 octopus models and 50 human models with 2048 points in each model. Shapes are normalized and moved to coordinate center. Each of the models is zoomed and rotated to make them oriented in same direction. During navigation, we take new points' nearest neighbor as input point cloud.

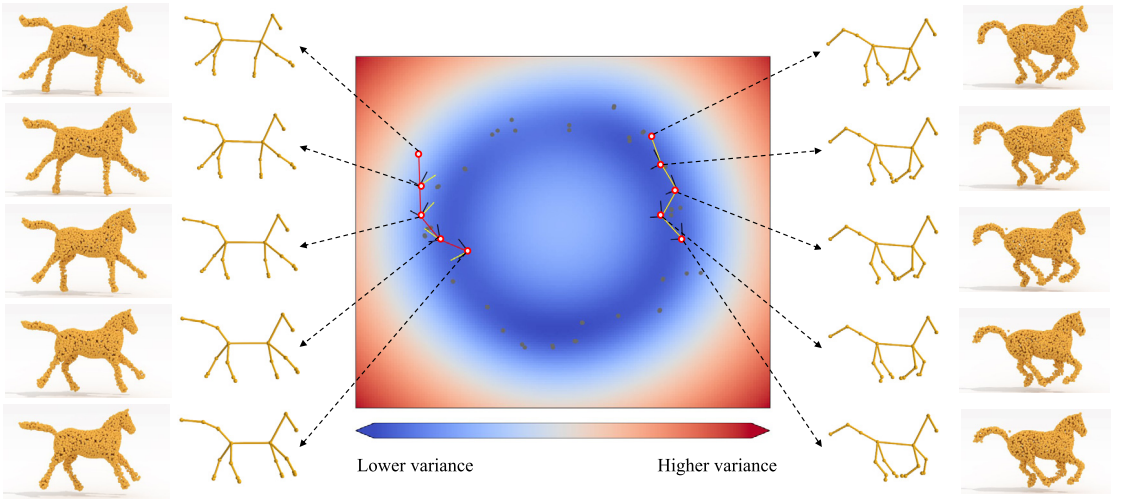


Fig. 5. 2D manifold of skeletons learned for exploration and generation. The navigation path is shown in red while the local coordinate of each point is shown in black lines. The yellow line is the direction recommended to go. The location of the original training skeletons are shown in gray, and the color map from blue to red indicate the variance to be a reasonable 3D shape, where blue means high variance and red means low one. Skeletons and corresponding shapes generated from paths are shown on the side.

5.2. Shape embedding and shape navigation

Fig. 5 shows an example of the manifold learnt with horse skeletons. The position of original postures in our embedding space are shown as gray dots. We use blue and red color to indicate the value of the variance, blue means a lower variance and a reasonable result. Variance around the original postures are lower, which is consistent with the perception that a model is more reasonable if looks similar to the existing model.

We show the generation ability with our navigation methods. Navigation process can start from everywhere, user may either choose a particular direction or navigate in default mode. In user-guided mode, user can choose an angle to define the degree of deviation from the main direction. In auto mode, system will automatically choose the direction that can explore interesting things. We show our user defined navigation method in Fig. 5 with horse dataset. We draw two navigation paths, in the left path, we use the direction of the angle of 30 degrees from the main direction, in the right path, we use direction recommended automatically by algorithms mentioned in Section 3. The navigation path is drawn in red. For each point, we draw local coordinate in black lines and the recommended in yellow. In default mode, a point will always explore new shapes around origin points. For a more intuitive representation of the results, we put the new skeletons into generation network for final point clouds. New skeletons and generated horses are shown around the embedding space.

We show our principle curves in Fig. 6 with 3 datasets. We sample 8 points in each embedding space and display the skeletons on the right side. Skeletons generated from points sampled along the curve results in representative results, which means the curve is the manifold we are looking for. Sampling around the principle curve can automatically augment the dataset and generate shapes similar to what we are looking for.

