Multiscale Mesh Deformation Component Analysis with Attention-based Autoencoders

Jie Yang, Lin Gao*, Qingyang Tan, Yi-Hua Huang, Shihong Xia and Yu-Kun Lai

Abstract—Deformation component analysis is a fundamental problem in geometry processing and shape understanding. Existing approaches mainly extract deformation components in local regions at a similar scale while deformations of real-world objects are usually distributed in a multi-scale manner. In this paper, we propose a novel method to exact multiscale deformation components automatically with a stacked attention-based autoencoder. The attention mechanism is designed to learn to softly weight multi-scale deformation components in active deformation regions, and the stacked attention-based autoencoder is learned to represent the deformation components at different scales. Quantitative and qualitative evaluations show that our method outperforms state-of-the-art methods. Furthermore, with the multiscale deformation components extracted by our method, the user can edit shapes in a coarse-to-fine fashion which facilitates effective modeling of new shapes.

Index Terms—Multi-Scale, Shape Analysis, Attention Mechanism, Sparse Regularization, Stacked Auto-Encoder

1 Introduction

With the development of 3D scanning and modeling technology, 3D mesh collections are becoming much more popular. These mesh models usually use fixed vertex connectivity with variable vertex positions to characterize different shapes. Analyzing these mesh model collections to extract meaningful components and using these components for new model generation (*e.g.* shape editing [1], [2], [3], deformation transfer [4], [5], [6]) are key research problems in these areas. Some works [7], [8], [9], [10] propose to extract deformation components from mesh data sets. They mainly focus on extracting local deformation components with sparse regularization at a uniform scale. However, real-world objects deform at multiple scales. For example, a person may have different facial expressions which are more localized deformations on the face, but the whole body can also be bent, which is a larger scale deformation.

Multi-scale techniques are getting increasingly popular in various fields. In Finite Element Methods, multiscale analysis is widely used [11], [12], [13]. In the spectral geometry field, research works apply multiscale technology on the deformation representation [14], physics-based simulation of deformable objects [15], and surface registration [16] by analyzing the non-isometric global and local (multiscale) deformation. Moreover, for shape editing, multiscale technology also enables modeling rich facial expressions on human faces [17].

Such multiscale deformation components are especially useful to support model editing from coarse level to fine level. One motivation of this work is to achieve editing consistent with perceptual semantics by modifying shapes at suitable scales. The user would be able to make rough editing of the overall shape at a large scale, as well as localized modifications to surface details at a small scale. Inspired by the recent advances in image processing with attention mechanism [18], the attention is formulated to focus on specific regions in our approach.

We propose a novel autoencoder architecture to extract multiscale local deformation components from shape deformation datasets. Our network structure is based on the mesh-based convolutional autoencoder architecture and also uses an effective representation of the shapes [19] as input which is able to encode large-scale deformations. In this work, a stacked autoencoder architecture is proposed such that the network can encode the residual value of the former autoencoder with the attention mechanism, which helps to separate the deformations into different scales and extract multiscale local deformation components. The network architecture is shown in Fig. 1. We further utilize a sparsity constraint on the parameters of the fully connected layers to keep the deformation components localized. The autoencoder architecture ensures the extracted deformation components are suitable for multiscale shape editing and helps reconstruct high quality shapes with less distortion.

Our contributions are twofolds:

- To the best of our knowledge, this is the first work that automatically extracts multiscale deformation components from a deformed shape collection. With these extracted components, the user can edit the 3D mesh shape in a coarse-to-fine fashion, which makes 3D modeling much more effective.
- To achieve this, we propose a novel deep architecture involving attentional stacked autoencoders. The attention mechanism is designed for learning the soft weights that help extract multiscale deformation components in the shape analysis and the stacked autoencoders are used to decompose the deformation of shape collections into different shape components with different scales.

All the components of the network are tightly integrated and help each other. The attention mechanism makes the follow-

* *Corresponding author is Lin Gao (gaolin@ict.ac.cn).*

- *Jie Yang, Lin Gao, Yi-Hua Huang and Shihong Xia are with the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, and also with the University of Chinese Academy of Sciences, Beijing, China. E-mail: {yangjie01, gaolin, huangyihua20g, xsh}@ict.ac.cn*
- *Qingyang Tan is with the University of Maryland, College Park, USA. E-mail: qytan@cs.umd.edu*
- *Yu-Kun Lai is with the School of Computer Science & Informatics, Cardiff University, U.K. E-mail: LaiY4@cardiff.ac.uk*

up autoencoders focus on a specific region, to allow extracting smaller-scale local deformation components. Extensive comparisons prove that our method extracts more meaningful deformation components than state-of-the-art methods.

In Sec. 2, we review the most related work. We then give a brief description of the input features used in our method in Sec. 3, and present detailed description of our novel autoencoder with the attention mechanism including implementation details in Sec. 4. Finally, we present experimental results, including extensive comparisons with state-of-the-art methods in Sec. 5 and draw conclusions in Sec. 6.

2 Related Work

Mesh deformation component analysis has attracted significant interest in the research of shape analysis and data-driven mesh deformation. Many data-driven methods for editing of either man-made objects [20], [21], [22] or general deformable surfaces [3], [19], [23], [24] benefit from extracted deformation components. Our work shares the same interest as theirs, aiming to assist users to edit shapes efficiently. Although our focus is to extract more meaningful multiscale deformation components automatically, it can be incorporated into existing data-driven deformation methods. In the following, we review work most related to ours.

3D shape deformation component extraction. With the increasing use of 3D models, the need for analyzing their intrinsic characteristics becomes mainstream. Early work [25] extracts principal components from the mesh data set by Principal Component Analysis (PCA), but the extracted components contain global deformations, which are not effective for users to make local edits. Some works [26] demonstrate that sparse constraints are effective for achieving localized deformation results. However, the classical sparse PCA [27] does not take the spatial information into consideration. By promoting sparsity in the spatial domain, many works extract localized deformation components with a sparsity constraint [7], [28], which outperform the standard PCA variants such as Clustering-PCA [29] with respect to choosing suitable compact basis modes, especially for producing more localized meaningful deformation. Moreover, the pioneering work [7] represents meshes with Euclidean coordinates, but this representation is sensitive to rigid and non-rigid transformations. Later work [8] extends the previous work [7] to better deal with rotations by using deformation gradients to represent shapes, but the method still cannot cope with larger rotations greater than 180° due to their inherent ambiguity. Based on deformation gradients, Neumann et al. [30] learn the arm-muscle deformation using a small set of intuitive parameters. The work [9] extends [7] by using a rotation invariant representation based on edge lengths and dihedral angles [31], so can handle large-scale deformations. However, the representation [31] is not suitable for extrapolation as this would result in negative edge lengths. This limits the capability of [9] for deformation component analysis, as extrapolation is often needed e.g. when utilizing the extracted deformation components for data-driven shape editing. Recent work [10] proposes a convolutional autoencoder based on an effective shape representation [19] to learn the localized deformations of a shape set, but their architecture is not suitable for extracting multiscale deformation components. Overall, different from these works [7], [8], [9], [10], our method can produce meaningful and multiscale localized deformation components. For the detailed deformation-based approaches, we refer readers to these surveys [32], [33].

Deep learning on 3D Shapes. With the development of artificial intelligence, deep learning and neural networks have made great progress in many areas, in particular 2D image processing. Hence, some researchers transform the non-uniform geometry signals defined on meshes of different topologies into a regular domain, while preserving shape information as much as possible, which enables powerful uniform Cartesian grid based CNN (Convolutional Neural Network) backbone architectures to be used on problems such as cross-domain shape retrieval [34], surface reconstruction [35] and shape completion [36]. DDSL [37] was recently proposed, which is a differentiable layer compatible with deep neural networks for learning geometric signals. However, due to the irregularity of 3D shapes, it is difficult to apply deep learning straightaway. Inspired by image processing, some works [38], [39], [40] apply deep learning to the voxel representation with regular connectivity. However, the voxel representation incurs significant computation and memory costs, which limits the resolution such methods can handle. Wang et al. [41], [42] improve the performance of voxel-based convolutions by proposing an adaptive octree structure to represent 3D shapes, and apply it to shape completion [43]. Meanwhile, recent works [44], [45] define the convolution on point clouds by using K-nearest-neighbors (KNN) and spherical convolutions. More recently, the work [46] proposed the EdgeConv operator for learning on the point cloud to improve the performance of segmentation and classification, and Guo et al. [47] use transformers [48] to learn the point cloud. In addition to the voxel and point cloud representations, shapes can also be represented as multiview projection images to perform 2D-CNNs [39], [49], [50] for 3D object recognition and classification. Such approaches are used to learn the local shape descriptors for shape segmentation and correspondence [51]. For applications that take meshes as input or generate meshes as output, turning meshes to alternative representations can lead to useful topology information to be lost. SubdivNet [52] presented a unified and flexible network architecture on 3D mesh data using the Jittor deep learning framework [53].

