Variational Autoencoders for Localized Mesh Deformation Component Analysis

Qingyang Tan†, Ling-Xiao Zhang†, Jie Yang, Yu-Kun Lai and Lin Gao*

Abstract—Spatially localized deformation components are very useful for shape analysis and synthesis in 3D geometry processing. Several methods have recently been developed, with an aim to extract intuitive and interpretable deformation components. However, these techniques suffer from fundamental limitations especially for meshes with noise or large-scale nonlinear deformations, and may not always be able to identify important deformation components. In this paper we propose a mesh-based variational autoencoder architecture that is able to cope with meshes with irregular connectivity and nonlinear deformations, assuming that the analyzed dataset contains meshes with the same vertex connectivity, which is common for deformation analysis. To help localize deformations, we introduce sparse regularization in this framework, along with spectral graph convolutional operations. Through modifying the regularization formulation and allowing dynamic change of sparsity ranges, we improve the visual quality and reconstruction ability of the extracted deformation components. Our system also provides a nonlinear approach to reconstruction of meshes using the extracted basis, which is more effective than the current linear combination approach. As an important application of localized deformation components and a novel approach on its own, we further develop a neural shape editing method, achieving shape editing and deformation component extraction in a unified framework, and ensuring plausibility of the edited shapes. Extensive experiments show that our method outperforms state-of-the-art methods in both qualitative and quantitative evaluations. We also demonstrate the effectiveness of our method for neural shape editing.

Index Terms—3D Meshes, Variational Autoencoder, Graph Convolution, Sparsity Regularization

1 Introduction

With the development of 3D scanning and modeling technology, mesh data sets are becoming increasingly popular. By analyzing these data sets with machine learning techniques, the latent knowledge can be exploited to advance geometry processing algorithms. In recent years, many research areas in geometry processing have benefited from this methodology, such as 3D shape deformation [1], 3D facial and human body reconstruction [2], [3], shape segmentation [4], etc. For shape deformation and human reconstruction, mesh sequences with different geometry and the same connectivity play a central role. Different geometric positions describe the appearance of the 3D mesh model while sharing the same vertex connectivity makes processing much more convenient. In such works, a key procedure is to build a low-dimensional control parametrization for the mesh data set, which provides a small set of intuitive parameters to control the generation of new shapes. For articulated models such as human bodies, the rigging method embeds a skeleton structure in the mesh to provide such a parametrization. However, the rigging operation is restrictive and does not generalize to other deformable shapes (e.g. faces). Parameterizing general mesh datasets which allows intuitive control in generating new shapes becomes an important and urgent research problem.

Early work extracted principal deformation components by using Principal Component Analysis (PCA) to reduce the dimensionality of the data set. However, such deformation components are global which do not lead to intuitive control. For example, when a user intends to deform the shape locally by specifying locally changed vertex positions as boundary conditions, the deformed shape tends to have unrelated areas deformed as well, due to the global nature of the basis. To address this, sparse localized deformation component (SPLOCS) extraction methods were recently proposed [5], [6], [7] on mesh deformation datasets with the same vertex connectivity. In these works the sparsity term is involved to localize deformation components within local support regions. However, these previous works suffer from different limitations: as we will show later, [5], [6] cannot handle large-scale deformations, and [7] is sensitive to noise so cannot extract the main deformation components robustly. Especially, the methods [5], [8] are suitable for analyzing localized shape deformation without large-scale rotations. In contrast, our method is designed to handle shape collections with large-scale nonlinear deformations while also performing well on shape collection with linear deformations.

We propose a mesh-based variational autoencoder (VAE) architecture to extract meaningful local deformation components. We represent deformations of shapes in the dataset based on a recent effective representation [9] which is able to cope with large deformations. We then build a variational autoencoder to transform the deformation representation to encoding in a latent space. Each convolutional layer involves convolutional operations

- † *Authors contributed equally*
- * *Corresponding author is Lin Gao (gaolin@ict.ac.cn)*
- *Qingyang Tan, Ling-Xiao Zhang, Jie Yang and Lin Gao are with the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences. QingYang Tan is also with University of Maryland, College Park, USA. Jie Yang and Lin Gao are also with University of Chinese Academy of Sciences, Beijing, China.*
  E-mail: {zhanglingxiao, yangjie01, gaolin}@ict.ac.cn, qytan@cs.umd.edu
- *Yu-Kun Lai is with School of Computer Science & Informatics, Cardiff University, UK.*
  E-mail: LaiY4@cardiff.ac.uk

defined on the mesh as a graph with arbitrary topology. We then introduce sparsity regularization to the weights in the fully-connected layers to promote identifying sparse localized deformations. In existing sparse deformation component extraction methods [5], an important factor that affects the extracted components is the sparse range, which includes globally tunable parameters deciding how local the extracted deformation components should be, but often a single global setting does not fit all the deformation components. To allow more flexible sparsity ranges while avoiding the need for adjusting parameters in the regularization term, we propose a novel sparsity weight formulation with sparsity ranges automatically learned, and adapted to suit individual deformation components. The variational autoencoder structure ensures that the extracted deformation components are suitable for reconstructing high quality shape deformations.

Our main contributions are:

- This is the first work that exploits convolutional neural network (CNN) based variational autoencoder with spectral graph convolutional operators for sparse deformation component extraction on meshes with irregular connectivity and large deformation.
- To achieve this, our method exploits the nonlinear representation capability of variational autoencoders, along with a sparse regularization. As a result, it is able to extract intuitive localized deformation components. It can also deal with datasets with large-scale deformations, and is insensitive to noise.
- We develop a formulation that automatically learns adaptive sparsity ranges for individual deformation components, leading to improved deformation extraction.

The method can extract important components even for challenging cases and generalizes well to reconstruction of unseen data. Extensive qualitative and quantitative experiments demonstrate that our method outperforms the state-of-the-art methods. We show an example of extracted deformation components (highlighted in blue) in Fig. 1, which are then combined to synthesize a novel and plausible shape. The architecture of our proposed network is illustrated in Fig. 2.

This paper substantially extends our original conference version [10] in the following ways: We improve the overall system architecture, by replacing the autoencoder architecture with a variational autoencoder and introducing a new sparsity regularization term with the sparsity ranges automatically learned. We further build our architecture based on graph convolutions in the spectral domain. As a result, our new architecture has better reconstruction and generation ability, and can derive more natural deformation components. We further introduce a framework for shape editing with neural design. It achieves shape editing by optimizing the latent vector such that the decoded shape satisfies the given control point constraints. Our method is capable of producing plausible deformed shapes for local and/or large-scale deformations, with only a small number of control points. We also show experimentally that our method with graph convolutions in the spectral domain achieves better results than with graph convolutions in the spatial domain, as used in [10]. The source code is available at https://github.com/IGLICT/MeshVAE_Component_Analysis.

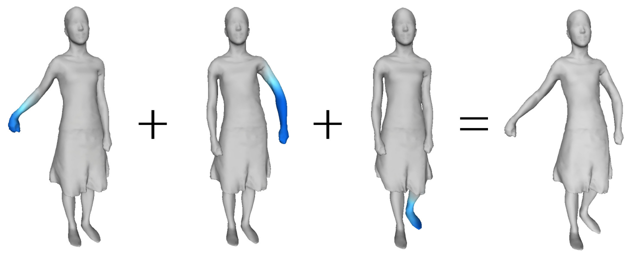


Figure 1. Synthesizing a new mesh model by combining deformation components derived from the Swing dataset [11] using our method with equal weights.

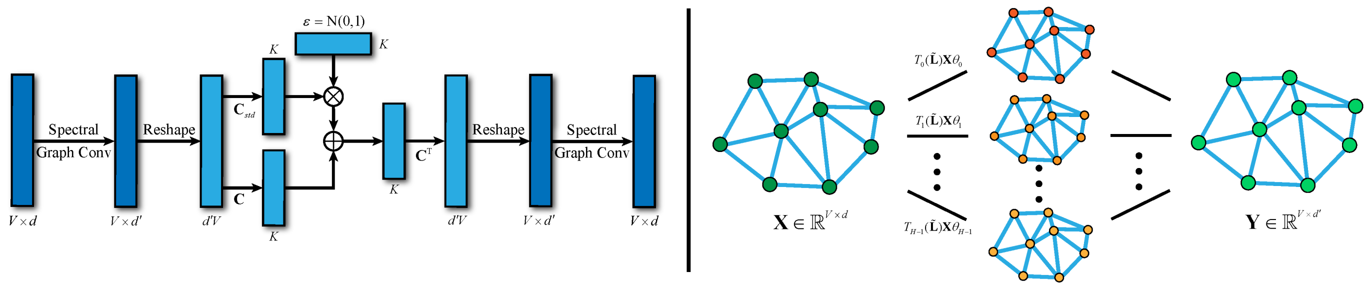
2 Related Work

2.1 Principal Deformation Component Analysis

With the increasing availability of 3D shapes, analyzing shape collections is becoming more important. Thanks to advances of shape matching methods [12], [13] and non-rigid registration technology [14], mesh data sets for deformable shapes with the same vertex connectivity are becoming much more common. Early work employs PCA to compress the mesh data set and extract global deformation components [15]. The deformation components from the PCA are globally supported, which is not intuitive for shape editing and deformation, especially when the user wants to deform the shape locally in the spatial domain [16]. Sparse regularization is effective in localizing deformations [17]. However, standard sparse PCA [18] does not take spatial constraints into account and therefore the extracted deformation components do not aggregate in local spatial domains. By incorporating spatial constraints, a sparsity term is employed to extract localized deformation components [5], [8], which performs better than region-based PCA variants (clustered PCA) [19] in terms of extracting meaningful localized deformation components. However, it uses Euclidean coordinates which cannot represent shapes with large rotations. Later work addresses this limitation by using more advanced shape representations including deformation gradients [6] and edge and dihedral angle representations [7]. However, the former cannot cope with rotations larger than 180° which are very common in animated mesh sequences, while the latter is not sensitive to the scale of the deformations which makes [7] not robust to noise. Unlike existing methods, we propose to exploit mesh-based variational autoencoders with sparse regularization along with an effective deformation representation [9] to extract high-quality deformation components, outperforming existing methods.