5.3. Shape generation

In this subsection, we demonstrate our generated results with some quantitative indicators. Fig. 7 shows some results created by our network. For each skeleton, we show results of different input shapes as guidance and get similar outputs even when the input shapes are quite different from each other, which means the network not only transform the shape in a simple way, but learned the semantic meaning for each point instead. Note that the generated shapes maintained detail information as the input gives, which will be extremely hard for those methods who generate details using skeletons only.

We also compare our methods with P2P-Net Yin et al. (2018a). We sample 2048 points for each skeleton, and feed the skeletons and their corresponding point clouds to the P2P-NET network which works by learning the displacement between skeletons and point clouds. As demonstrated in Fig. 8, although P2P-NET is capable of generating coarse shapes, the points on the surface are a bit messy, while our method gives good results with fine details and smooth surface in the same situation. We believe the undesired artifacts of P2P-NET are caused by the limitation of small datasets with which it can hardly learn the generation process, while our generation method learns the displacement between each two shapes with a skeleton as guidance where the generated new shapes naturally carry details of input shapes.

We evaluate the quantitative generation performance of our method and P2P-NET on the three datasets used above. We remind that all the training and test point clouds are normalized such that the diagonal lengths of their bounding boxes are equal to 1. We list several metrics as below and demonstrate our results measured with them in Table 1.

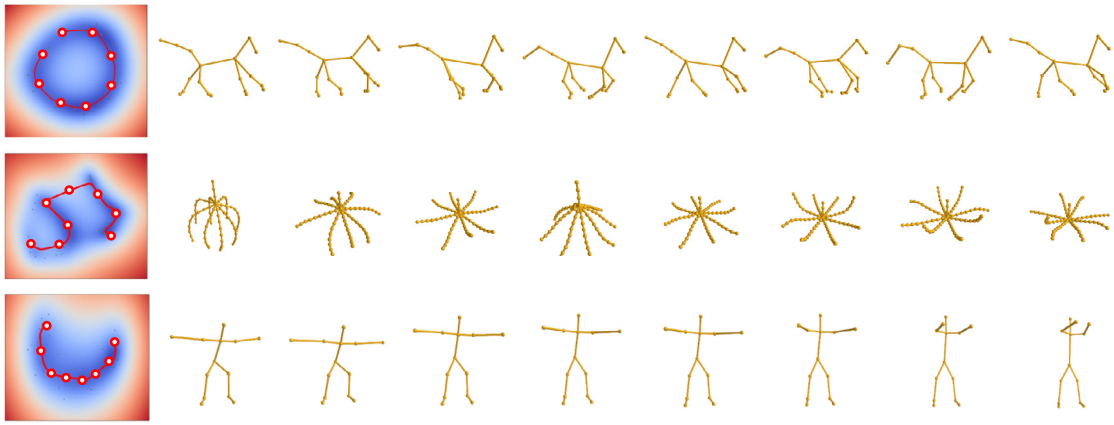


Fig. 6. Principle curves (red) in 2D manifold. We show sample points on curve as white dots and list the skeletons in the order they appear from the left end of the curve to the right end.

Table 1
Quantitative evaluation of our network with different metrics comparing to P2P-NET Yin et al. (2018a). Values in bold indicate better results.

Dataset	Separate rate		Coverage		Earth Mover's distance	
	P2P-NET	Ours	P2P-NET	Ours	P2P-NET	Ours
Horse	0.19%	**0.07%**	45.38%	**69.01%**	0.0573	**0.0201**
Octopus	0.95%	**0.13%**	26.60%	**59.17%**	0.0961	**0.0231**
Human	1.79%	**0.11%**	36.75%	**66.99%**	0.0870	**0.0230**

Point separation rate. For each point in a predicted point cloud, we search its closest point in its neighbors and regard the distance from the nearest point as its distance to the surface of the point cloud. If the distance between point p and its closest point is larger than 0.02, we consider p as a separated point. We define the separation rate of predicted points as the percentage of separated points among all points in predicted point cloud. We report the mean separation rate of all test examples in Table 1. Our lower separation rate indicates that our method generates a smoother surface.