Alternatively, as a mesh can be represented as a graph, CNNs can be extended to graph CNNs and novel mesh pooling [54] in the spatial domain [55], [56], or spectral domain [57], [58], [59], [60] for mesh-based deep learning. For the comprehensive understanding on different 3D representation for deep learning, please refer to these surveys [61], [62], [63], [64]. In the spatial domain, works [65], [66], [67], [68] apply variational autoencoders on 3D meshes for various applications such as reconstruction, interpolation, completion and embedding. The work [69] uses autoencoders to analyze deformable solid dynamics. However, none of the existing methods can extract multiscale deformation components, which we address by using a novel attention-based mesh convolutional network architecture.

Attention mechanism on convolutional networks. Deep neural networks have proved their superiority in extracting high-level semantics and highly discriminative features on various image datasets. Researchers now pay more attention to using convolution features more effectively on fine-grained datasets to improve the performance. Such works can be widely seen in different areas of computer vision and natural language processing, such as image translation (DA-GAN) [70], person re-identification [71], [72], document classification [73], object detection [74], [75], video classification [76], etc. The work [77]

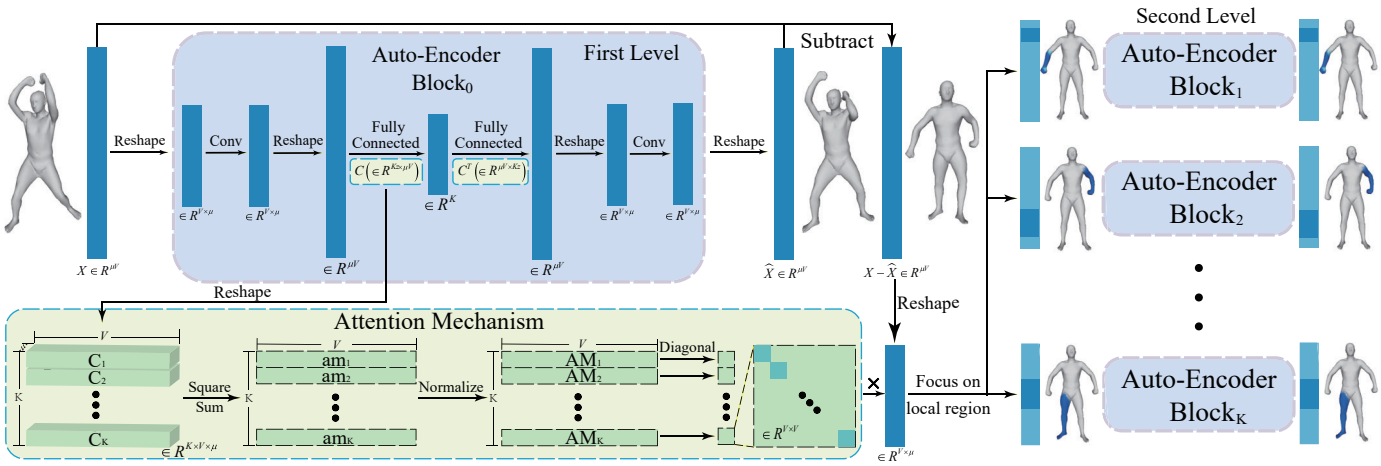


Figure 1. Our network architecture with attention mechanism and stacked autoencoders. We obtain large-scale deformation components and attention masks from the first-level autoencoder AE_0. For the second-level autoencoders $AE_k, 1 \leq k \leq K$, we put the residual value $X - \widehat{X}$ of the first-level autoencoder AE_0 with the attention masks into K autoencoders focusing on different sub-regions of the shape. We can then obtain small-scale deformation components by these autoencoders. This architecture can be further extended to include more scale levels. The autoencoder has mirrored encoder and decoder structure. The encoder has one convolution layer and a fully connected layer, and the encoder and decoder share the same trainable parameters. K, V and μ are the dimension of the latent space (the number of attention masks), the number of vertices and the dimension of the vertex features, respectively. By making the latent vector as a one-hot vector, we can extract the K attention masks $AM_k, 1 \leq k \leq K$ from the parameter C as the top-left corner of the figure shows. Please refer to Sec. 4.2 for details.

proposes a "soft attention" mechanism which predicts soft weights and computes a weighted combination of the items in machine translation. In [78], a hierarchical co-attention method is proposed to learn the conditional representation of the image given a problem. Following [78], [79] extends the co-attention model to higher orders. Some works effectively utilize attention as a way to focus on specific regions for learning. Wang et al. [80] demonstrate the benefit of guiding the feature learning by using residual attention learning for improving the recognition performance. Another example is the attention-focused CNN (RA-CNN) [18] based on the Attention Proposal Network (APN), which actively identifies the effective region and uses bi-linear interpolation to adjust the scale, and then the enlarged region of interest is used for improved fine-grained classification. However, the above works apply the attention mechanism in the 2D domain. Our work extends the attention mechanism to the 3D domain to extract multiscale deformation features based on an effective shape representation [19].

3 Deformation Representation and Convolution Operator

The input of our overall network is based on the recently proposed as-consistent-as-possible (ACAP) deformation representation [19], which can cope with large-scale deformations of shapes and is defined only on vertices, making mesh-based convolutions easier to implement. To validate it is a good choice, we compare this with a recently proposed general-purpose mesh autoencoder (AE) architecture DEMEA [67]. Please refer to Sec. 5.3.6 for the details.

For a given shape set with N shapes that share the same connectivity each with V vertices, without loss of generality, we choose the first shape as the reference shape. For the patch which consists of the i^{th} vertex and its 1-ring neighbor vertices, we can calculate its deformation gradient $\mathbf{T}_{m,i} \in \mathbb{R}^{3\times3}$. The deformation gradient of the patch is defined on the i^{th} vertex of the m^{th} shape, which describes the local deformation w.r.t. the reference shape. $\mathbf{T}_{m,i}$ of shape m is obtained by minimizing the following formula: $\arg\min_{\mathbf{T}_{m,i}} \sum_{j\in\mathcal{N}_i} c_{ij} \|(\mathbf{p}_{m,i} - \mathbf{p}_{m,j}) - \mathbf{T}_{m,i}(\mathbf{p}_{1,i} - \mathbf{p}_{1,j})\|_2^2$ where $\mathcal{N}_i$ is the 1-ring neighbor vertices of vertex i and $c_{ij} = \alpha_{ij} + \beta_{ij}$ is the cotangent weight [1], where α_{ij} and β_{ij} are the angles in the two faces that share a common edge (i, j). Then $\mathbf{T}_{m,i}$ can be decomposed as $\mathbf{T}_{m,i} = \mathbf{R}_{m,i}\mathbf{S}_{m,i}$ using the polar decomposition, where $\mathbf{S}_{m,i} \in \mathbb{R}^{3\times3}$ is a symmetry matrix that describes the scaling/shear deformation and $\mathbf{R}_{m,i} \in \mathbb{R}^{3\times3}$ is an orthogonal matrix that describes the rotation. For the rotation matrix $\mathbf{R}_{m,i}$, it can be represented by the rotation axis $\omega_{m,i}$ and rotation angle $\omega_{m,j}$. But the mapping from the rotation matrix to rotation axis and angle is one-to-many. For shapes with large-scale rotations, the rotation axis and rotation angle of adjacency vertices may become inconsistent, which results in artifacts when synthesizing new shapes, as shown in [19]. Gao et al. propose a two-step integer optimization to solve the problem which makes the rotation angle and rotation axis of adjacent vertices as consistent as possible. For the details, please refer to [19].