2.2 Neural Network Applications for 3D Shapes

Neural networks have achieved great success in different areas of computer science. Compared with 2D images, 3D shapes are more difficult to process, mainly due to their


$\varepsilon = N(0,1)$
K
Spectral Graph Conv
Reshape
$\mathbf{C}_{std}$
$\mathbf{C}$
$\mathbf{C}^{\mathrm{T}}$
Reshape
Spectral Graph Conv
$V \times d$
$V \times d'$
$d'V$
K
$d'V$
$V \times d'$
$V \times d$
$T_0(\tilde{\mathbf{L}})\mathbf{X}\theta_0$
$T_1(\tilde{\mathbf{L}})\mathbf{X}\theta_1$
$T_{H-1}(\tilde{\mathbf{L}})\mathbf{X}\theta_{H-1}$
$\mathbf{X} \in \mathbb{R}^{V \times d}$
$\mathbf{Y} \in \mathbb{R}^{V \times d'}$

(a) Network Architecture (b) Spectral Graph Convolution

Figure 2. The proposed network architecture (a) and the spectral graph convolution (b). The network is built based on the spectral convolutional operation and with a variational autoencoder structure, and we set $d' = 9$. The input and output are mesh features $\mathbf{X} \in \mathbb{R}^{V \times 9}$. K is the dimension of the latent space and we set $K = 50$. $\mathbf{C}_{std}$ and $\mathbf{C}$ are the weights of fully connected layers to map the features to the latent space. The spectral graph convolution uses graph Fourier transform to filter the features in the spectral domain, and more details are described in Sec. 4.1.

irregular connectivity and limited data availability. Nevertheless, some effort was made in recent years. For 3D object recognition, Su et al. [20] and Shi et al. [21] represent 3D shapes using multi-view projections or by converting them to panoramic views and utilizing 2D CNNs. Maturana and Scherer [22] treat 3D shapes as voxels and extend 2D-CNNs to 3D-CNNs to recognize 3D objects. In addition, Li et al. [23] analyze a joint embedding space of 2D images and 3D shapes. Tulsiani et al. [24] abstract complex shapes using 3D volumetric primitives. For 3D shape synthesis, Wu et al. [25] use deep belief networks to generate voxelized 3D shapes. Girdhar et al. [26] combine an encoder for 2D images and a decoder for 3D models to reconstruct 3D shapes from 2D input. Yan et al. [27] generate 3D models from 2D images by adding a projection layer from 3D to 2D. Choy et al. [28] propose a novel recurrent network to map images of objects to 3D shapes. Sharma et al. [29] train a volumetric autoencoder using noisy data with no labels for tasks such as denoising and completion. Wu et al. [30] exploit the power of the generative adversarial network with a voxel CNN. In addition to voxel representation, Sinha et al. [31] propose to combine ResNet and geometry images to synthesize 3D models. Li et al. [32] and Nash and Williams [33] propose to use neural networks for encoding and synthesizing 3D shapes based on pre-segmented data. Some works [34], [35] propose novel networks based on Signed Distance Functions (SDFs) for representing detailed geometry of objects with arbitrary topology. The works [36], [37] further synthesize structured 3D shapes, but these methods mainly work for man-made objects and also require semantic segmentation of training meshes. All the methods above for synthesizing 3D models are restricted by their representations or primitives adopted, which are not suitable for analyzing and generating 3D motion sequences with rich details, and none of these methods are designed for extracting deformation components.

2.3 Convolutional Neural Networks (CNNs) on Arbitrary Graphs and Meshes

Traditional CNNs are defined on 2D images or 3D voxels with regular grids. Some research has explored the potential to extend CNNs to irregular graphs (such as meshes) through construction in the spectral domain [38], [39], [40] or the spatial domain [41], [42] focusing on spatial construction. Such representations were exploited in recent work [43], [44] for finding correspondences or performing part-based segmentation on 3D shapes. Our method is based on representative constructions in the spectral domain and utilizes this to build a variational autoencoder for analyzing deformation components.

3 Feature Representation

To represent large-scale deformations, we adapt a recently proposed deformation representation [9]. Given a dataset with N shapes with the same vertex connectivity, each shape is denoted as S_m, $m \in [1, \ldots, N]$. $\mathbf{p}_{m,i} \in \mathbb{R}^3$ is the i^{th} vertex on the m^{th} mesh model. The deformation gradient $\mathbf{T}_{m,i} \in \mathbb{R}^{3\times 3}$ representing local shape deformations can be obtained by minimizing:

$$\arg\min_{\mathbf{T}_{m,i}} \sum_{j\in N(i)} c_{ij} \|(\mathbf{p}_{m,i} - \mathbf{p}_{m,j}) - \mathbf{T}_{m,i}(\mathbf{p}_{1,i} - \mathbf{p}_{1,j})\|_2^2. \quad (1)$$

where c_{ij} is the cotangent weight $c_{ij} = \cot\alpha_{ij} + \cot\beta_{ij}$, and α_{ij} and β_{ij} are angles opposite to the edge connecting vertices v_i and v_j, and $N(i)$ is the index set of 1-ring neighbors of the i^{th} vertex. By polar decomposition $\mathbf{T}_{m,i} = \mathbf{R}_{m,i}\mathbf{S}_{m,i}$, the affine matrix $\mathbf{T}_{m,i} \in \mathbb{R}^{3\times 3}$ can be decomposed into an orthogonal matrix $\mathbf{R}_{m,i}$ describing rotations, and a real symmetry matrix $\mathbf{S}_{m,i}$ for scale and shear deformations. The rotation matrix $\mathbf{R}_{m,i}$ can be rewritten as rotating around an axis $\boldsymbol{\omega}_{m,i}$ by an angle $\theta_{m,i}$. However, the mapping from the axis-angle representation to rigid rotation is surjective but not one to one: The rotation angles and axes in the set $\Omega_{m,i}$ correspond to one rigid rotation:

$$\Omega_{m,i} = \{(\boldsymbol{\omega}_{m,i}, \theta_{m,i} + t \cdot 2\pi), (-\boldsymbol{\omega}_{m,i}, -\theta_{m,i} + t \cdot 2\pi)\} \quad (2)$$

where t is an arbitrary integer. To overcome this, [9] proposes a novel representation to select the unique and consistent axis-angle representation by solving a global optimization to minimize the differences between adjacent rotation axes and angles.

For each vertex i of shape m, we obtain feature $\mathbf{q}_{m,i} = \{\mathbf{r}_{m,i}, \mathbf{s}_{m,i}\} \in \mathbb{R}^9$ by extracting from matrices $\log(\mathbf{R}_{m,i})$ and $\mathbf{S}_{m,i}$. To fit the range of activation function $tanh$ (explained later in Sec. 4.2) and to avoid the gradient vanishing problem [45], we need to scale the feature values. Denote by $\mathbf{r}_{m,i}^{j}$ and $\mathbf{s}_{m,i}^{j}$ the j^{th} element of $\mathbf{r}_{m,i}$ and $\mathbf{s}_{m,i}$ respectively. Separately for each element j, we linearly scale $\mathbf{r}_{m,i}^{j}$ and

$\mathbf{s}^j_{m,i}$ from $[r_{min}, r_{max}]$ and $[s_{min}, s_{max}]$ to $[-0.95, 0.95]$ to avoid the gradient vanishing problem [45] to acquire preprocessed $\widetilde{\mathbf{r}}^j_{m,i}$ and $\widetilde{\mathbf{s}}^j_{m,i}$, where $r_{min} = \min_{m,i,j} \mathbf{r}^j_{m,i}$, and r_{max}, s_{min}, s_{max} are defined similarly. Then, we have $\mathbf{X}_{m,i} = \{\widetilde{\mathbf{r}}_{m,i}, \widetilde{\mathbf{s}}_{m,i}\}$ as the deformation feature for vertex i of shape m.

4 Network Architecture

In this section, we present our framework including convolutional operations on irregular meshes, overall network structure, sparsity constraints and reconstruction loss.

4.1 Convolutional Operation

CNNs have demonstrated power in several computer vision tasks, including classification, object detection, semantic segmentation, etc. Researchers have extended CNNs to graphs without regular grids, including techniques in both spatial and spectral domains. In our network, we choose the convolutional operator in the spectral domain due to its superior performance, rather than the spatial domain as used in [10].

We follow [40] to build our spectral graph CNN. We use the same formulation of input and output data as in the spatial domain. Besides input and output data, we define the graph of a mesh as $G = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where $\mathcal{V}$ is the set of vertices and $|\mathcal{V}| = V$, $\mathcal{E}$ is the set of edges, $\mathbf{W} \in \mathbb{R}^{V \times V}$ is the weighted adjacency matrix on the edges of the graph, and V is the number of vertices. The operation is illustrated in Fig. 2 (b).

We can denote the spectral graph convolution used in the network as

$$\mathbf{Y} = \sum_{h=0}^{H-1} T_h(\widetilde{\mathbf{L}})\mathbf{X}\theta_h \tag{3}$$

where $\mathbf{X} \in \mathbb{R}^{V \times d}$ and $\mathbf{Y} \in \mathbb{R}^{V \times d'}$ are the input and output features, H is the order of the filter, which means the spectral filter is H-localized, i.e. only acting on local neighborhoods with a distance limit of H edges, $\mathbf{T}_h(\widetilde{\mathbf{L}}) \in \mathbb{R}^{V \times V}$ is the Chebyshev polynomial of order h evaluated at $\widetilde{\mathbf{L}}$, and $\theta_h \in \mathbb{R}^{d \times d'}$ are trainable weights, d and d' are the dimensions of input data and output data respectively. $\widetilde{\mathbf{L}} = 2\mathbf{L}/\lambda_{max} - \mathbf{I}_n$ is the scaled Laplacian matrix, where λ_{max} is the maximum eigenvalue of $\mathbf{L}$ and $\mathbf{L} = -\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ is the normalized graph Laplacian matrix, $\mathbf{I}_n$ is the n-dimensional identity matrix, $\mathbf{D} \in \mathbb{R}^{V \times V}$ is the diagonal degree matrix where $D_{ii} = \sum_j \mathbf{W}_{ij}$ is the i^{th} entry of the diagonal of the degree matrix. Please refer to [40] for more details. We use $H = 3$ in all experiments and we compare the generalization ability of different values of H in Sec. 6.

4.2 Network Structure

The whole network pipeline is illustrated in Fig. 2 (a). It is built based on the convolutional operation and with a variational autoencoder structure. The input to the encoder part contains preprocessed features which are shaped as $\mathbf{X} \in \mathbb{R}^{V \times 9}$, where 9 is the dimension of the deformation representation for a vertex. Then we apply a convolutional layer with $tanh$ as the output activation function. We tested alternative functions like $ReLU$, but they performed worse in the quantitative analysis. The output from the convolutional layer is reshaped as a vector $\mathbf{f} \in \mathbb{R}^{d'V}$, where d' is the output dimension of the convolutional layer. We use a fully connected layer with $\mathbf{C} \in \mathbb{R}^{K \times d'V}$ and $\mathbf{C}_{std} \in \mathbb{R}^{K \times d'V}$ to map the features to the latent space $\mathbf{z} \in \mathbb{R}^K$ where K is the dimension of the latent space:

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma}\boldsymbol{\varepsilon} = \mathbf{C}\mathbf{f} + \mathbf{C}_{std}\mathbf{f}\boldsymbol{\varepsilon}, \tag{4}$$

where $\boldsymbol{\varepsilon} = \mathcal{N}(\mathbf{0}, \mathbf{I})$ is the Gaussian noise with zero mean and unit variation. To reconstruct the shape representation from $\mathbf{z}$, we use the decoder, which basically mirrors the encoder steps. We first use the transpose of $\mathbf{C}$ to transfer from the latent space back to the feature space:

$$\widehat{\mathbf{f}} = \mathbf{C}^{\mathrm{T}}\mathbf{z}. \tag{5}$$

For the decoder, we use the same number of convolutional layers and activation function as the encoder. The output of the whole network is $\widehat{\mathbf{X}} \in \mathbb{R}^{V \times 9}$ which has the identical dimension as the input and can be scaled back to the deformation representation [9] and used for reconstructing the deformed shape.