Coverage. Defining matched point as the closest neighbor in target point clouds, we estimate the coverage indicator for each point p as the fraction of the points in target point cloud that is matched to points in predicted point cloud. Higher coverage score means the predicted point cloud can be represented by target point cloud in a better way. Table 1 demonstrates that compared to P2P-NET, the information contained in target shapes is better represented by our method.

Earth mover's distance. The Earth Mover's distance (EMD) Rubner et al. (2000) is a widely used solution of a transportation problem which attempts to transform one set to the other. The EMD between two point clouds $S_1 \subseteq R^3$ and $S_2 \subseteq R^3$ is defined as

$$d_{EMD}(S_1, S_2) = min_{\Phi: S_1 \to S_2} \sum_{p \in S_1} (\|p - \Phi(p)\|_2), \tag{16}$$

where Φ is a bijection map between points in two point clouds. The EMD indicates the fidelity of our predicted model, we demonstrate that compared to P2P-NET, our method generates shapes with stable and better fidelity.

5.4. Data augmentation for specific tasks

In this section, we show how our method can be used to improve specific training tasks under the framework we introduced above and demonstrate our augmentation improvement with quantitative indicators. By design, our method augments a small point cloud dataset without requesting any correspondence information, which is convenient and sufficient for tasks such as shape classification, wherein the label of one shape is not related to the order of shape points. Thus, to augment dataset for classification tasks, user can simply follow the method described above to generate new shapes of the same category. Note that generated shapes are not strictly aligned with input shapes. For example, a point on the mouth of a human shape may not be transformed to the mouth of the target shape. This inconsistency of points between existing and generated shapes is harmful for tasks such as shape segmentation and point-to-point correspondence, wherein data augmentation is required to maintain contextual information of the dataset (for example, augmenting datasets with segmentation labels should generate new shapes with consistent segmentation labels). However, the tags provided in the training dataset for segmentation or correspondence can be used to improve our augmentation method simply by defining new constraints or losses.