Next, for each vertex i on the m^{th} shape, we can obtain the feature $q_{m,i} = \{r_{m,i}, s_{m,i}\} \in \mathbb{R}^9$ by extracting non-trivial elements $r_{m,i} \in \mathbb{R}^3$ and $s_{m,i} \in \mathbb{R}^6$ from the logarithm of rotation matrix $\mathbf{R}_{m,i}$ and scaling/shear matrix $\mathbf{S}_{m,i}$ respectively. Finally, the ACAP feature of the m^{th} shape can be represented by $\{q_{m,i} | 1 \leq i \leq V\}$. Due to the use of the $tanh$ activation function [10], we further linearly scale each element in $r_{m,i}$ and $s_{m,i}$ to $[-0.95, 0.95]$ separately. Then we concatenate $q_{m,i}, 1 \leq i \leq V$ together in the vertex order to form a long vector $X_m \in \mathbb{R}^{\mu V\times1}$ as the feature of the m^{th} shape, where $\mu = 9$ is the dimension of the ACAP feature of each vertex.

We further introduce the graph convolutional operator used in our architecture. As illustrated in [10], the output of the convolution operator for every vertex is a linear combination of the inputs of the vertex and its 1-ring neighbor vertices, along

with a bias. The output $\mathbf{y}_i$ for the i^{th} vertex is defined as follows:

$$\mathbf{y}_i = \mathbf{W}_{point}\mathbf{x}_i + \mathbf{W}_{neighbor}\frac{1}{D_i}\sum_{j=1}^{D_i}\mathbf{x}_{n_{ij}} + \mathbf{b} \quad (1)$$

where $\mathbf{x}_i$ is the input feature vector of the i^{th} vertex, D_i is the degree of the i^{th} vertex, and n_{ij} $(1 \le j \le D_i)$ is the j^{th} neighbor vertex of the i^{th} vertex. $\mathbf{W}_{point}$, $\mathbf{W}_{neighbor} \in \mathbb{R}^{\mu\times\mu}$ and $\mathbf{b} \in \mathbb{R}^{\mu}$ are the trainable parameters of the graph convolutional layer. All these weights are shared by all the vertices and their neighborhoods in the same convolutional layer and learned during the training of the network.

4 METHODOLOGY

In this section, we introduce our network architecture from these three main aspects: our novel autoencoder structure, attention mechanism and redundant component removal. Firstly, we introduce the novel autoencoder structure. Then we describe our attention mechanism, which is applied to help autoencoders to extract multi-scale deformation components. Lastly, we explain how we remove redundant components, followed by implementation details of our network. This architecture is flexible to support multiple levels of scales. In most cases, two levels of deformation scales are sufficient to represent deformation in the dataset of this paper, so we mainly focus on describing the two-level architecture, which can be extended to more levels straightforwardly. Please refer to Sec. 5.3.3 for details.

4.1 Autoencoder Block

As Fig. 1 illustrates, we achieve the multiscale structure by stacking autoencoder blocks. In the first (coarsest) level, we have one autoencoder AE_0, and in the second level, K autoencoders AE_k $(k = 1, 2, \ldots, K)$ are built, each focusing on one local region through an attention mechanism, which will be detailed later. The number of second level AEs is determined by the dimension of the latent space of AE_0.

For each shape $m, 1 \le m \le N$, we represent it by the pre-processed ACAP feature $X_m \in \mathbb{R}^{V\times\mu}$, as described in Sec. 3. We use an encoder to map the feature to a 128-dimensional latent code and a decoder which reconstructs the shape ACAP feature from a latent code z. Both of the encoder and decoder have one mesh-based graph convolutional layer and a fully-connected layer and their network structure is symmetrical, where the learnable parameters of the fully connected layer are defined as $\mathbf{C} \in \mathbb{R}^{K_z\times\mu V}$, K_z is the dimension of the latent space. Especially, the fully connected layers of encoder and decoder share the same learnable parameter $\mathbf{C}$ without bias. The latent vector z for all the N shapes form a matrix $\mathbf{Z} \in \mathbb{R}^{N\times K_z}$. Similar to [10], all the layers use the $tanh$ activation function. Figure 1 illustrates the autoencoder architecture in the top left corner.

The output $\widehat{X} \in \mathbb{R}^{\mu V}$ of the whole autoencoder block can be scaled back to the ACAP deformation representation and reconstruct the Euclidean coordinates using [19]. For every autoencoder, we optimize the following loss function that includes three terms: reconstruction loss that ensures accurate reconstruction of the input, sparsity loss $\Omega(\mathbf{C})$ that promotes localized deformation components, and non-trivial regularization term $\mathcal{V}(\mathbf{Z})$ to avoid creating trivial solutions. The total loss for an autoencoder block AE_k is as follows:

$$\mathcal{L}_{AE_k} = \lambda_1 L_{recon} + \lambda_2\Omega(\mathbf{C}) + \mathcal{V}(\mathbf{Z}) \quad (2)$$

where $AE_k, 0 \le k \le K$ represent the k^{th} autoenoder, λ_1, λ_2 are the balancing weights.

The reconstruction loss is the MSE (mean square error) loss, defined as $L_{recon} = \frac{1}{N}\sum_{i=1}^{N}\|X_i - \widehat{X}_i\|_2^2$. For the non-trivial regularization term, $\mathcal{V}(\mathbf{Z}) = \frac{1}{K_z}\sum_{j=1}^{K_z}\max((\max_m |Z_{jm}| - \theta), 0)$, where Z_{jm} is the weight for the j^{th} dimension of the m^{th} shape, and θ is a positive number and we set $\theta = 5$ in our experiments.

The above two loss terms are the same as the previous work [10]. However, we define the loss $\Omega(\mathbf{C})$ differently: we choose the step function Λ (eq. 3) to map geodesic distances to $\{0, 1\}$, rather than the previously used clipped linear interpolation function. This is because our network architecture extracts hierarchical deformation components, so at any level, a fixed component size (rather than a range) is preferred. Autoencoder blocks at different levels will produce localized deformation components of different scales by adjusting the tunable parameter d. Our sparsity loss term $\Omega(\mathbf{C})$ is defined as: $\Omega(\mathbf{C}) = \frac{1}{K_z}\sum_{k=1}^{K_z}\sum_{i=1}^{V}\Lambda_{ik}\|C_{k,i}\|_2$, where $C_{k,i}$ is the μ-dimensional vector of component k of vertex i, $\|\cdot\|_2$ is group sparsity ($\ell_{2,1}$ norm) , and Λ_{ik} is sparsity regularization parameters defined as follows:

$$\Lambda_{ik} = \begin{cases} 0 & d_{ik} < d \\ 1 & d_{ik} \ge d \end{cases} \quad (3)$$

where the Λ_{ik} is a binary function, where d_{ik} denotes the normalized geodesic distance [81] from vertex i to the center point c_k of component k, which is defined as $c_k = \text{argmax}_i \|C_{k,i}\|_2$, and is updated in each iteration of network training. d is a tunable parameter, which controls the size of the deformation region of a component. Larger d corresponds to bigger deformed regions of the shape. For our task, AEs of different levels choose different values of d. Please refer to Sec. 4.4 for the default value of d.

We train the whole network end-to-end by adding all the losses of autoencoder blocks together as $L_{total} = \sum_{k=0}^{K}\mathcal{L}_{AE_k}$, which includes a first-level AE (AE_0) and K second-level AEs (AE_k, $1 \le k \le K$).

4.2 Attention Mechanism for Multiscale Analysis

Similar to 2D images, there are many tasks such as image recognition that benefit from focusing on different levels of images by an attention mechanism, e.g. [18]. In this work, we aim to analyze 3D shape deformation in a multiscale fashion, by designing an attention mechanism that facilitates our autoencoder blocks to extract multiscale deformation components.