The tied weight formulation of the variational autoencoder makes it more like PCA, and we assume that $\mathbf{F} \in \mathbb{R}^{N \times d'V}$ is assembled by stacking all the features $\mathbf{f}$ extracted from the convolutional layer for N models in the dataset. Then, $\mathbf{C}$ can be seen as K deformation components of $\mathbf{F}$, and $\mathbf{Z} \in \mathbb{R}^{N \times K}$ stacks the latent representations of the N models in the dataset, which is treated as combinational weights to reconstruct the shape.

4.3 Sparsity Constraints and Reconstruction Loss

Following the idea from [5], we use group sparsity ($\ell_{2,1}$ norm) to urge deformation components to only capture localized deformations. The constraints are added on $\mathbf{C}$ as:

$$\Omega(\mathbf{C}) = \frac{1}{K}\sum_{k=1}^{K}\sum_{i=1}^{V} \Delta_{ik}\|\mathbf{C}^i_k\|_2, \tag{6}$$

where $\mathbf{C}^i_k$ is the d'-dimensional vector associated with component k of vertex i, and Δ_{ik} is a sparsity regularization weight.

Different from the original fixed sparsity constraints used in [10], we determine the sparsity weights as follows

$$\Delta_{ik} = \delta(d_{ik}, \mathbf{d}_k) = \begin{cases} 0 & d_{ik} \le \mathbf{d}_k \\ 1 & d_{ik} > \mathbf{d}_k. \end{cases} \tag{7}$$

Here, $\delta(\cdot,\cdot)$ is a function that maps a geodesic distance to 0 or 1 on the two sides of $\mathbf{d}_k$, where $\mathbf{d}_k, 1 \le k \le K$ is a learnable parameter in the k^{th} entry of $\mathbf{d}$. $\mathbf{d}$ allows different sparsity ranges for different components; such flexibility leads to more effective deformation component extraction. We initialize $\mathbf{d}_k$ as 0.2, and $\mathbf{d}_k$ is directly optimized as learnable parameters. Since $\mathbf{d}$ is learned during training, we also avoid the manual adjustment process for d_{min} and d_{max} required for the method in [10]. During training, increasing $\mathbf{d}_k$ ($1 \le k \le K$) (thus setting more δ_{ik} to 0) will decrease $\Omega(\mathbf{C})$. To avoid degenerate solution with all $\mathbf{d}_k$ to be ones, we introduce a regularization term

$$\Phi(\mathbf{d}) = \|\mathbf{d}\|_2 \tag{8}$$

TABLE 1
Errors of applying our method to generate unseen data from Swing [11], Horse [46], Face [47], Jumping [11], Humanoid, a fat person (ID: 50002) from the MPI DYNA [48] datasets, SCAPE [49] and Female Human [50]. We train all these methods with 50 components ($K = 50$). We compare different methods with E_{rms} and $STED$ including methods [5], [6], [8] that are suitable for deformations and methods [7], [9], [10] that are suitable for non-linear deformations. For the sake of completeness, we also evaluate these methods [5], [6], [8] for linear deformations on non-linear deformation datasets such as SCAPE and Swing. Especially the $STED$ error is designed for motion sequences with a focus on 'perceptual' error of models. The $STED$ error of SCAPE and Female Human are not given because they are not sequential datasets, so $STED$ is not applicable.

Dataset	Metric	Methods for Linear Deformations			Methods for Non-linear Deformations			
		Bernard et al. [8]	Huang et al. [6]	Neumann et al. [5]	Wang et al. [7]	Gao et al. [9]	Tan et al. [10]	Ours
Swing	E_{rms}	21.7586	24.4950	15.1941	23.3055	25.2994	14.0836	**13.3625**
	$STED$	0.1139	0.04344	0.08309	0.04250	0.05214	0.03789	**0.03352**
Horse	E_{rms}	20.1994	18.0624	**10.5185**	29.6090	42.4180	12.9605	12.4006
	$STED$	0.4111	0.05273	0.08074	0.04332	0.07841	0.04004	**0.0397**
Face	E_{rms}	2.9853	12.3221	2.9106	8.5620	2.9260	2.9083	**1.2047**
	$STED$	0.02662	0.01827	0.008611	0.01320	0.009676	0.007344	**0.007135**
Jumping	E_{rms}	45.8762	37.9915	47.8150	44.3362	22.3001	24.4827	**20.9021**
	$STED$	0.4323	0.06305	0.1728	0.05400	0.04995	0.04862	**0.04592**
Humanoid	E_{rms}	6.6320	16.1995	14.3610	60.9925	5.6510	3.4912	**2.9795**
	$STED$	0.04612	0.02247	0.07319	0.03757	0.01283	0.01313	**0.01234**
Fat	E_{rms}	5.7207	8.1438	5.5785	47.0212	5.0909	4.7795	**3.2062**
	$STED$	0.1505	0.02185	0.06059	0.02511	0.01453	0.01625	**0.01241**
SCAPE	E_{rms}	14.6233	24.6597	16.3970	79.9577	17.1754	13.5560	**11.8712**
	$STED$	/	/	/	/	/	/	/
Female	E_{rms}	6.3172	9.9553	7.2462	25.1312	10.3842	5.8214	**5.2471**
	$STED$	/	/	/	/	/	/	/

to suppress this. Since the gradient of δ does not always exist, we use the straight through estimator (STE) [51] to approximate it.

d_{ik} denotes the normalized geodesic distance from the i^{th} vertex to the c_k^{th} vertex. c_k is the index of the center point of component k and is defined as:

$$c_k = \underset{i}{\operatorname{argmax}} \|\mathbf{C}_k^i\|_2. \quad (9)$$

c_k is updated after optimizing $\mathbf{C}$ in each training iteration. Since c_k corresponds to the maximum weight in $\mathbf{C}_k$, when the k^{th} element of $\mathbf{z}$ changes, point c_k's feature will deform the most. Thus, we use Eq. (9) to define the center point. To fit the training process of the neural network, we precompute all the geodesic distances between two vertices using [52], which are then normalized by the largest pairwise geodesic distance.

Since $\mathbf{C}^{\mathrm{T}}\mathbf{Z} = (\mathbf{C}^{\mathrm{T}}\mathbf{A})(\mathbf{A}^{-1}\mathbf{Z})$, for any invertible matrix $\mathbf{A}$, to avoid trivial solutions with arbitrarily small $\mathbf{C}$ values and arbitrary large $\mathbf{Z}$ values, we also add constraints to $\mathbf{Z}$ as a regularization term:

$$\Pi(\mathbf{Z}) = \frac{1}{K}\sum_{j=1}^{K} \max((\max_m |\mathbf{Z}_{m,j}| - \theta), 0), \quad (10)$$

where $\mathbf{Z}_{m,j}$ is the weight of the j^{th} element of the m^{th} model, and θ is a small positive number. We set $\theta = 5$ in all the experiments. We use Mean Square Error (MSE)

$$L_{recon} = \frac{1}{N}\sum_{m=1}^{N} \|\widehat{\mathbf{X}}_m - \mathbf{X}_m\|_2^2 \quad (11)$$

to urge the network to reconstruct the representation of models, where $\mathbf{X}_m$ and $\widehat{\mathbf{X}}_m$ are input and reconstruction of m^{th} model (data term), and the KL divergence

$$L_{KL} = D_{KL}(q(\mathbf{z}|\widetilde{\mathbf{f}})\|p(\mathbf{z})) \quad (12)$$

promotes the latent space to form a Gaussian distribution, where $p(\mathbf{z})$ is the Gaussian prior distribution and $q(\mathbf{z}|\widetilde{\mathbf{f}})$ is the posterior distribution given input feature $\widetilde{\mathbf{f}}$. Finally, the total loss function is:

$$\mathcal{L} = L_{recon} + \lambda_{KL}L_{KL} + \lambda_{\Omega}\Omega(\mathbf{C}) + \lambda_{\Pi}\Pi(\mathbf{Z}) + \lambda_{\Phi}\Phi(\mathbf{d}), \quad (13)$$

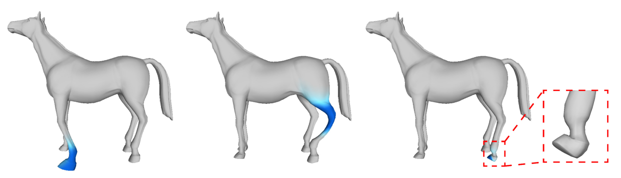
We initialize $\mathbf{d}_k = 0.2, 1 \leq k \leq K$ for all K components and set $\lambda_{KL} = 0.01, \lambda_{\Omega} = 1000$, $\lambda_{\Pi} = 1$, $\lambda_{\Phi} = 1$. We use ADAM algorithm [53] and set the learning rate to 0.001 to train the network.

5 Applications

Once trained, the network can be used to perform many useful tasks, including dimensionality reduction, reconstruction, component analysis, shape synthesis and neural shape editing. The first two applications are straightforward, so we now give details for performing the last three applications.

5.1 Component Analysis

The matrix $\mathbf{C}$ corresponds to the localized deformation components. We assume the r^{th} model is the reference model (which can be the first model in the dataset) which has a


(a) Neumann et al.

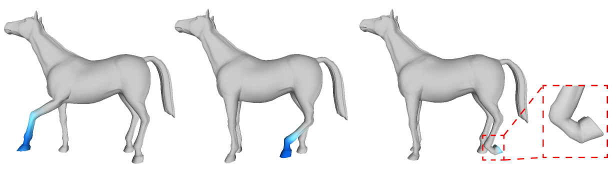
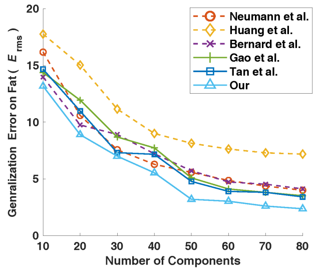
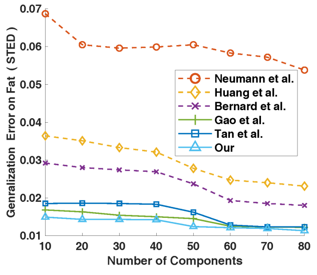

(b) Ours

Figure 3. Components of the horse dataset [46] extracted by [5] and our method. The method [5] cannot cope with large deformations and suffers from artifacts since it is designed to deal with linear deformations. Moreover, our method also produces more plausible results for small components such as hooves, thanks to the dynamic sparsity constraints.