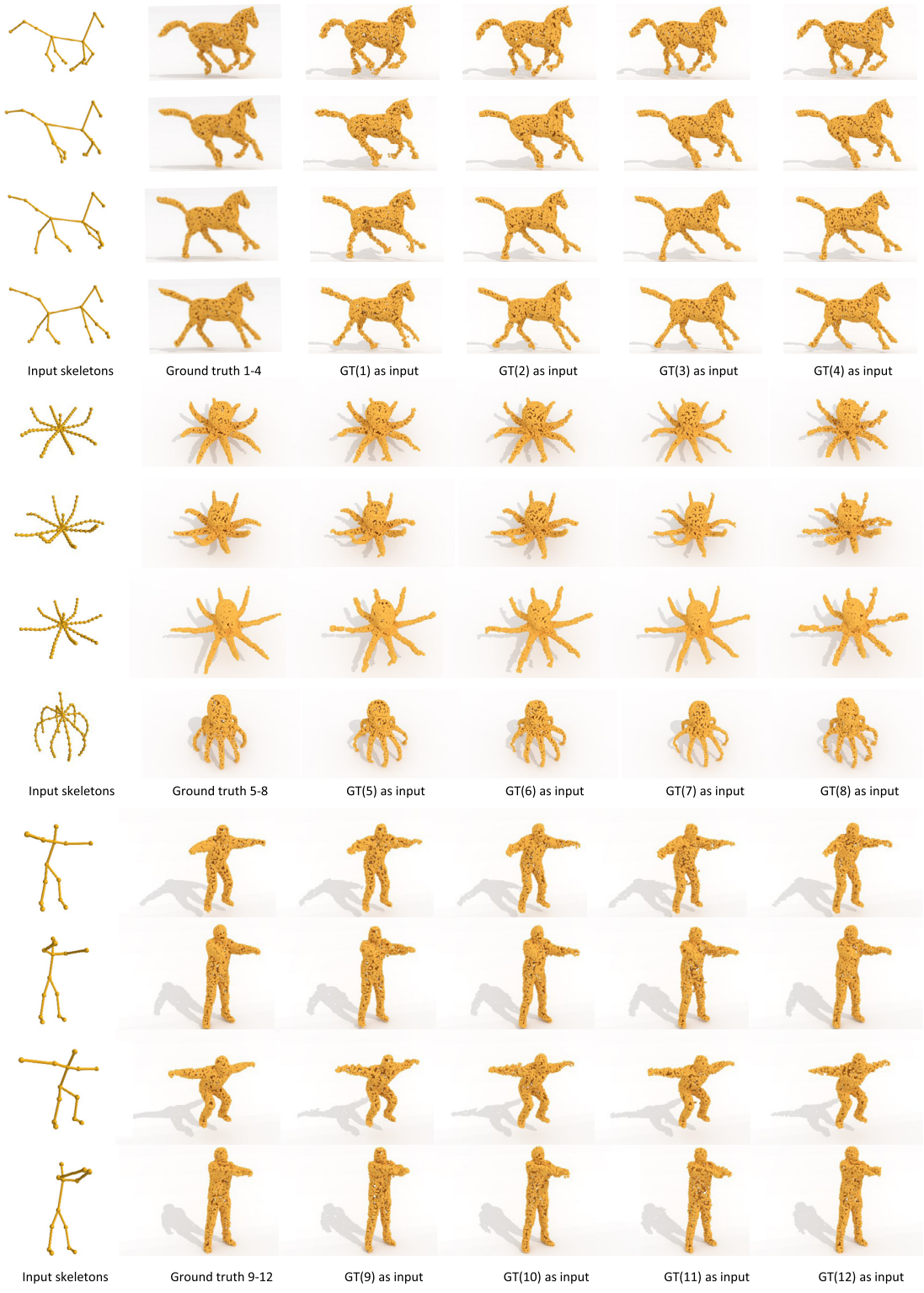


Fig. 7. Horse, octopus and human models generated from our network. Shapes in column 3-6 are results with inputs referenced in subscript below and skeletons in column 1.

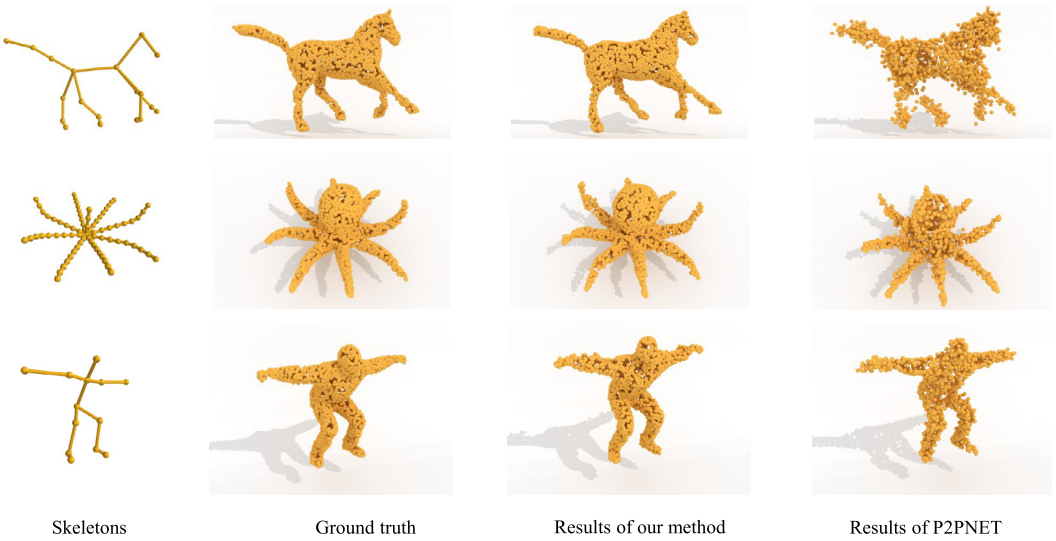


Fig. 8. Different models generated using our method and P2P-NET method. Since different input shapes have little effect on the results of our method, we randomly choose one as input from dataset for each category.