Mostly, the deformation datasets (e.g. human, horse and fabric) have both global scale and local scale deformations. Hence, it is naturally to extract these deformations in a multiscale manner. To make the second-level AEs focus on sub-regions to extract finer-level components and then form a multiscale structure, we extract learnable attention masks from the fully-connected layer of the first-level AE AE_0. Our attention mechanism is shown in the bottom left corner of Fig. 1. Due to the sparsity constraint $\Omega(\mathbf{C})$, the parameter $\mathbf{C}$ of the fully connected layer represents the sparse deformation components. $C_k \in \mathbb{R}^{V\mu\times 1}(1 \le k \le K)$ corresponding to each row in $\mathbf{C}$ represents a deformed sub-region of the shape. The deformed sub-regions can be regarded as the interested mask of the second-level AEs. So in every iteration of training, we can extract each row of $\mathbf{C}$ by setting the latent vector to a one-hot vector for second-level AEs:

$$C_k = \mathbf{C}^T \times OH_k \tag{4}$$

where OH_k is a K-dim column vector, with the k^{th} entry set to 1 and the rest set to 0. Then we reshape C_k to a 2D array with the size $V \times \mu$, denoted as C_k^r. The unnormalized attention mask $am_{k,i} \in \mathbb{R}^{K \times V}$ for the k^{th} component of the i^{th} vertex is defined as

$$am_{k,i} = \sum_{j=1}^{\mu} {C_{k,ij}^r}^2. \tag{5}$$

where the $C_{k,ij}^r$ is the (i,j) entry of C_k^r. We further normalize it to obtain the normalized attention mask $AM \in \mathbb{R}^{K \times V}$, where

$$AM_{k,i} = \frac{am_{k,i}}{\sum_{k=1}^{K} am_{k,i}}. \tag{6}$$

So for the first-level autoencoder AE_0, the residual value of the reconstruction is $X - \widehat{X}$ and the normalized attention mask is AM. We reshape $(X - \widehat{X})$ to a 2D array $X_{res} \in \mathbb{R}^{V \times \mu}$. For the second-level autoencoder $AE_k, 1 \leq k \leq K$ as in Fig. 1, its input is $diag(AM_k) \times X_{res}$, where $diag(\cdot)$ returns a square diagonal matrix with the elements of the vector on the main diagonal. The input of each second-level AE is therefore the weighted (by the corresponding attention mask) residual of the first-level AE. Therefore, this attention mechanism ensures that the second-level AEs can reconstruct smaller scale deformations that cannot be well captured by AE_0, and each AE_k focuses on an individual local region. The sum of every column of AM is one according to Eq. 6, which ensures the sum of inputs to $AE_k, 1 \leq k \leq K$ equals the residual value X_{res} of the first-level autoencoder AE_0.

Under the supervision of loss function and attention mechanism, the first-level autoencoder AE_0 is capable of capturing large-scale deformation and the second-level AEs can capture smaller-scale deformations in specific regions of the shape. Consequently, our network can learn multiscale deformation components of the whole shape set, and the multiscale deformation components can be extracted from the parameters of fully connected layers of all AEs.

4.3 Redundant Component Removal

For all autoencoder blocks, we extract a fixed number of deformation components for each AE. For fair comparison and to capture all deformations, we set $K_z = 10$ for AE_0, $K_z = 5$ for $AE_k, 1 \leq k \leq K$. Since the multiscale analysis and our setting aim to extract deformation components as much as possible to avoid missing any components, it is possible that the second-level AEs may contain some redundant components, which do not correspond to significant deformations, and are often caused by subtle deformations compared to the reference mesh. To address this, we remove these components if the contained information is lower than the given threshold. Fig. 5 shows the results of the network output without this process, and there are some components corresponding to slight deformations compared to the reference mesh). In comparison, Fig. 11 shows that a small subset of components retained after the redundant component removal is sufficient to capture the meaningful deformations on the whole dataset. Thus, all results in our paper are processed by the redundant component removal, and the process is explained in detail as follows.

The process aims to make our results more compact and reasonable and is done after network training. It can gain trade off between the multiscale decomposition and avoidance of overfitting on training data. We define the following deformation strength of a deformation component to filter subtle or noisy deformation components. The strength $I(X_m)$ on the features X_m is defined as:

$$I(X_m) = \frac{\sum_{i=1}^{V} 1(\|(X_{diff})_i\| > \epsilon_1)\|(X_{diff})_i\|}{\sum_{i=1}^{V} 1(\|(X_{diff})_i\| > \epsilon_1)} \tag{7}$$

where $X_{diff} = X_m - X_r$, and $X_r, X_m \in \mathbb{R}^{V \times \mu}$ are the features of the reference mesh and the extracted deformation components from the autoencoders respectively. $\|\cdot\|$ is the ℓ_2 norm of the vector, and $1(\cdot)$ gives 1 if the condition is true, and 0 otherwise. $\epsilon_1 = 1e-6$.

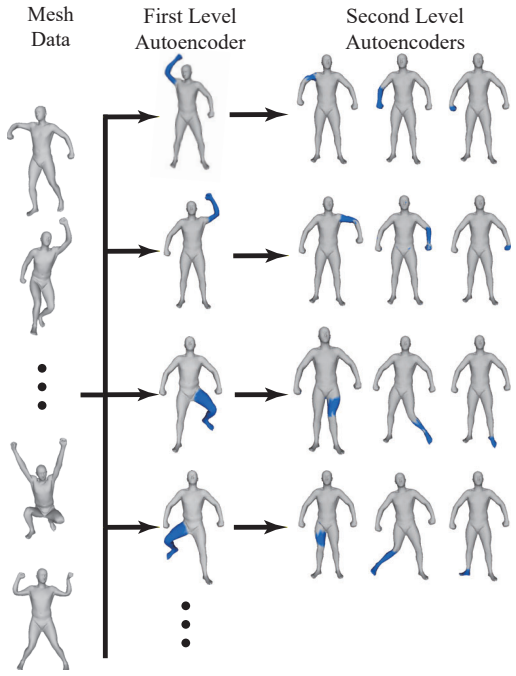


Figure 2. The multiscale structure of deformation components on the shape set SCAPE [82]. In the figure, we filter the redundant components. As a result, our method can learn deformation components of different scales. The first column shows some examples from the SCAPE dataset, the second column presents coarse level deformation components from the first-level AE_0, and the third column gives the fine level deformation components from the second-level $AE_k, 1 \leq k \leq K$.

If the extracted deformation component corresponds to a slight deformation, defined as its strength being smaller than a threshold ϵ_2, we will remove the component. In our experiments, we set $\epsilon_2 = 0.01$. Finally, we obtain the multiscale structure of the deformation components, as shown in Figs. 2, 9, 10, 11, 12 and 13, where deformations at different scales are indicated with arrows.

We also check if our network produces similar components (near duplicates). To achieve this, we test the similarity of the extracted deformation components. Fig. 3 visualizes the cosine similarity matrix of the components extracted from the first-level AE. It shows that the components have low similarity, because our AE applies the localization constraint and reconstruction error minimization to ensure different components in the latent space represent different parts on the shape; having duplicated components would lead to reduced representation capability, so higher total loss.

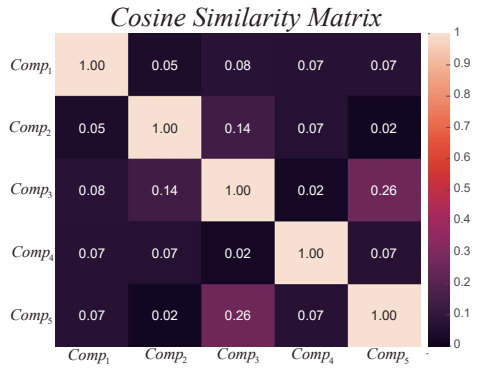


Figure 3. Visualization of the cosine similarity matrix of the components extracted from the first-level AE. It shows that the components have low similarity. The value (0-1) in each grid indicates the similarity between two components. Larger values mean more similar.