(a)



(b)

Figure 4. Errors of applying our model to generate unseen data, using a fat person (ID: 50002) from the MPI DYNA [48] dataset. We randomly select one model from every ten models for training and remaining for testing. We evaluate metrics E_{rms} and $STED$ error with different component numbers. Our method outperforms other methods even for limited numbers of components. The methods shown with dashed lines are methods for linear deformations, and others are methods for non-linear deformations.

latent vector $\mathbf{Z}_r$. To analyze the i^{th} deformation component, we calculate the minimum and maximum values of the i^{th} element of the embedding, denoted by $\mathbf{Z}_{i_{min}} = \min_m \mathbf{Z}_{m,i}$ and $\mathbf{Z}_{i_{max}} = \max_m \mathbf{Z}_{m,i}$. We can then obtain latent vectors $\widehat{\mathbf{Z}}_{i_{min}}$ and $\widehat{\mathbf{Z}}_{i_{max}}$ corresponding to the two extreme values of the i^{th} component by replacing the i^{th} component of $\mathbf{Z}_r$ with $\mathbf{Z}_{i_{min}}$ and $\mathbf{Z}_{i_{max}}$, respectively. Applying the vectors to the decoder produces the output mesh features $\widehat{\mathbf{X}}_{min}$ and $\widehat{\mathbf{X}}_{max}$. We work out the differences $\|\widehat{\mathbf{X}}_{min} - \mathbf{X}_r\|$ and $\|\widehat{\mathbf{X}}_{max} - \mathbf{X}_r\|$ and the one that has larger distance from the reference model $\mathbf{X}_r$ is chosen as the representative shape for the i^{th} deformation component, with the corresponding latent vector denoted as $\mathbf{Z}_{i_h}$. The displacement of each vertex feature indicates the strength of the deformation, which can be visualized to highlight changed positions.

5.2 Shape Synthesis

To synthesize new models, the user can specify a synthesis weight $\mathbf{w}_{s_i}$ for the i^{th} deformation component, and the deformed shape in the latent space can be obtained as:

$$\mathbf{z}_{s_i} = \mathbf{Z}_{r,i} + (\mathbf{Z}_{i_h} - \mathbf{Z}_{r,i}) \times \mathbf{w}_{s_i}, \tag{14}$$

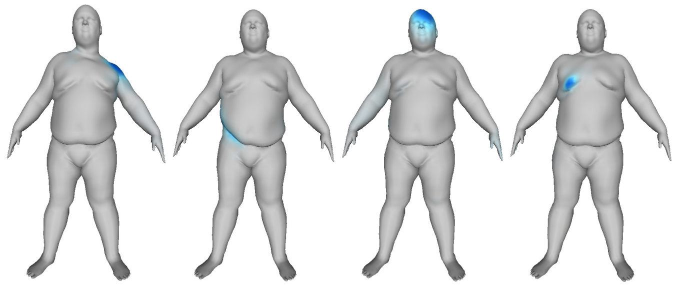
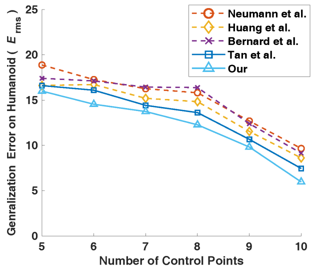
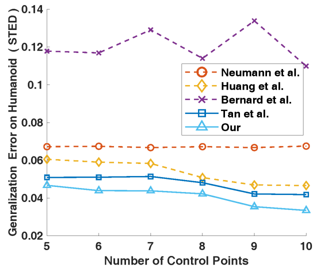


Figure 5. Components extracted by our method on the Fat dataset when K is beyond 50. Bigger K will give better reconstruction ability but also leads to extracting components of weak semantics and small deformation.



(a)



(b)

Figure 6. We use limited control points to reconstruct unseen data in the humanoid dataset, and report the generalization errors. The result shows that our method performs well, consistently with smallest errors in both metrics. The methods shown with dashed lines are methods for linear deformations, and others are methods for non-linear deformations.

where $\mathbf{z}_{s_i}$ represents the i^{th} element of obtained weight $\mathbf{z}_s$ in the latent space. Then, by feeding $\mathbf{z}_s$ in as input to the decoder, the synthesized model feature can be obtained which can be used for reconstructing the synthesized shape. The interpolation from $w_{s_i} = 0$ to $w_{s_i} = 1$ in the latent vector is an interpolation from the reference model to the model with the i^{th} deformation component. If the user wants the synthesized result to contain less characteristic of the i^{th} deformation component, he/she can decrease w_{s_i}.

Besides, we can produce plausible new shapes from a standard Gaussian sampling in the latent space because of the strong generative capability of VAE, and details are shown in Sec. 6.2.2.

5.3 Neural Shape Editing

To edit shapes, users specify a few control points on the mesh and move one or more of them to deform the shape. In this paper, we propose a novel neural shape editing method. On the one hand, the results of shape editing will avoid artifacts and side-effects with the help of local deformation components, so this provides a natural and important application for the key technique developed in the paper. On the other hand, unlike existing methods [9] that treat data-driven shape editing as two steps, separately for deformation component extraction and shape editing, our method formulates both steps in a unified neural network framework. This further improves the plausibility of the edited shapes.

Specifically, we introduce a learning-based framework for shape editing, based on our sparse deformation compo-

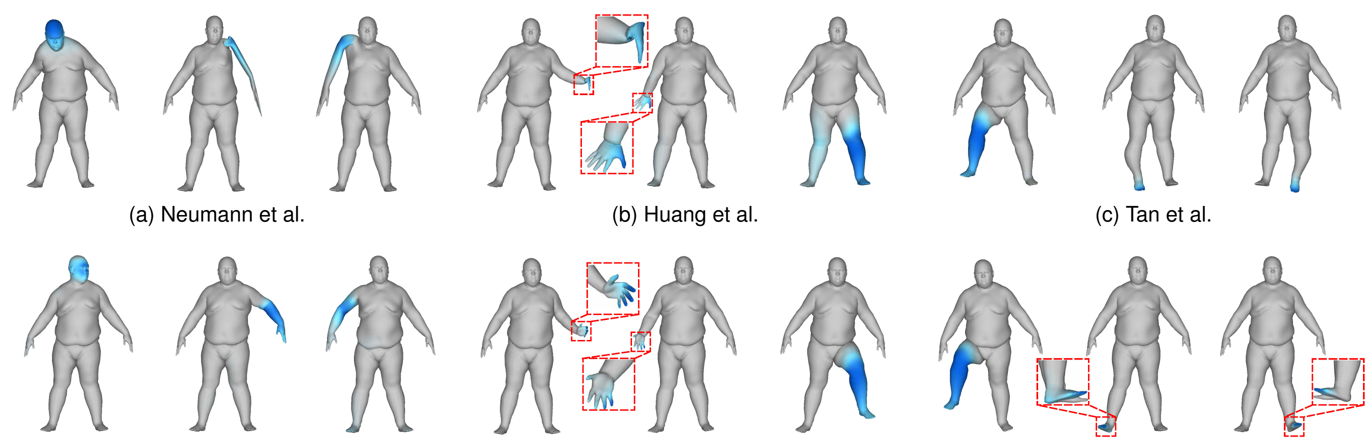


Figure 7. Comparison of deformation components extracted from a fat person (ID: 50002) in the MPI DYNA [48] dataset, using different methods including (a) Neumann et al. [5], (b) Huang et al. [6], (c) Tan et al. [10] and (d) Ours. Our method extracts more plausible components than other methods. [5] cannot extract plausible components because it focuses on linear deformation. [6] and [10] cannot cope with small components while our method produces better results because of the dynamic sparsity constraints.

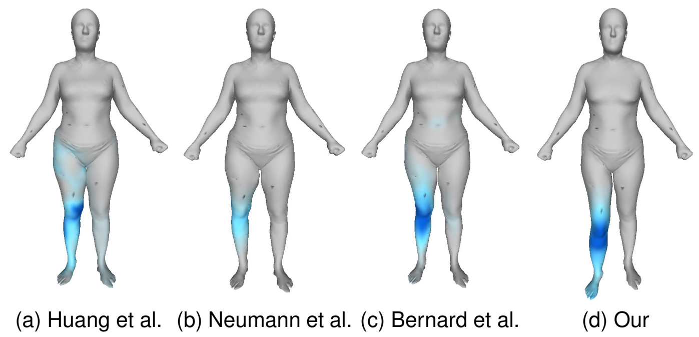


Figure 8. Comparison of extracted components on the Female Human dataset, where the component extracted by our method is similar to Bernard et al.'s method. Our methods mainly focuses on nonlinear deformation shape collections. The results illustrate that our method also copes well with linear deformation shapes collections and is able to obtain similar results as state-of-the-art method [8] on linear shape collections.

nent analysis framework. Given the deformation representation $\mathbf{X}$ of the input model, the latent vector $\mathbf{z}$ is computed through the encoder. Then, the decoded feature $\hat{\mathbf{X}}$ can be fed into a differentiable layer F that is able to reconstruct 3D vertex coordinates from feature $\hat{\mathbf{X}}$. Finally, we minimize the following editing loss L_{edit} to optimize the vertex positions to satisfy the position constraints of target control points. The loss term L_{edit} is defined as

$$L_{edit}(z) = ||M(F(Dec(\mathbf{z}))) - Q||_2^2, \tag{15}$$

where F is the differentiable layers that can reconstruct the 3D vertex coordinates from the deformation representation $\mathbf{X_0}$, Dec is the decoder of VAE and M is a function that gathers vertices corresponding to the control point set Q. Here, the layer F is implemented as follows. Given the decoded ACAP feature $\hat{\mathbf{X}}$, which includes the local rotation at each vertex, the vertex positions $\mathbf{p}'$ of the deformed mesh can be worked out by solving a least-squares problem, leading to a linear system [37], [54]: $\mathbf{L_p p' = b}$, where $\mathbf{L_p}$ is discrete Laplace-Beltrami matrix which is pre-computed, $\mathbf{p}'$ is the unknown vertex positions, $\mathbf{b}$ is the vector, which can be derived directly from $\hat{\mathbf{X}}$. The whole process of solving the linear system is differentiable and implemented using TensorFlow [55]. Please refer to Appendix C for more details about the implementation.

During the optimization, we fix the weights of the whole network, and only optimize the latent vector $\mathbf{z}$. We use ADAM algorithm and set the learning rate to 0.01. The loss converges after 1000 steps. Note that the neural shape editing is different from the sparse control ability experiments in Sec. 6.1. The latter needs to extract the components explicitly and then uses traditional deformation methods to deform the shape. In contrast our learning-based framework purely uses the neural network to deform the shape without explicitly extracting any deformation components or relying on existing deformation method. The evaluations are shown in Sec. 6.4.

6 Experimental Results

6.1 Quantitative Evaluation

In this section, We compare the generalization ability of our method with several state-of-the-art methods. These methods are divided into two types. Methods dealing with linear deformations include original SPLOCS [5], SPLOCS with deformation gradients [6], and Bernard *et al.* [8]. Methods dealing with nonlinear deformations includes Wang *et al.*'s method [7], SPLOCS with the feature from [9] as used in this paper, and our original conference version [10]. Although these linear methods are designed for linear deformation datasets, and usually produce less satisfactory results on nonlinear datasets, we compare linear and nonlinear methods both on strong nonlinear deformation datasets for the sake of completeness.