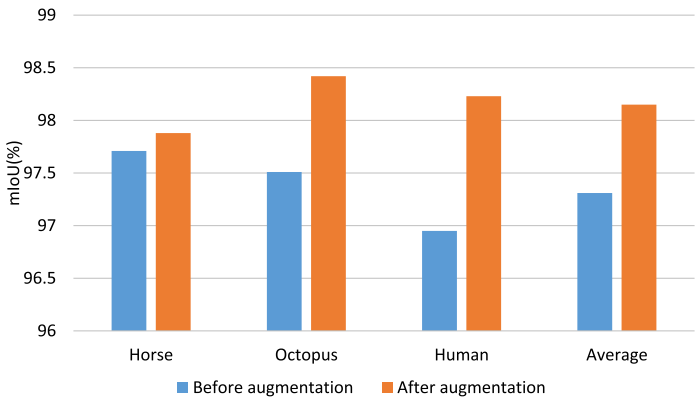


Fig. 9. Segmentation results before and after augmentation.

For example, to augment datasets tagged with segmentation labels, we can preserve the segmentation label by constraining the deformation process within points of same labels. To this end, we define the shape loss for tag-preserving augmentation as

$$L'_{chamfer}(\hat{S}, S) = \sum_{p\in\hat{S}} min_{q\in T_p(S)} \|p-q\|_2^2 + \sum_{q\in S} min_{p\in T_q(\hat{S})} \|p-q\|_2^2, \tag{17}$$

where $T_p(S)$ is the points on the shape S that have same tags with point p. To augment datasets with point-to-point correspondences, we can simply change $T_p(S)$ to the corresponding points. Here we show our augmentation results for segmentation task as an example. We apply PointNet++ Qi et al. (2017) part segmentation network to implement the shape segmentation task. Note that PointNet++ is designed to be invariant to rigid transformation, thus our method is quite suitable in this case. We evaluate the segmentation result with mIoU, aka average IoUs, for all part types in each category, as is used in Qi et al. (2017). We put horse, octopus and human datasets together to train the segmentation network. For octopus category, we choose 36 shapes from the original dataset and use 30 of them as training data. For human and horse datasets, we choose 40 of them as training data as before. We augment each training dataset to 100 shapes. Then we train the network using datasets with and without augmentation and evaluate the performance using the same testing set. Fig. 9 shows the improvements of segmentation performance using augmented dataset compared to original training data, our augmentation method improves the performance even when the data is nearly saturation.

6. Discussion and conclusion

We propose a quantitative and flexible 3D shape dataset augmentation framework, which contains a point cloud embedding method, a flexible navigation method as well as a generative network who learns deformation process in embedding space. Our novel idea is to contract skeletons for point clouds and project them onto a shared embedding space and learn the deformation process in an embedding space. Different kinds of navigation methods are provided for flexible control. This method requires no point-to-point correspondence and yet is capable of generating new, high quality shapes from a small dataset. The results presented in the previous section demonstrate the efficacy of our framework.

Despite the attractive properties of this methodology detailed in our system framework, our approach still has some limitations that should be overcome in our future work. For the purpose of controlling the generation process in a quantitative and flexible way, we use the skeleton-based representation and align the parts into a same-length vector. However, the alignment method can not explore the topological variations in a better sense, thus the method is not suitable for dataset with shapes of quite different topology, such as tables, desks and chairs. Furthermore, the skeleton extraction step may filter out the information of details, which is predicted automatically by our network and thus not controllable by user. In the future, we may try to change our way of presentation to encode more structure information. Currently, our point cloud is represented by 2048 points, which is barely sufficient for models with huge details. To generate shapes with more details is also an issue of interest to us.

Acknowledgements

The work is supported by the National Natural Science Foundation of China under Grant Nos. 61672077, 61532002 and 61872347, the Applied Basic Research Program of Qingdao under Grant No. 161013xx, and the National Science Foundation of USA under Grant Nos. IIS-1715985 and IIS-1812606.