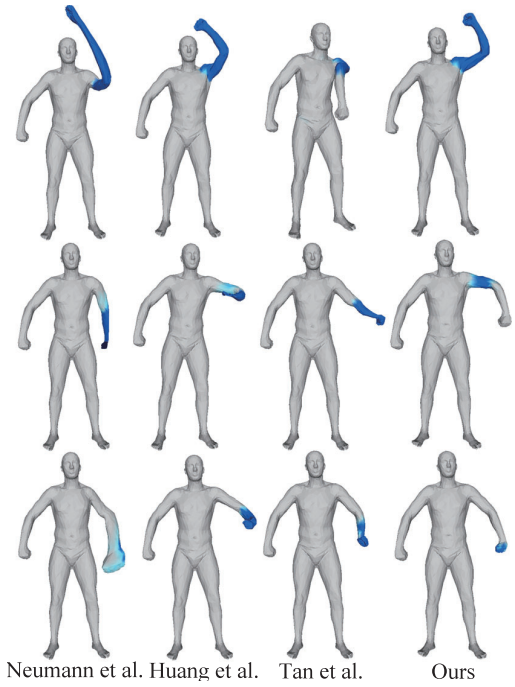


Figure 4. Comparison of deformation components located in the left arm of SCAPE [82], which are extracted by different methods. The deformed region is highlighted in blue. Every row shows the components located in a similar region. It shows that our results are more reasonable.

4.4 Implementation Details

Our experiments were carried out on a computer with an i7-6850K CPU, 16GB RAM and an Nvidia GTX 1080Ti GPU.

Datasets: We compare with the state-of-the-art methods on the SCAPE dataset [82], Horse dataset [4], Face dataset [83], Humanoid dataset [9], Dress dataset, Pants dataset [84], Flag dataset, Skirt Dataset, Fat person (ID:50002) from the Dyna [85] Dataset, Coma [66] Dataset, Swing and Jumping datasets [86]. The Dress, Flag and Skirt datasets were synthesized by the NVIDIA Clothing Tools on 3ds MAX. Our used data ranges from rigged deformations like human motions to non-rigged deformations such as faces and cloth, from small datasets with several hundreds shapes to large datasets containing thousands of shapes (e.g., Dyna). For the above datasets, they are mainly sequence models which have similar deformation between neighboring shapes. So for testing our model's generalizablity, we select one from every ten models to training and the rest to testing, i.e., the ratio is 9:1 for test and training. For non-sequence data like SCAPE, we spilt the training set and test set randomly with a ratio 1:1. As a special case for the Coma dataset, we use the same setting as DEMEA [67] for fair comparison. The statistics of the datasets are shown in Table 1, which lists the number of shapes each dataset contains, as well as the numbers of training examples and testing examples. All datasets are very easy to obtain: some are public data, and others are synthesized data using professional software. We will release the synthesized data for future research.

Table 1
Data Statistics. We summarize the data statistics of 10 datasets in our experiments. Each sequence dataset is split into training set and test set with a ratio of 1:9, with the exception of the Coma [66] dataset, where we use the same setting as DEMEA [67] for fair comparison.

DataSet	# All Shapes	# Training Shapes	#Testing Shapes
Swing	280	28	252
Scape	71	36	35
Pants	241	25	216
Humanoid	154	16	138
Horse	49	5	44
Flag	500	50	450
Dress	500	50	450
Jump	150	15	135
Fat	4737	469	4274
Face	385	39	346
Skirt	231	23	208
Coma	20465	17794	2671

In our experiments, our network takes the ACAP features of 3D shapes as input, which can describe the large scale deformations, and are calculated using the method in [19]. We have two levels of autoencoders, the first level only has one AE (AE_0), and the second-level has the same number of AEs ($AE_k, 1 \leq k \leq K$) as the number of attention masks. Since in most wild datasets, the deformation of shapes is not very exaggerated, two levels of autoencoders are enough to extract the multiscale localized deformation components in our experiments, but this can be easily extended if necessary. We perform experiments on the SCAPE dataset to demonstrate how we choose the suitable hyperparameters in Sec. 5.3, and the same fixed hyperparameters are used for different categories above. As shown in that section, our stacked AEs have the lowest error with the following default parameters: $\lambda_1 = 10.0, \lambda_2 = 1.0$, where λ_1, λ_2 are the weights of reconstruction error and sparsity constraint terms, respectively. $d = [d_1, d_2] = [0.4, 0.2], K_z = [10, 5]$, corresponding to the coarse and fine levels. Here, we train the whole network end-to-end rather than separately. We set the learning rate as 0.001 with the exponential decay by the ADAM solver [87] to train the network end-to-end until it converges, which takes approximately $10,000$ epochs. For all AEs, we set the batch size as 256, which is randomly sampled from the training data set. For a typical dataset, the training of stacked AEs takes about 10 hours. Once our network is trained, the extracted components are produced efficiently: outputting one component only takes about 50 milliseconds.

As we will later discuss, compared with separate training in Table 4, training the network end-to-end can result in smaller reconstruction errors E_{rms} on all datasets. The main reason is that the network can adjust the attention mask by minimizing the

Table 2
Errors of applying our method to generate unseen data from Horse [4], Face [83], Humanoid [9], Pants [84], Flag dataset, Fat person (ID:50002 from the Dyna dataset [85]), Synthetic Dress/Skirt dataset, CoMA [66] dataset, Swing and Jumping [86] datasets. From the table, the generation ability of our network is better than the other methods on the E_{rms} error and $STED$ error.

Dataset	Metric	Method					
		Ours	Tan et al.	Wang et al.	Huang et al.	Neumann et al.	Bernard et al.
Horse	E_{rms}	**6.9246**	12.9605	29.6090	18.0624	7.3682	20.1994
	$STED$	**0.0336**	0.04004	0.04332	0.05273	0.08074	0.4111
Face	E_{rms}	**1.4409**	2.9083	8.5620	12.3221	2.9106	2.9853
	$STED$	**0.0071**	0.007344	0.01320	0.01827	0.008611	0.02662
Jumping	E_{rms}	**16.3475**	24.4827	44.3362	37.9915	29.3368	49.9374
	$STED$	**0.0321**	0.04862	0.05400	0.06305	0.1268	0.4308
Humanoid	E_{rms}	**3.2127**	3.4912	60.9925	16.1995	14.3610	6.6320
	$STED$	0.0226	**0.01313**	0.03757	0.02247	0.07319	0.04612
Swing	E_{rms}	**12.2615**	14.0836	29.5329	24.495	15.1942	22.6571
	$STED$	**0.0311**	0.03789	0.04224	0.04343	0.0830	0.1139
Pants	E_{rms}	**6.4083**	7.8986	39.2946	10.1880	28.4118	23.6785
	$STED$	**0.0372**	0.0414	0.0540	0.04958	0.1762	0.06484
Dress	E_{rms}	**11.5117**	34.0579	35.2816	35.2340	55.5806	12.2239
	$STED$	**0.0397**	0.0415	0.0635	0.0782	0.0784	0.2167
Fat	E_{rms}	**4.3456**	4.5609	25.9187	5.3215	7.4522	5.3348
	$STED$	0.0052	0.0053	0.0055	**0.0035**	0.0372	0.0289
Flag	E_{rms}	**20.0627**	26.3174	62.2925	51.2551	23.1364	23.1535
	$STED$	**0.0157**	0.0176	0.0354	0.2183	0.0914	0.0169
Skirt	E_{rms}	**3.9863**	13.0794	27.2709	28.2209	5.6254	5.0889
	$STED$	**0.0291**	0.0422	0.0342	0.0332	0.0491	0.0601
CoMA [66]	E_{rms}	**0.6791**	2.2063	7.1492	2.3796	2.4172	1.8023
	$STED$	**0.0403**	0.0430	0.0687	0.0447	0.0441	0.1537

loss function, and conversely, the adjusted attention mask will result in smaller reconstruction errors E_{rms}. Such collaborative optimization leads to better results, as shown in Table 4.

5 Experimental Results & Evaluation

In this section, we will evaluate our method on the above datasets from the following aspects: Quantitative Evaluation, Qualitative Evaluation and Applications.

5.1 Quantitative Evaluation

We compare the generation ability of our method with the state-of-the-art methods [7], [8], [9], [10], [28] on various datasets. In this experiment, we select one model from every ten models for training and the remaining for testing. After training, we align all the models and scale them to fit within a unit ball. Then we use E_{rms} (root mean square) error [88] and $STED$ error [89] to compare the generalization error on the test data (i.e., the reconstruction error for unseen data) with the various methods. In particular, $STED$ error is designed for motion sequences with a focus on 'perceptual' error of models. To ensure fairness, we train each autoencoder to extract 50 components. As Table 2 shows, the performance of our method is better than the existing methods on both E_{rms} and $STED$. Because the Euclidean coordinate representation is sensitive to rotation, the extracted deformation components of the methods [7], [28] have clear artifacts and implausible deformation, leading to larger reconstruction errors. Due to the limitation of the edge lengths and dihedral angle representation for extrapolation, the reconstruction using the method [9] can also be inaccurate and unstable. The method [8] is not capable of encoding large-scale deformation (e.g. folds on the fabric), so it cannot recover the original deformation accurately in such cases. The method [10] uses a large-scale deformation representation to achieve good performance, but it cannot produce multiscale deformation components. In comparison, our method can keep lower reconstruction errors by using stacked AEs and analyzing the residual value of the first-level AE to extract effective multiscale deformation components.