For the SCAPE dataset, we randomly choose 36 models as the training set and the remaining 35 models as the test set. After training, we compare the generalization error on the test set with different methods, using E_{rms} (root mean square) error [56]. For sequential datasets and

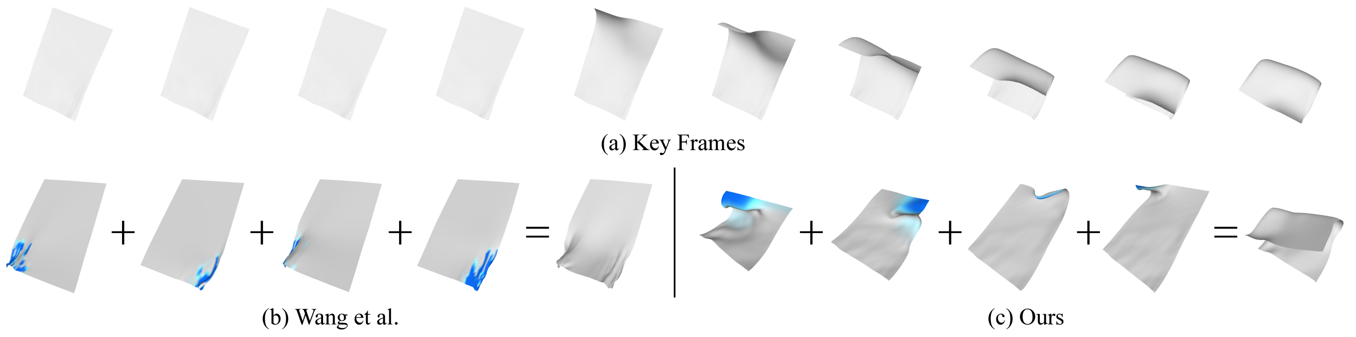


Figure 9. Top row: key frames of a flag dataset we created through physical simulation. Bottom row: synthesized models corresponding to the first four deformation components extracted by Wang et al. [7] and our method. We also present the synthesis results by combining the four components with equal weights, which show that our extracted components are more plausible.

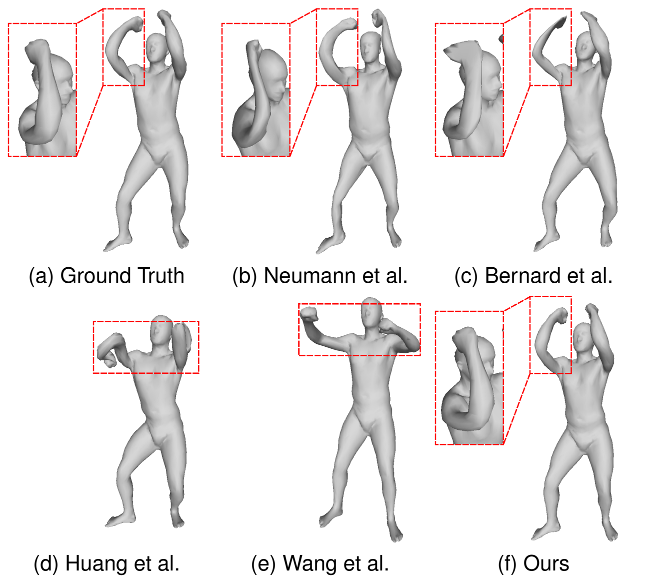


Figure 10. Visual comparison of reconstruction results of the SCAPE dataset [49]. [5], [8] and [6] are not designed to handle large-scale nonlinear deformation analysis, so it is understandable that these methods do not work well.

Female dataset [50], we randomly select one model from every ten models for training and remaining for testing due to the similarity of adjacent shapes. The sequential datasets include Swing [11], Horse [46], Face [47], Jumping [11], Humanoid, a fat person (ID: 50002) from the MPI DYNA [48] datasets. For all used dataset in our paper, we also calculate the variation of rotation angles of each vertex in regions segmented manually to represent the amount of nonlinearity. Please refer to Appendix A for more details. We compare E_{rms} error as well as $STED$ error [57], which is designed for motion sequences with a focus on 'perceptual' error of models. The results are shown in Table 1. Although for the Horse dataset [58], the method [5] has a lower E_{rms} error than our method, their method cannot cope with such dataset with large deformations and suffers from artifacts since it is designed for handling linear deformations, as shown in Fig. 3.

In Fig. 4, we evaluate quantitative performance under metric E_{rms} and $STED$ with different component numbers from 10 to 80 on the Fat person dataset. The methods shown with dashed lines are linear methods, and others are nonlinear methods. The results indicate that our method has better quantitative reconstruction results than other methods, with lower reconstruction errors when sufficient components are used. Comparison with SPLOCS using [9] demonstrates that our network is effective, beyond the benefits from the representation. Besides, we find that the increasing size of latent space is actually good for reconstruction. From Fig. 4, it shows that the descent of E_{rms} becomes slower when the number of components is beyond 50. So indeed, increasing K leads to better reconstruction. However, as the number of components increases beyond 50, the components with vague semantics and small deformation would appear, as shown in Fig. 5. Besides, 50 is the best trade-off point on these curves, which is known as the 'elbow point' in the literature [59]. In all used datasets in our paper, 50 components are enough for extracting all meaningful local components with good quantitative performance, so we choose 50 components for our experiments setting.

In Fig. 6, we compare the sparse control ability of different methods on Humanoid dataset. The methods shown with dashed lines are linear methods, and others are nonlinear methods. We randomly select a few points on the mesh and test the ability of different methods to recover the whole mesh through these limited control points. This situation is similar to the scenario that users put limited control points on significant joints to acquire models with meaningful actions. To obtain control points evenly distributed on the mesh surface, we randomly choose the first point, and then use Voronoi sampling to acquire the other points. For both methods, we choose 50 components, and for [5] and [8], we solve the reconstruction problem directly using the limited points, while for the other methods, we use data-driven deformation with the extracted components. The results show that our method performs well, consistently with smallest errors in both metrics.

In Fig. 7, we compare the deformation components extracted from a fat person (ID: 50002) in the MPI DYNA [48] dataset, using different methods including (a) Neumann et al. [5], (b) Huang et al. [6], (c) Tan et al. [10] and (d) Ours. Our method extracts more plausible components than other methods. [5] cannot extract plausible components because it focuses on linear deformation. [6] and [10] cannot cope with small components while our method produces better results

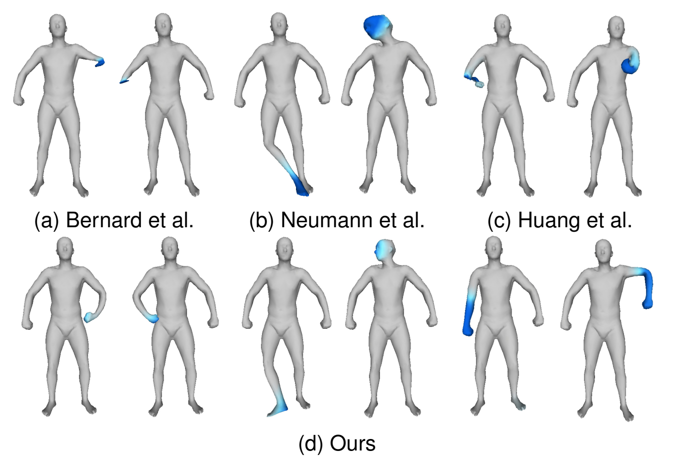


Figure 11. Comparison of deformation components located in similar areas, which are extracted by different methods including (a) Bernard et al. [8], (b) Neumann et al. [5], (c) Huang et al. [6] and (d) Ours. Our method can handle large-scale rotation better than the other linear methods without artifacts like irrational amplification and shrinkage.

because of the dynamic sparsity constraints.

In Fig. 8, we evaluate the qualitative results in a linear deformation shape collection [50] for a fair comparison with methods designed for such cases. Although our method mainly focuses on nonlinear datasets, the results show that our method also has similar performance as the state-of-the-art method [8] in the linear deformation shape set [50]. In Table 1, we evaluate the quantitative performance on this linear deformation shape set. Under the E_{rms} metric, our method has slightly improved the performance, compared to the method [8].

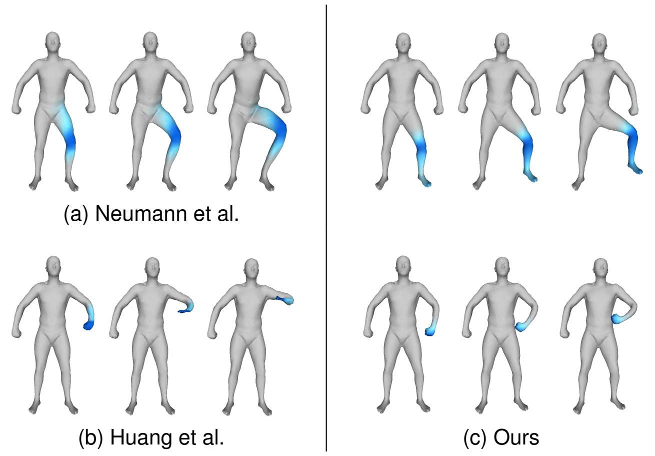


Figure 12. Synthesis results with different components of the SCAPE dataset [49]: The first row contains the components about lifting the left leg extracted by [5] (left column) and our method (right column) with weights 0.3, 0.7 and 1.1. The second row contains the components about lifting the left arm extracted by [6] (left column) and our method (right column) with weights 0.3, 0.7 and 1.1.

6.2 Qualitative Evaluation

6.2.1 Flag Dataset

To verify our method's ability to capture primary deformation components even when there is significant noise, we test on a flag dataset created by physical simulation

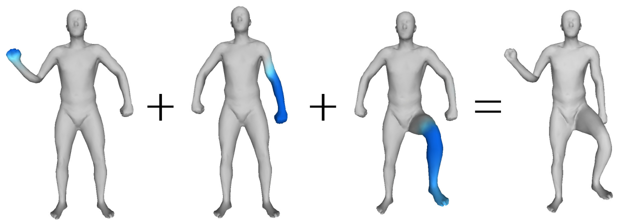


Figure 13. Synthesized models based on components derived from SCAPE dataset [49] by our method.

and compare our method with [7]. For both methods, we extract 20 components. Following the component analysis in Sec. 5.1, we sort the distance calculated between each extracted component and the reference shape in descending order and get the first four components (bottom row) along with the key frames of the dataset (top row), as shown in Fig. 9. Our method is able to extract the main movements (large-scale swinging of the flag), and separate local movements in the left and right parts of the flag. The synthesized result with the four components extracted by our method is reasonable. In contrast, [7] does not handle different scales well and only captures the noise around the corners of flags rather than the major deformation modes, and the reconstructed shape does not capture the true deformation.

6.2.2 SCAPE Dataset

Fig. 10 shows the visual comparison of reconstruction results on the SCAPE dataset. From the visualization results, we can see that these works [5], [6], [8] cannot handle large-scale rotations well and fail to reconstruct plausible models in such cases, because these methods mainly handle linear deformations, and are hard to reconstruct plausible results on such nonlinear deformation datasets, while the work [7] can be affected by noise in the dataset and cannot recover some shapes precisely. Our method does not have such drawbacks.

In Fig. 11, we compare the components extracted by our method and linear methods [5], [6], [8]. And Fig. 12 shows two groups of components about lifting the left leg (extracted by our method and [5]) and left arm (extracted by our method and [6]) with different weights. These justify that our method can handle large-scale rotation better than the other linear methods without artifacts like unreasonable bulging and shrinkage.

In addition, our proposed method also has powerful synthesis and generation ability. We show synthesis results by combining several different deformation components in Fig. 13. And in Fig. 14, we show the randomly generated new shapes using our variational autoencoder, which demonstrates that our VAE architecture can generate plausible shapes, thanks to the Gaussian distribution prior in the latent space.