References

Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.J., 2017. Learning representations and generative models for 3D point clouds. CoRR arXiv:1707.02392 [abs]. URL: http://arxiv.org/abs/1707.02392, arXiv:1707.02392.

Alhashim, I., Li, H., Xu, K., Cao, J., Ma, R., Zhang, H., 2014. Topology-varying 3D shape creation via structural blending. ACM Trans. Graph. 33, 158. https://doi.org/10.1145/2601097.2601102. URL: http://doi.acm.org/10.1145/2601097.2601102.

Bailey, S.W., Watt, M., O'Brien, J.F., 2016. Repurposing hand animation for interactive applications. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 1–10. URL: http://graphics.berkeley.edu/papers/Bailey-RHA-2016-07/.

Campbell, N.D.F., Kautz, J., 2014. Learning a manifold of fonts. ACM Trans. Graph. 33, 91. https://doi.org/10.1145/2601097.2601212. URL: http://doi.acm.org/10.1145/2601097.2601212.

Cao, J., Tagliasacchi, A., Olson, M., Zhang, H., Su, Z., 2010. Point cloud skeletons via Laplacian based contraction. In: Proceedings of the 2010 Shape Modeling International Conference. IEEE Computer Society, Washington, DC, USA, pp. 187–197. URL: https://doi.org/10.1109/SMI.2010.25.

Chen, K., Choy, C.B., Savva, M., Chang, A.X., Funkhouser, T.A., Savarese, S., 2018. Text2shape: generating shapes from natural language by learning joint embeddings. CoRR arXiv:1803.08495 [abs]. URL: http://arxiv.org/abs/1803.08495, arXiv:1803.08495.

Dai, A., Qi, C.R., Nießner, M., 2016. Shape completion using 3D-encoder-predictor CNNs and shape synthesis. CoRR arXiv:1612.00101 [abs]. URL: http://arxiv.org/abs/1612.00101, arXiv:1612.00101.

Fan, H., Su, H., Guibas, L.J., 2016. A point set generation network for 3D object reconstruction from a single image. CoRR arXiv:1612.00603 [abs]. URL: http://arxiv.org/abs/1612.00603, arXiv:1612.00603.

Girdhar, R., Fouhey, D.F., Rodriguez, M., Gupta, A., 2016. Learning a predictable and generative vector representation for objects. CoRR arXiv:1603.08637 [abs]. URL: http://arxiv.org/abs/1603.08637, arXiv:1603.08637.

Han, Z., Liu, Z., Han, J., Bu, S., 2015. 3d shape creation by style transfer. Vis. Comput. 31, 1147–1161. https://doi.org/10.1007/s00371-014-0999-1. URL: https://doi.org/10.1007/s00371-014-0999-1.

Hastie, T., Stuetzle, W., 1989. Principal curves. J. Am. Stat. Assoc. 84, 502–516. https://doi.org/10.1023/A:1026543900054.

Hernández-García, A., König, P., 2018. Do deep nets really need weight decay and dropout? arXiv preprint arXiv:1802.07042.

Hu, R., Yan, Z., Zhang, J., van Kaick, O., Shamir, A., Zhang, H., Huang, H., 2018. Predictive and generative neural networks for object functionality. ACM Trans. Graph. 37, 151.

Huang, S., Fu, H., Wei, L., Hu, S., 2016. Support substructures: support-induced part-level structural representation. IEEE Trans. Vis. Comput. Graph. 22, 2024–2036. https://doi.org/10.1109/TVCG.2015.2473845.

Lawrence, N., 2005. Probabilistic non-linear principal component analysis with Gaussian process latent variable models. J. Mach. Learn. Res. 6, 1783–1816. URL: http://dl.acm.org/citation.cfm?id=1046920.1194904.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436.

Levine, S., Wang, J.M., Haraux, A., Popović, Z., Koltun, V., 2012. Continuous character control with low-dimensional embeddings. ACM Trans. Graph. 31, 28. https://doi.org/10.1145/2185520.2185524. URL: http://doi.acm.org/10.1145/2185520.2185524.