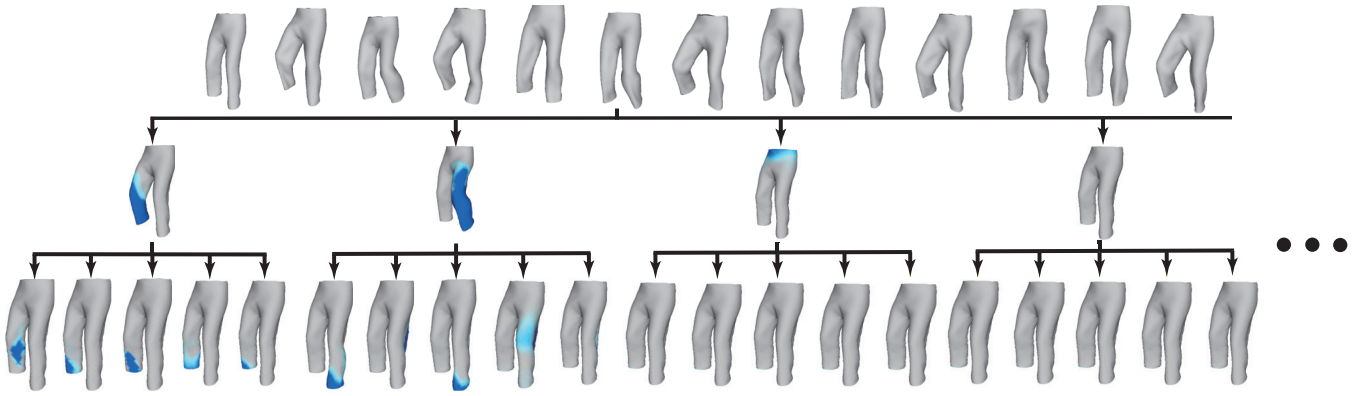


Figure 5. The multiscale structure of deformation components on the Pants dataset [84]. In the figure, to demonstrate the necessity of redundant component removal, we do not filter the redundant components, and the symbol '$\cdots$' indicates that there are more results which contain slight deformation. In comparison, results shown in Fig. 11 have been processed by the redundant component removal in Sec. 4.3. As a result, our method can learn meaningful deformation components of different scales, and a compact set of deformation components can be extracted.

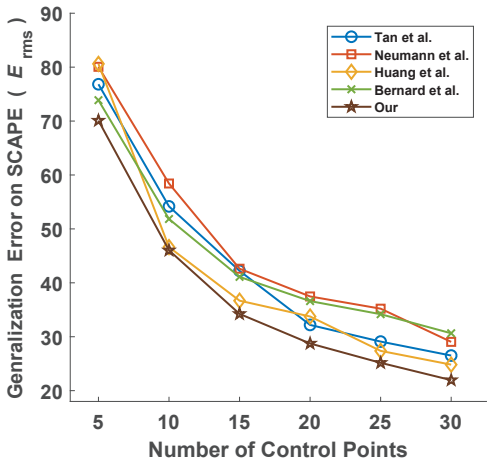


Figure 6. The reconstruction error using sparse control points to deform the SCAPE data set [82]. The control points are obtained by furthest point sampling. The generalization error is measured by the data-driven deformation with the extracted deformation components of various methods. In this figure, our performance is better than the other methods with lower errors.

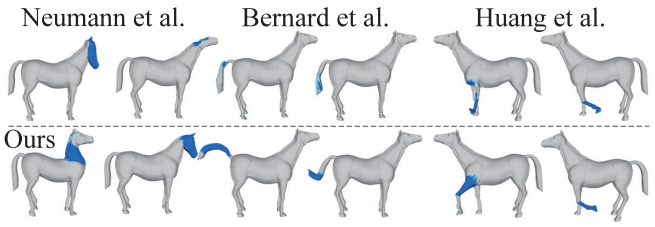


Figure 7. Comparison of deformation components located in a similar area on the Horse [92], which are extracted by different methods. The deformed regions are highlighted in blue. The first row shows the results of other methods, and the second row gives the results of our method. Every column shows a component located in a similar region.

Meanwhile, our extracted components are served for data-driven deformation. In the real world, the user usually edits the shape by a limited number of control points. To demonstrate the ability of each method to reconstruct deformed models by limited control points, we use the furthest point sampling [90], [91] to sample the control points to ensure that the sampled points distribute on the shape evenly. Under the constraint of the control points, we use the same number of the extracted components to perform data-driven deformation on the SCAPE dataset. As shown in Fig. 6, the reconstruction by our extracted components always keeps a lower error. In comparison, due to the use of Euclidean coordinate representation, the methods [7], [28] fail to reconstruct shapes accurately. Our multiscale deformation components better characterize the deformation of the shape, leading to reduced errors.

5.2 Qualitative Evaluation

We also provide qualitative evaluation of extracted multiscale deformation components, by comparing the visualization results of our method with the other methods. In our experiments, we use two levels of autoencoders. For each level, we extract deformation components of the corresponding scale from the fully connected layer using the same method as [10]. For every autoencoder, we can extract the same number of components as the dimension of the latent space. Due to our setting ($K_z = [10, 5]$), the first level and second level autoencoders each output 10 and 5 deformation components respectively. We apply the post-processing step to remove redundant components, as described in Sec. 4.3. For qualitative evaluation, we visualize retained extracted components and further color the deformed regions compared with the reference mesh.

As independently extracted deformation components do not correspond to each other, we adopt the visualization method in [8], [9], [10], and manually select two components with similar deformation areas as much as possible. Figs. 4 and 7 show the comparison with various methods on the SCAPE and Horse datasets. We show the corresponding deformation components in a similar area on the shape (left arm of the SCAPE and every major part of the horse). Our results are more meaningful, capturing major deformation modes in a multiscale manner.

We then verify if our extracted components are more semantically meaningful than other methods by a user study. We test it on three datasets (SCAPE, Horse and Pants) and compare our method with five state-of-the-art methods [7], [8], [9], [10], [28]. We adopt a scoring method to determine whether a component has a certain semantics meaningfulness. The scoring is based on

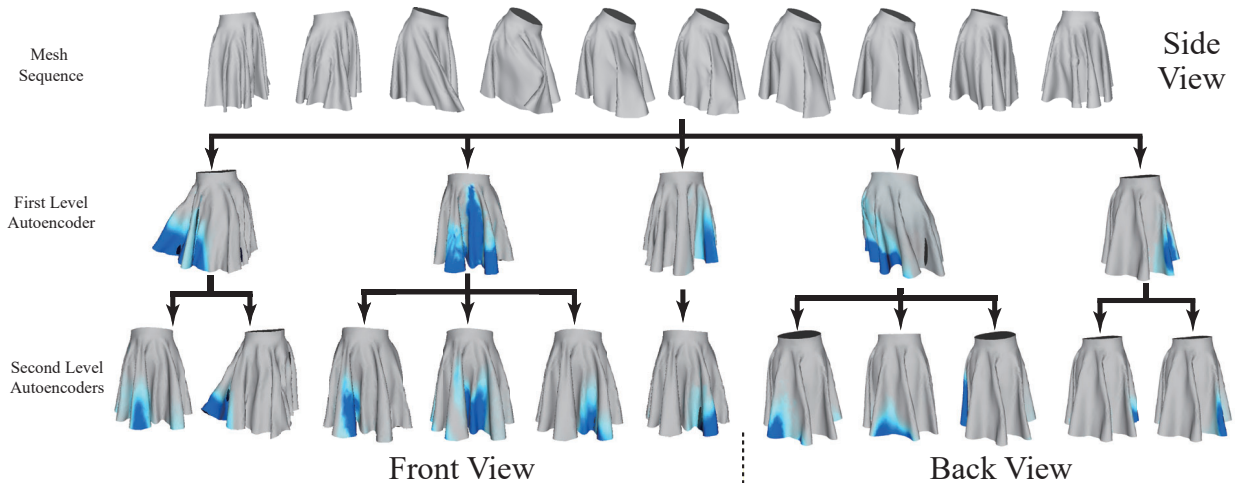


Figure 8. Multiscale deformation components on the skirt cloth dataset, extracted by our method with redundant component removal. The dataset is obtained by simulation using a physics engine, which contains challenging cloth folds and complex motions. Some samples of the skirt dataset are shown in the first row. In the second and third rows, deformation components extracted by the first-level AE and second-level AEs are visualized. The results show that our method can learn to extract deformation components of different scales with the multiscale structure.