6.2.3 Swing and Jumping Datasets

For Swing and Jumping datasets from [11], we align all the models and then train the network. The synthesis results with weights $0.3, 0.7, 1.1$ of our method are compared with those of [7] in Fig. 15. The first row of components are about shaking head to left from the Jumping dataset. Our method

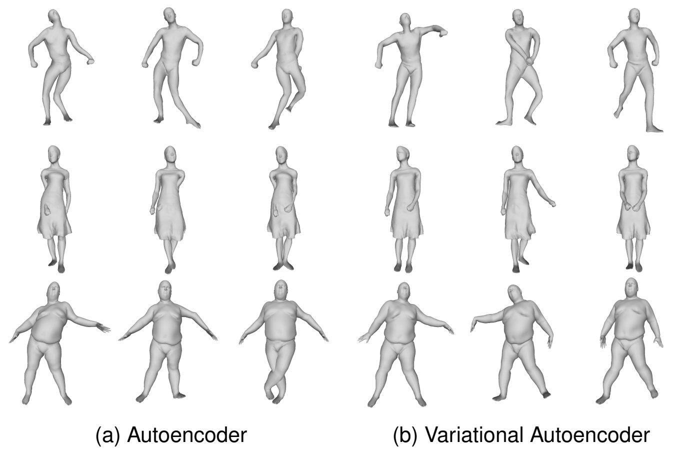


Figure 14. Comparison of randomly generated new shapes with autoencoder (a) and variational autoencoder (b) architectures on three datasets. We rrandomly sample the latent vector $\hat{\varepsilon} = \mathcal{N}(\mathbf{0}, \mathbf{I})$ with Gaussian distribution to generate new shapes by our trained decoder. It demonstrates that our VAE architecture can generate plausible shapes, thanks to the Gaussian distribution prior in the latent space.

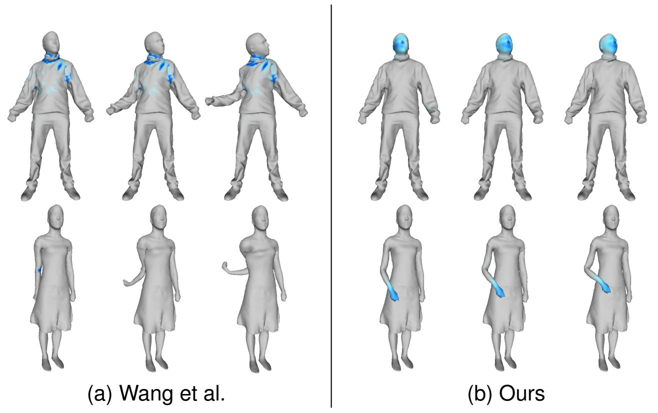


Figure 15. Synthesis results with components of Jumping and Swing datasets [11]: The first row contains the components about shaking head extracted by [7] (left column) and our method (right column) with weights 0.3, 0.7 and 1.1. The second row contains the components about lifting the right arm extracted by [7] (left column) and our method (right column) with weights 0.3, 0.7 and 1.1.

(right column) focuses on the movement of the head and can produce reasonable models, while models generated by [7] (left column) are disturbed by the clothes, and have artifacts of arm structure. The second row of models are about lifting the left arms from the Swing dataset, Wang *et al.* [7] (left column) even finds wrong direction for this movement. We also show synthesis results with equal weights for the Swing dataset by combining three different components in Fig. 1.

6.2.4 DYNA Dataset

We compare the extracted components between fixed and dynamic sparsity constraints on a fat person (ID:50002) from the MPI DYNA dataset [48]. As shown in Fig. 16, both fixed and dynamic constraints can extract reasonable components such as the leg lifting action, while the dynamic sparsity constraints are able to extract more plausible smaller components such as those around the chest and hands. Quantitatively, the E_{rms} error of dynamic sparsity constraints is 3.2062, which is smaller than 4.7795 of fixed sparsity constraints.

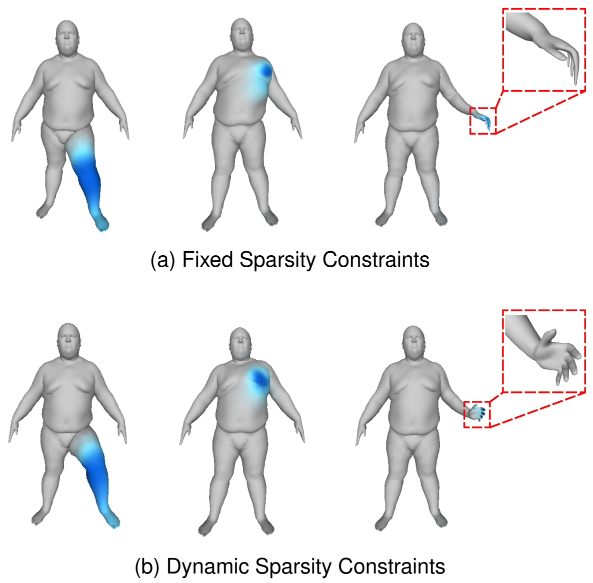

(a) Fixed Sparsity Constraints
(b) Dynamic Sparsity Constraints

Figure 16. Components of a fat person (ID: 50002) from the MPI DYNA dataset [48] extracted by our method with fixed sparsity constraints used in [10] and dynamic sparsity constraints. It shows that dynamic sparsity constraints extract more plausible small components and E_{rms} is 3.2062, which is smaller than 4.7795 of fixed constraints.

6.3 Parameter Settings and Ablation Studies

In this section, we show the ablation studies of each regularizor and explain the choice of weights of each loss term. We then compare E_{rms} of generating unseen data under different numbers of output channels of the convolution layer d'. We also evaluate the difference between spectral and spatial convolution operators, non-weighted and cotangent weighted adjacency matrix, as well as fixed and dynamic sparsity constraints. Finally, we evaluate the difference between autoencoder and variational autoencoder architecture, and the choice of reference shape.

6.3.1 The choice of regularization terms

To extract sparse deformation components, we include several regularization terms to the reconstruction loss, including $\Omega(\mathbf{C})$, $\Phi(\mathbf{d})$ and $\Pi(\mathbf{Z})$. Compared to more general L2 norm, our regularization terms have more specific impacts rather than only avoiding overfitting. $\Omega(\mathbf{C})$ urges deformation components to only capture localized deformations and avoid overfitting. Without $\Phi(\mathbf{d})$, the learned $\mathbf{d}_k$ will grow arbitrarily large, leading to the failure of sparse constraints $\Omega(\mathbf{C})$, as excessively large local neighborhoods are no longer local. $\Pi(\mathbf{Z})$ avoids arbitrarily large values of the maximum value in each component which may reduce the influence of other components, while using the *L2-norm* will constrain all element of $\mathbf{Z}$ to be small, which is not expected. There is a quantitative experiment (Table 2) to evaluate the difference between the *L2-norm* and *max*. We evaluate the reconstruction performance on the test set (SCAPE). We can see that the *L2-norm* is a strong constraint on the latent space $\mathbf{Z}$ and not friendly for network training. For a smaller weight on the $\Pi(\mathbf{Z})$, the network can converge (but it is not sufficiently effective, leading to worse performance than using *max*). On the other hand, if the weight is slightly higher, the loss on the training data is too high to converge on the training set.

Fig. 17 shows the ablation studies of each regularizor on the SCAPE dataset. (a) is the ground truth shape. Without $\Omega(\mathbf{C})$, the network gets poor generation ability because of overfitting and cannot extract local deformation components, as shown in Fig. 17 (b). Without $\Phi(\mathbf{d})$, the network also cannot extract local deformation components as shown in Fig. 17 (c) because $\mathbf{d}$ is increasing. Without $\Pi(\mathbf{Z})$, the network extracts poor components as shown in Fig. 17 (d). We also compute the E_{rms} of unseen data for these ablation studies, and they are 24.1153, 14.5911, 13.9916 and 11.8712 from (b) to (e). Without any of the regularizor, the network will get poor generation ability. In addition, Appendix B shows evaluation of the weight of each loss term.

6.3.2 The choice of d'

The number of output channels of the convolution layer d' is another adjustable parameter. We compare the E_{rms} of generating unseen data from the SCAPE dataset [49] with different d' from 3 to 15 and stride is 3, and errors are 57.0250, 20.0905, 11.8712, 18.4666 and 20.9998. And we choose $d' = 9$ in our network because this leads to the minimum E_{rms}.

6.3.3 Graph Convolution Operator and Adjacency matrix

We compare E_{rms} between different graph convolutions (spatial [10] and spectral graph convolutions).

In Table 3, we use the same autoencoder network architecture to evaluate E_{rms} of generating unseen data under different graph convolution operators and adjacency matrix formulations. It is shown that both SCAPE [49] and Swing [11] datasets get the best results using spectral graph convolution with $H = 3$ when the network uses the same adjacency matrix.

For a certain H number for the spectral domain GCN kernel, we can consider $(H-1)$-ring neighborhoods within one layer. However, we need to stack $H-1$ spatial GCN layers and introduce more parameters to achieve a similar effect. Normally, neural networks with more parameters may cause overfitting on the training set, and may not perform well on the test set. To show this problem, we compare the E_{rms} of network with different numbers of spatial CNN layers form 1 to 3 on the SCAPE and Swing datasets. As shown in Table 4, the network gets the lowest testing errors when the number of spatial CNN layers is 1, while training errors keep decreasing when we increase the number of layers. We can see a clear overfitting phenomenon when we stack more spatial GCN layers.

Meanwhile, in the original spatial version, the neighbour weight for convolution operation on point i is only adjusted by the degree of the center point i ($1/deg(v_i)$), shared by all neighbors $j \in \{1, 2, \ldots, deg(v_i)\}$. While, after adding Laplacian matrix in the spectral GCN, the neighborhood weights are adjusted by both center point's degree and neighbor's degree ($\frac{1}{\sqrt{deg(v_i)}\sqrt{deg(v_j)}}$). In this way, spectral GCN can better fit the irregular graph of the mesh. This also helps with the performance improvement in Table 3.

We also evaluate E_{rms} between different adjacency matrix formulations, including unweighted (where $w_{ij} = 0$ means there is no edge connecting vertices v_i and v_j, and $w_{ij} = 1$ otherwise) and cotangent weighted adjacency matrix (where w_{ij} is the cotangent weight of the edge). As shown in Table 3, the network using unweighted adjacency matrix produces lower errors than with cotangent weighted one, when the network uses the same graph convolution operator. This is probably for the deformed shapes, cotangent weights can only be obtained from one shape, which do not generalize well to other deformed shapes.

TABLE 2
Evaluation on *max* v.s. *L2-norm* in $\Pi(\mathbf{Z})$. We compare the E_{rms} on unseen data of SCAPE between different weights of $L2-norm$ and our original version (*max* with weight 1). In order to achieve the same performance as our original version, we adjusted the weight to a small value 0.00001. If the weight is slightly higher, the loss on the training data is too high to converge on training set. This shows that the *L2-norm* is a strong constraint to our network, which is not helpful for network training.