Li, J., Xu, K., Chaudhuri, S., Yumer, E., Zhang, H., Guibas, L., 2017. Grass: generative recursive autoencoders for shape structures. ACM Trans. Graph. 36, 52. https://doi.org/10.1145/3072959.3073637. URL: http://doi.acm.org/10.1145/3072959.3073637.

Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2016. Pointnet: deep learning on point sets for 3D classification and segmentation. CoRR arXiv:1612.00593 [abs]. URL: http://arxiv.org/abs/1612.00593, arXiv:1612.00593.

Qi, C.R., Yi, L., Su, H., Guibas, L.J., 2017. Pointnet++: deep hierarchical feature learning on point sets in a metric space. CoRR arXiv:1706.02413 [abs]. URL: http://arxiv.org/abs/1706.02413, arXiv:1706.02413.

Ranaweera, W., Chilana, P., Cohen-Or, D., Zhang, H., 2017. ExquiMo: an exquisite corpse tool for co-creative 3D shape modeling. In: International Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics).

Rasmussen, C.E., Williams, C.K.I., 2005. Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press.

Rubner, Y., Tomasi, C., Guibas, L.J., 2000. The Earth mover's distance as a metric for image retrieval. Int. J. Comput. Vis. 40, 99–121. https://doi.org/10.1023/A:1026543900054. URL: https://doi.org/10.1023/A:1026543900054.

Sung, M., Su, H., Kim, V.G., Chaudhuri, S., Guibas, L.J., 2017. Complementme: weakly-supervised component suggestions for 3D modeling. CoRR, arXiv: 1708.01841 [abs]. URL: http://arxiv.org/abs/1708.01841, arXiv:1708.01841.

Tatarchenko, M., Dosovitskiy, A., Brox, T., 2017. Octree generating networks: efficient convolutional architectures for high-resolution 3D outputs. In: Proc. of the IEEE International Conf. on Computer Vision. ICCV, p. 8.

Titsias, M., Lawrence, N.D., 2010. Bayesian Gaussian process latent variable model. In: Teh, Y.W., Titterington, M. (Eds.), Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. PMLR, Chia Laguna Resort, Sardinia, Italy, pp. 844–851. URL: http://proceedings.mlr.press/v9/titsias10a.html.

Turmukhambetov, D., Campbell, N.D., Goldman, D.B., Kautz, J., 2015. Interactive sketch-driven image synthesis. Comput. Graph. Forum 34, 130–142. https://doi.org/10.1111/cgf.12665. URL: http://doi.org/10.1111/cgf.12665.

Von-Tycowicz, C., Schulz, C., Seidel, H.P., Hildebrandt, K., 2015. Real-time nonlinear shape interpolation. ACM Trans. Graph. (TOG) 34, 34.

Wu, J., Zhang, C., Xue, T., Freeman, B., Tenenbaum, J., 2016. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In: Advances in Neural Information Processing Systems, pp. 82–90.

Xia, Q., Li, S., Qin, H., Hao, A., 2015. Modal space subdivision for physically-plausible 4D shape sequence completion from sparse samples. In: Pacific Graphics Short Papers. The Eurographics Association, pp. 19–24.

Xu, D., Zhang, H., Wang, Q., Bao, H., 2006. Poisson shape interpolation. Graph. Models 68, 268–281.

Yin, K., Huang, H., Cohen-Or, D., Zhang, H.R., 2018a. P2P-NET: bidirectional point displacement network for shape transform. CoRR: arXiv:1803.09263 [abs]. URL: http://arxiv.org/abs/1803.09263, arXiv:1803.09263.

Yin, K., Huang, H., Ho, E.S.L., Wang, H., Komura, T., Cohen-Or, D., Zhang, R., 2018b. A sampling approach to generating closely interacting 3D pose-pairs from 2D annotations. IEEE Trans. Vis. Comput. Graph. Available online: https://ieeexplore.ieee.org/document/8353147.