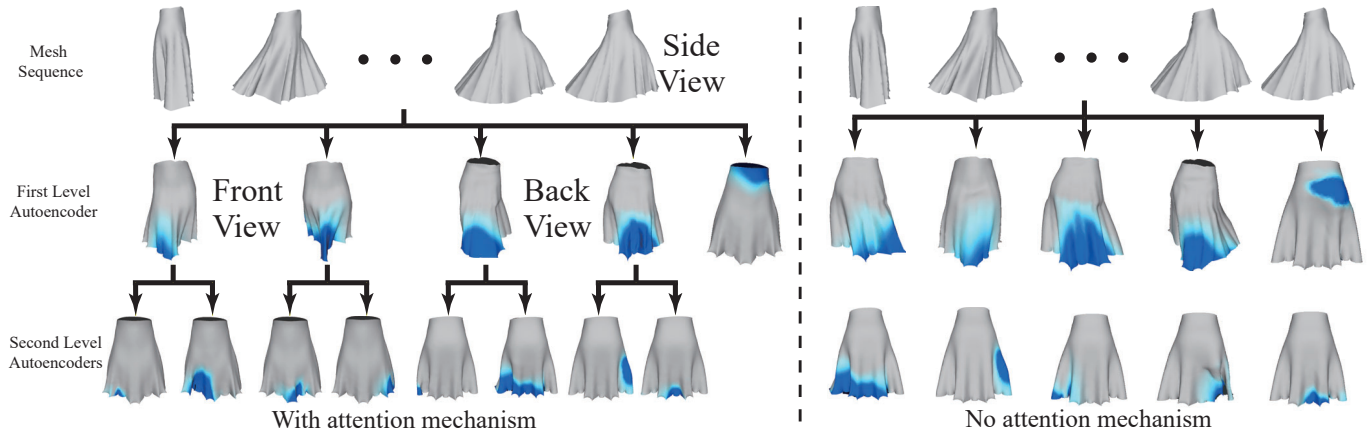


Figure 9. Multiscale deformation components on the Dress dataset, extracted by our method with redundant component removal. The dataset is obtained by simulating a lady walking forward in a skirt using a physics engine. In this figure, we compare the results with (left) and without (right) the attention mechanism. The first row shows some examples in the dress dataset from the side view. For the results on the left with attention mechanism, the second row visualizes the deformation components extracted by the first-level AE (AE_0), which correspond well to major deformations caused by the leg movement and wind, where the two results on the left are the deformations of the front of the skirt, and the two results on the right are the deformation of the back of the skirt. The third row shows the detailed deformations of the cloth during the movement, which are extracted by the second-level AEs (AE_k, $1 \leq k \leq K$). In contrast, the results on the right without attention are similarly extracted by two levels of AEs, but no longer have a multiscale structure.

the participants perception of semantic appropriateness of each component. For each data set, we first render the components extracted by all methods into the same rendering style and mix them together. We let the users browse all the pictures to have an idea of distribution first, and then let the users score in the range from 0 to 100 for each rendered image using a slider. 10 participants were involved in the user study. Then we work out the average score of each method by every participant as shown in Table 3. Our method receives highest scores in all the datasets.

In summary, compared with existing methods, our method can extract plausible and reasonable localized deformation components with semantic meanings, while the other methods have some distortions and cannot extract multiscale deformation components. Note that our extracted components correspond to *deformations*, rather than semantic *parts*, so it is natural that they are not always aligned with semantic segmentation, but instead aligned at the motion sequence level, as observed also in previous works [7], [8], [9], [10], [28].

Table 3
The user study that verifies the semantic meaning of deformation components extracted by various methods. We ask 10 participants and report their average scores of each method on the three datasets.

DataSet	Neumann et al.	Bernard et al.	Wang et al.	Huang et al.	Tan et al.	Our
Horse	46.69	30.37	48.37	56.28	52.41	**68.00**
SCAPE	33.87	40.87	55.29	48.13	61.33	**72.54**
Pants	36.37	55.75	23.00	50.50	46.00	**73.37**

In addition, we show more visualization results of multiscale deformation components on the following datasets: Swing [86], a fat person (ID: 50002) from Dyna [85], Flag, Dress and Skirt.

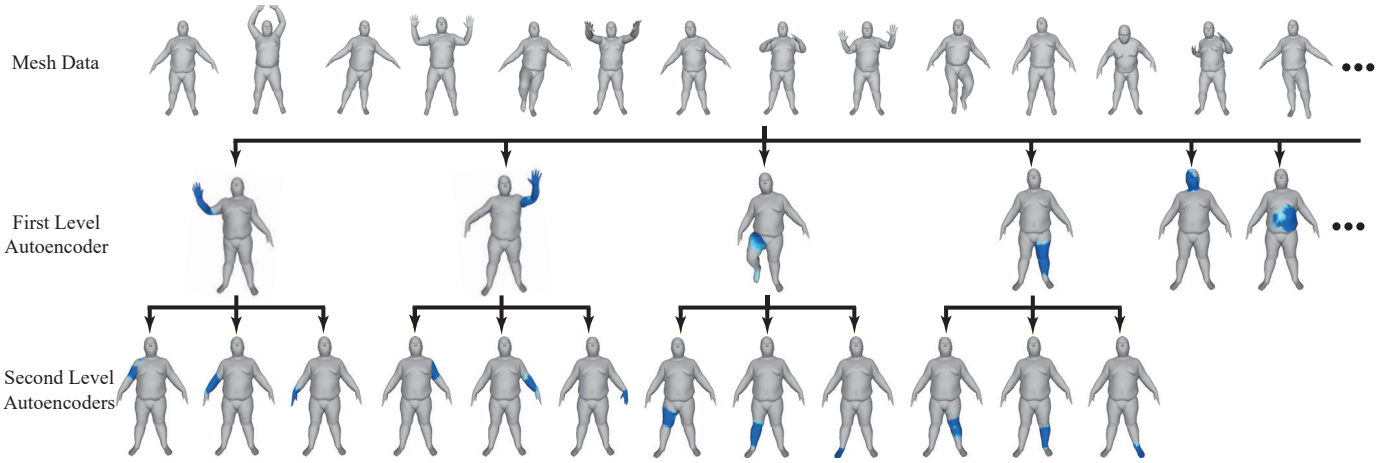


Figure 10. The multiscale structure of deformation components on the fat person (ID: 50002) from the Dyna [85] dataset, extracted by our method with redundant component removal. The first row shows example shapes in the dataset, the second row presents coarse-level deformation components from the first-level AE_0, and the third row shows the fine-level deformation components from the second-level AEs.

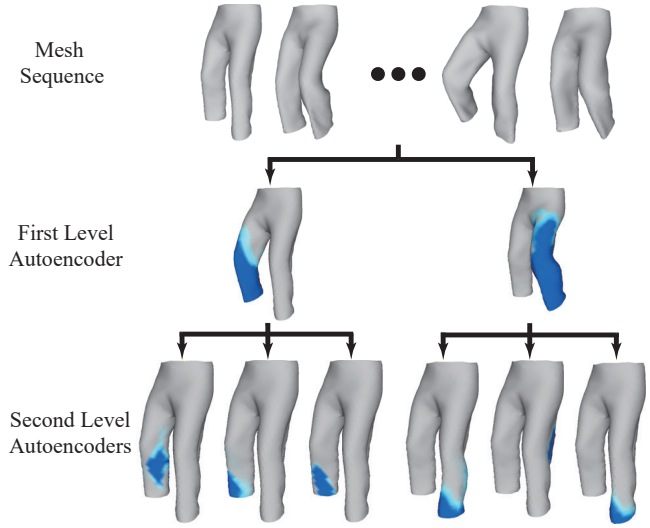


Figure 11. The multiscale structure of deformation components on the running pants data set [84]. In the figure, we have removed redundant components. Our method can learn deformation components at different scales. The first row shows example shapes of the running pants, the second row gives deformation components extracted by the first-level AE, which correspond to deformations caused by leg movement, and the third row presents the detailed deformations of the cloth from the second-level AEs.