L2-norm/*max*	*max*	*L2-norm*				
Weight	1	1e-7	1e-5	1e-3	0.1	10
E_{rms}	11.8712	13.3312	12.8427	166.093	163.699	170.286

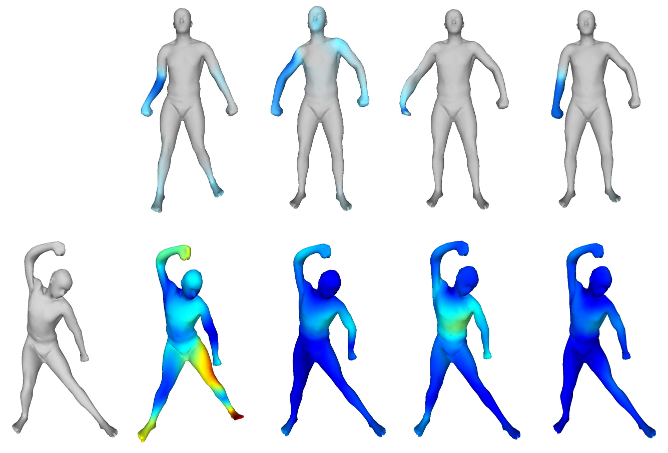

(a) Ground Truth
(b) w/o $\Omega(\mathbf{C})$
(c) w/o $\Phi(\mathbf{d})$
(d) w/o $\Pi(\mathbf{Z})$
(e) Full

Figure 17. Ablation studies of each regularizor on SCAPE dataset. The first row shows the extracted components and the second row shows a reconstruction shape in unseen data. (a) is the test shape. (b) is the result without $\Omega(\mathbf{C})$. (c) is the result without $\Phi(\mathbf{d})$. (d) is the result without $\Pi(\mathbf{Z})$. (e) is the result of full loss.

6.3.4 Fixed and Dynamic Sparsity Constraints

In Table 5, we use the same autoencoder network architecture as with the spectral convolution operator and $H = 3$ to evaluate E_{rms} with fixed and dynamic sparsity constraints. It demonstrates that dynamic sparsity constraints produce lower reconstruction error than fixed sparsity constraints used in [10].

6.3.5 Autoencoder and Variational Autoencoder

We then compare E_{rms} with autoencoder and variational autoencoder network architectures. For the SCAPE dataset, mean E_{rms} is 11.8712 of variational autoencoder architecture, which is lower than 12.0668 of autoencoder architecture. This is because the variational autoencoder forces the distribution of latent space close to standard normal

TABLE 3
Errors of applying spatial [10] and spectral graph CNNs with different adjacency matrix formulations to generate unseen data from the SCAPE [49] and Swing [11] datasets. For spectral graph CNN, we also evaluate the errors of generating unseen data with different H values. It shows that the spectral graph CNN with $H = 3$ and unweighted adjacency matrix gets the lowest error.

Dataset	Adjacency Matrix	Tan et al. [10]	Spectral CNN		
			$H = 3$	$H = 4$	$H = 5$
SCAPE	Non-weighted	13.556	**13.3964**	14.3267	15.5826
	Cotangent weight	16.505	14.4026	14.3127	16.4832
Swing	Non-weighted	14.0836	**13.7278**	16.0635	16.1074
	Cotangent weight	14.4273	14.0495	16.0616	16.1978

TABLE 4
Errors of different number of spatial layers from 1 to 3 on SCAPE and Swing dataset. It shows that the network gets the lowest error when the number of Spatial CNN layers is 1, while training errors keep decreasing when we increase the number of layers. We can see a clear overfitting phenomenon when we stack more spatial GCN layers.

Dataset	Evaluation Set	# Spatial CNN layers		
		1	2	3
SCAPE	Training Set	5.4093	4.2454	4.1737
	Test Set	**13.5560**	15.6696	18.5327
Swing	Training Set	6.6540	6.2412	4.2306
	Test Set	**14.0836**	17.9519	18.9323

distribution, thus it prevents network overfitting, making the network have better generation ability. As shown in Fig. 14, our variational autoencoder architecture can generate more plausible new shapes than using an autoencoder architecture.

In summary, we use spectral graph convolution with $H = 3$ and unweighted adjacency matrix in our network with a variational autoencoder. More quantitative results of other datasets are shown in Table 1, which demonstrate that our network architecture improves the generation ability.

6.3.6 Choice of reference shape to compute the feature

Although our feature is computed based on the reference shape, there are no requirements for the first shape (and we choose the first shape only for simplicity). To verify this, we randomly choose a shape as the reference for calculating deformation features and evaluate the E_{rms} of unseen data on the SCAPE dataset. The E_{rms} of results with randomly selected reference shapes (11.6587 and 11.9094 for E_{rms} of two different randomly selected) are similar to the original results with the first shape as reference (11.8712 for E_{rms}).

6.4 Neural Shape Editing

Fig. 18 shows the evaluations of neural shape editing. Given several control points (highlighted with green balls in (a)) and an input model (a), the latent vector is optimized according to the positions of control points with the fixed-weight network. The green balls shown in (b) to (d) are fixed points and red balls are edited points. The positions of these control points are used as constraints to optimize the latent vector. We show two examples here. The first row shows a simple editing by raising the right arm and left leg, and the second row is a more complicated editing example where all hands and legs are moved. For each example, (a) is the input model, (b) is the result of autoencoder with sparse constraint $\Omega(\mathbf{C})$, (c) is the result of variational autoencoder without sparse constraint $\Omega(\mathbf{C})$, and (d) is the result of our full method (variational autoencoder and with sparse constraint). Using the autoencoder architecture instead of variational autoencoder, some large-scale deformations, such as raising the arm over the hand, cannot be properly represented by the network as shown in (b). Without sparse constraint $\Omega(\mathbf{C})$, the weights $\mathbf{C}$ are not sparse and local, which means the network cannot extract

TABLE 5
Comparison between fixed and dynamic sparsity constraints with spectral graph CNN and $H = 3$, to generate unseen data from SCAPE [49] and Swing [11] datasets. It demonstrates that dynamic sparsity constraints get lower reconstruction errors than fixed sparsity constraints used in [10].

Dataset	Fixed Sparsity Constraints	Dynamic Sparsity Constraints
SCAPE	13.3964	**12.0668**
Swing	13.7278	**13.5102**

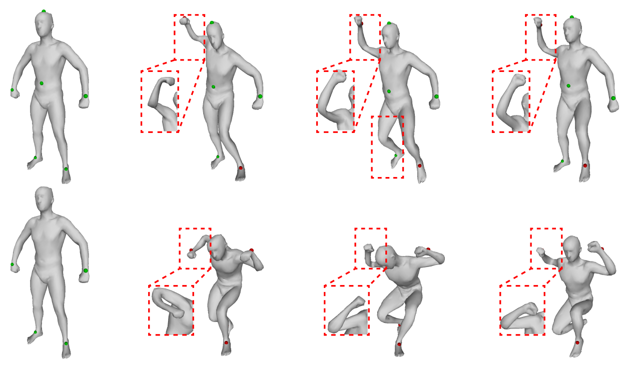


Figure 18. Evaluation of neural shape editing. Given several control points (highlighted with green balls in (a)) and an input model (a), the latent vector is optimized according to the positions of control points with the fixed-weight network. The green balls shown in (b) to (d) are fixed points and red balls are edited points. The positions of these control points are used as constraints to optimize the latent vector. We show two examples here. The first row shows a simple editing by raising the right arm and left leg, and the second row is a more complicated editing example where all hands and legs are moved. For each example, (a) is the input model, (b) is the result of autoencoder with sparse constraint $\Omega(\mathbf{C})$, (c) is the result of variational autoencoder without sparse constraint $\Omega(\mathbf{C})$, and (d) is the result of our full method (variational autoencoder and with sparse constraint). Using the autoencoder architecture instead of variational autoencoder, some large-scale deformations, such as raising the arm over the hand, cannot be properly represented by the network as shown in (b). Without sparse constraint $\Omega(\mathbf{C})$, the weights $\mathbf{C}$ are not sparse and local, which means the network cannot extract local deformation components, and the model will deform globally even if only a few local points are edited, such as the right leg in the top row of (c) is affected while only the left leg and right hand are edited, and the head in the bottom row of (c) is turned. In contrast for our full method, the variational autoencoder with sparse constraint is capable of learning a more plausible and smooth latent space, and produces appropriate local components with the help of the sparsity constraint $\Omega(\mathbf{C})$, leading to much better deformation results.

the latent vector. We show two examples here. The first row shows a simple editing by raising the right arm and left leg, and the second row is a more complicated editing example where all hands and legs are moved. For each example, (a) is the input model, (b) is the result of autoencoder with sparse constraint $\Omega(\mathbf{C})$, (c) is the result of variational autoencoder without sparse constraint $\Omega(\mathbf{C})$, and (d) is the result of our full method (variational autoencoder and with sparse constraint). Using the autoencoder architecture instead of variational autoencoder, some large-scale deformations, such as raising the arm over the hand, cannot be properly represented by the network as shown in (b). Without sparse constraint $\Omega(\mathbf{C})$, the weights $\mathbf{C}$ are not sparse and local, which means the network cannot extract local deformation components, and the model will deform globally even if only a few local points are edited, such as the right leg in the top row of (c) is affected while only the left leg and right hand are edited, and the head in the bottom row of (c) is turned. In contrast for our full method, the variational autoencoder with sparse constraint is capable of learning a more plausible and smooth latent space, and produces appropriate local components with the help of the sparsity constraint $\Omega(\mathbf{C})$, leading to much better deformation results.

7 Conclusion & Future work

In this paper, we propose a variational autoencoder based on graph convolutions in the spectral domain and a sparsity regularization term with the sparsity range automatically learned to extract localized deformation components. Extensive quantitative and qualitative evaluations show that our method is effective, outperforming state-of-the-art methods and can derive more natural deformation components. These local deformation component analysis methods depend on mesh collections with the same connectivity of triangles. This kind of datasets defines a regular domain with fixed topology, which is friendly for shape analysis. It also has the benefit of ensuring that the extracted deformation component corresponds to the same semantic region on different shapes in the dataset. Nevertheless, it has limited transfer ability and generalization ability across the shapes with different connectivity. To deal with it, it is possible to perform non-rigid registration [14] to align a template mesh to each shape in the collection to obtain meshes of the same topology. Such alignment also helps make learning more effective, thanks to a common domain. In future work, we would like to further develop methods that do not reply on the same triangle connectivity to increase the generalization ability. Another interesting direction is to employ the idea of [8] to optimize the local support region in a parameter-free manner.

Acknowledgment

This work was supported by the Newton Advanced Fellowship from the Royal Society (No. NAF\R2\192151), the Youth Innovation Promotion Association, CAS (No. 2019108), the National Natural Science Foundation of China (No. 62061136007 and No. 61872440), the Beijing Municipal Natural Science Foundation (No. L182016) and the Tencent AI Lab Rhino-Bird Focused Research Program (No. JR202024).

References

[1] L. Gao, Y. Lai, D. Liang, S. Chen, and S. Xia, "Efficient and flexible deformation representation for data-driven surface modeling," *ACM Trans. Graph.*, vol. 35, no. 5, pp. 158:1–158:17, 2016.

[2] C. Cao, D. Bradley, K. Zhou, and T. Beeler, "Real-time high-fidelity facial performance capture," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 46:1–46:9, 2015.

[3] F. Bogo, A. Kanazawa, C. Lassner, P. V. Gehler, J. Romero, and M. J. Black, "Keep it SMPL: automatic estimation of 3D human pose and shape from a single image," in *Proceedings of the 14th European Conference on Computer Vision (ECCV)*, 2016, pp. 561–578.

[4] K. Guo, D. Zou, and X. Chen, "3D mesh labeling via deep convolutional neural networks," *ACM Trans. Graph.*, vol. 35, no. 1, pp. 3:1–3:12, 2015.

[5] T. Neumann, K. Varanasi, S. Wenger, M. Wacker, M. A. Magnor, and C. Theobalt, "Sparse localized deformation components," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 179:1–179:10, 2013.

[6] Z. Huang, J. Yao, Z. Zhong, Y. Liu, and X. Guo, "Sparse localized decomposition of deformation gradients," *Comput. Graph. Forum*, vol. 33, no. 7, pp. 239–248, 2014.

[7] Y. Wang, G. Li, Z. Zeng, and H. He, "Articulated-motion-aware sparse localized decomposition," *Comput. Graph. Forum*, vol. 36, no. 8, pp. 247–259, 2017.

[8] F. Bernard, P. Gemmar, F. Hertel, J. M. Gonçalves, and J. Thunberg, "Linear shape deformation models with local support using graph-based structured matrix factorisation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5629–5638.

[9] L. Gao, Y. Lai, J. Yang, L. Zhang, S. Xia, and L. Kobbelt, "Sparse data driven mesh deformation," *IEEE Trans. Vis. Comput. Graph.*, vol. 27, no. 3, pp. 2085–2100, 2021.

[10] Q. Tan, L. Gao, Y. Lai, J. Yang, and S. Xia, "Mesh-based autoencoders for localized deformation component analysis," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2018, pp. 2452–2459.

[11] D. Vlasic, I. Baran, W. Matusik, and J. Popovic, "Articulated mesh animation from multi-view silhouettes," *ACM Trans. Graph.*, vol. 27, no. 3, p. 97, 2008.

[12] F. Bernard, J. Thunberg, P. Swoboda, and C. Theobalt, "HiPPI: Higher-order projected power iterations for scalable multi-matching," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 10 283–10 292.

[13] F. Bernard, Z. K. Suri, and C. Theobalt, "MINA: convex mixed-integer programming for non-rigid shape alignment," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 13 823–13 832.

[14] M. Zollhöfer, M. Nießner, S. Izadi, C. Rhemann, C. Zach, M. Fisher, C. Wu, A. W. Fitzgibbon, C. T. Loop, C. Theobalt, and M. Stamminger, "Real-time non-rigid reconstruction using an RGB-D camera," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 156:1–156:12, 2014.

[15] M. Alexa and W. Müller, "Representing animations by principal components," *Comput. Graph. Forum*, vol. 19, no. 3, pp. 411–418, 2000.

[16] L. Williams, "Performance-driven facial animation," in *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH'90*, 1990, pp. 235–242.

[17] L. Gao, G. Zhang, and Y. Lai, "L_p shape deformation," *Science China Information Sciences*, vol. 55, no. 5, pp. 983–993, 2012.

[18] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *Journal of Computational and Graphical Statistics*, vol. 15, no. 2, pp. 265–286, 2006.

[19] J. R. Tena, F. D. la Torre, and I. A. Matthews, "Interactive region-based linear 3D face models," *ACM Trans. Graph.*, vol. 30, no. 4, p. 76, 2011.

[20] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015, pp. 945–953.

[21] B. Shi, S. Bai, Z. Zhou, and X. Bai, "DeepPano: Deep panoramic representation for 3-D shape recognition," *IEEE Signal Process. Lett.*, vol. 22, no. 12, pp. 2339–2343, 2015.

[22] D. Maturana and S. A. Scherer, "Voxnet: A 3D convolutional neural network for real-time object recognition," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 922–928.

[23] Y. Li, H. Su, C. R. Qi, N. Fish, D. Cohen-Or, and L. J. Guibas, "Joint embeddings of shapes and images via CNN image purification," *ACM Trans. Graph.*, vol. 34, no. 6, pp. 234:1–234:12, 2015.

[24] S. Tulsiani, H. Su, L. J. Guibas, A. A. Efros, and J. Malik, "Learning shape abstractions by assembling volumetric primitives," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1466–1474.

[25] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920.

[26] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta, "Learning a predictable and generative vector representation for objects," in *Proceedings of the 14th European Conference on Computer Vision (ECCV)*, 2016, pp. 484–499.

[27] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee, "Perspective transformer nets: Learning single-view 3D object reconstruction without 3D supervision," in *Proceedings of the 29th Conference on Neural Information Processing Systems (NeurIPS)*, 2016, pp. 1696–1704.

[28] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3D-R2N2: A unified approach for single and multi-view 3D object reconstruction," in *Proceedings of the 14th European Conference on Computer Vision (ECCV)*, 2016, pp. 628–644.

[29] A. Sharma, O. Grau, and M. Fritz, "VConv-DAE: Deep volumetric shape learning without object labels," in *Proceedings of the 14th European Conference on Computer Vision (ECCV)*, 2016, pp. 236–250.

[30] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in *Proceedings of the 30th International Conference on Neural Information Processing Systems (NeurIPS)*, 2016, pp. 82–90.

[31] A. Sinha, A. Unmesh, Q. Huang, and K. Ramani, "SurfNet: Generating 3D shape surfaces using deep residual networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 791–800.

[32] J. Li, K. Xu, S. Chaudhuri, E. Yumer, H. R. Zhang, and L. J. Guibas, "GRASS: generative recursive autoencoders for shape structures," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 52:1–52:14, 2017.

[33] C. Nash and C. K. I. Williams, "The shape variational autoencoder: A deep generative model of part-segmented 3D objects," *Comput. Graph. Forum*, vol. 36, no. 5, pp. 1–12, 2017.

[34] J. J. Park, P. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 165–174.

[35] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Stoll, and C. Theobalt, "PatchNets: Patch-based generalizable deep implicit 3D shape representations," in *Proceedings of the 16th European Conference on Computer Vision (ECCV)*, 2020, pp. 293–309.

[36] K. Mo, P. Guerrero, L. Yi, H. Su, P. Wonka, N. J. Mitra, and L. J. Guibas, "StructureNet: hierarchical graph networks for 3D shape generation," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 242:1–242:19, 2019.

[37] L. Gao, J. Yang, T. Wu, Y. Yuan, H. Fu, Y. Lai, and H. Zhang, "SDM-NET: deep generative network for structured deformable mesh," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 243:1–243:15, 2019.

[38] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *2nd International Conference on Learning Representations (ICLR)*, 2014.

[39] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proceedings of the 30th International Conference on Neural Information Processing Systems (NeurIPS)*, 2016, pp. 3837–3845.

[40] ——, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proceedings of the 29th Conference on Neural Information Processing Systems (NeurIPS)*, 2016, pp. 3837–3845.

[41] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML)*, 2016, pp. 2014–2023.

[42] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Proceedings of the 28th International Conference on Neural Information Processing Systems (NeurIPS)*, 2015, pp. 2224–2232.

[43] D. Boscaini, J. Masci, E. Rodolà, M. M. Bronstein, and D. Cremers, "Anisotropic diffusion descriptors," *Comput. Graph. Forum*, vol. 35, no. 2, pp. 431–441, 2016.

[44] L. Yi, H. Su, X. Guo, and L. J. Guibas, "SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6584–6592.

[45] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber *et al.*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[46] R. W. Sumner and J. Popovic, "Deformation transfer for triangle meshes," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 399–405, 2004.

[47] L. Zhang, N. Snavely, B. Curless, and S. M. Seitz, "Spacetime faces: High resolution capture for modeling and animation," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 548:1–558:11, 2004.

[48] G. Pons-Moll, J. Romero, N. Mahmood, and M. J. Black, "Dyna: a model of dynamic human shape in motion," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 120:1–120:14, 2015.

[49] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "SCAPE: shape completion and animation of people," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 408–416, 2005.

[50] Y. Yang, Y. Yu, Y. Zhou, S. Du, J. Davis, and R. Yang, "Semantic parametric reshaping of human body models," in *Proceedings of the 2014 Second International Conference on 3D Vision (3DV)*, 2014, pp. 41–48.

[51] Y. Bengio, N. Léonard, and A. C. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[52] K. Crane, C. Weischedel, and M. Wardetzky, "Geodesics in heat: A new approach to computing distance based on heat flow," *ACM Trans. Graph.*, vol. 32, no. 5, pp. 152:1–152:11, 2013.

[53] D. P. Kingma and J. Ba, "ADAM: A method for stochastic optimization," in *3rd International Conference on Learning Representations (ICLR)*, 2015.

[54] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling," in *Proceedings of the fifth Eurographics Symposium on Geometry Processing (SGP)*, 2007, pp. 109–116.

[55] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 16*, 2016, pp. 265–283.

[56] L. Kavan, P. J. Sloan, and C. O'Sullivan, "Fast and efficient skinning of animated meshes," *Comput. Graph. Forum*, vol. 29, no. 2, pp. 327–336, 2010.

[57] L. Vása and V. Skala, "A perception correlated comparison method for dynamic meshes," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 2, pp. 220–230, 2011.

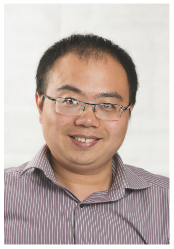
[58] R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popovic, "Mesh-based inverse kinematics," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 488–495, 2005.

[59] R. B. Cattell, "The scree test for the number of factors," *Multivariate Behavioral Research*, vol. 1, no. 2, pp. 245–276, 1966.

Qingyang Tan received the B.Eng. degree in Computer Science and Technology from University of Chinese Academy of Sciences. He is a Ph.D. student at University of Maryland, College Park. His research interests include computer graphics and geometric processing.

Ling-Xiao Zhang received his Master of Engineering's degree in Computer Technology from Chinese Academy of Sciences in 2020. He is currently an assistant engineer in the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include computer graphics and geometric processing.

Jie Yang received a bachelors degree in mathematics from Sichuan University in 2016. He is currently a PhD candidate in the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include computer graphics and geometric processing.



Yu-Kun Lai received his bachelor's degree and PhD degree in computer science from Tsinghua University in 2003 and 2008, respectively. He is currently a Professor in the School of Computer Science & Informatics, Cardiff University. His research interests include computer graphics, geometry processing, image processing and computer vision. He is on the editorial boards of *Computer Graphics Forum* and *The Visual Computer*.

Lin Gao received the bachelor's degree in mathematics from Sichuan University and the PhD degree in computer science from Tsinghua University. He is currently an Associate Professor at the Institute of Computing Technology, Chinese Academy of Sciences. He has been awarded the Newton Advanced Fellowship from the Royal Society and the AG young researcher award. His research interests include computer graphics and geometric processing.