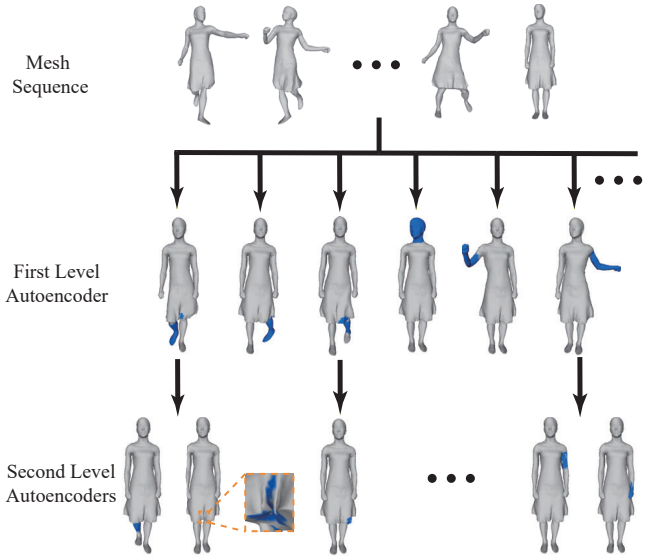


Figure 12. The multiscale structure of deformation components on the Swing [86] dataset. The first row shows some example shapes in the dataset, the second row presents coarse-level deformation components from the first-level AE_0, and the third row shows the fine-level deformation components from the second-level AEs ($AE_k, 1 \leq k \leq K$).

The Flag, Dress and Skirt datasets are synthesized by physical simulation. The Skirt dataset contains more complex motion and deformations. In Figs. 10, 12, 9, 8, 13, the components extracted by our method are in a multiscale manner and consistent with semantic meanings and our method can learn deformation components of different scales with multiscale structure from complex datasets.

We further compare shape editing using various methods by given control points and deformation components they extract. An example is shown in Fig. 14. We use the 8 control points (rendered as green balls) on the 29^{th} shape in the SCAPE dataset [82] manually chosen by the user. Then we apply the data-driven method [19] to reconstruct it with the help of the extracted deformation components by various methods. As the left part of Fig. 14 shows, our result is similar to the ground truth and plausible, while the other results have some artifacts and distortions, such as the right arm of [9] and left arm of [10]. The right part of Fig. 14 shows the three main activated components during data-driven deformation. Since the SCAPE contains much large-scale rotation, the method [10] only focuses on extracting large-scale deformation, but fails to capture important fine details, which results in the serious distortion in the arm due to lack of essential components.

5.3 Parameter Settings and Ablation Study

In this section, we evaluate the model sensitivity to the parameters, including the weights (λ_1, λ_2) in the loss function, the size of deformation region (d of Λ_{ik}), the effect of attention mechanism

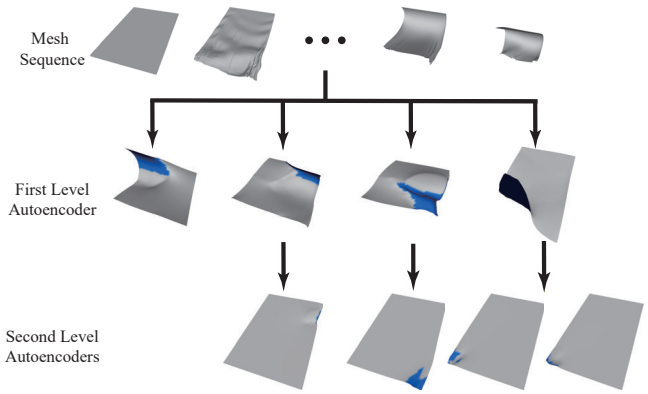


Figure 13. The multiscale structure of deformation components on the Flag dataset extracted by our method. The first row shows some example shapes of the dataset, the second row presents coarse-level deformation components from the first-level AE (AE_0), and the third row shows the fine-level deformation components from the second-level AEs ($AE_k, 1 \leq k \leq K$).

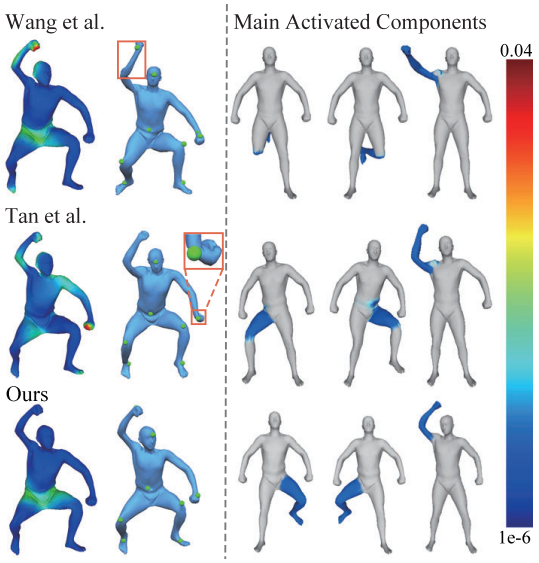


Figure 14. Comparison of shape reconstruction with different methods. The first column shows the error heat maps on the ground truth shape between the editing results and ground truth. The second column presents the editing results of different methods. The right side of the figure shows the three main activated deformation components during data-driven deformation. We use the same control points manually chosen by the user (8 vertices on the 29^{th} shape in the SCAPE [82]) to reconstruct the shape with the data-driven deformation method [19] and the same number of components. The figure shows that our result is more plausible than the existing methods, and it is similar to the ground truth.

on generalization error and the difference between joint training and separate training.

5.3.1 The choice of λ_1, λ_2

We test the influence of parameters λ_1, λ_2 on the generalization ability of the network. We evaluate it by E_{rms} errors of reconstructing unseen shapes on the SCAPE dataset. By fixing λ_1 to the default value 10, we change λ_2 from 0.2 to 30 as the right curve of Fig. 15 shows. The result shows our network is robust to different choices of λ_2. With fixed λ_2, we change λ_1 from 0.2 to 30 as the left curve of Fig. 15 shows. The result justifies that our

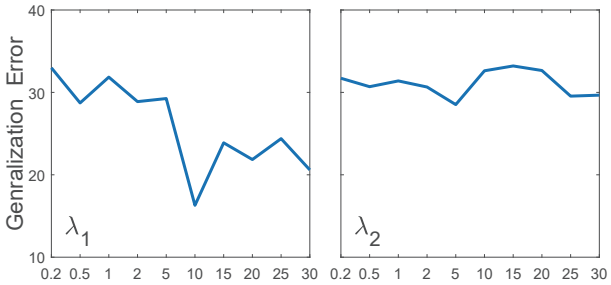
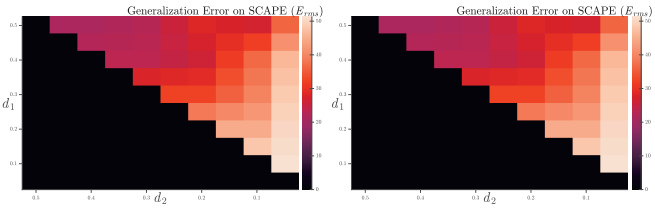


Figure 15. E_{rms} errors of generating unseen shapes with our overall autoencoder w.r.t. the weights λ_1 and λ_2. The figure shows that our network can get lower errors when $\lambda_1 = 10$ and is robust to different choice of λ_2.



(a) SCAPE [82] (b) CoMA [66]

Figure 16. The relationship between E_{rms} of generating unseen shapes and (d_1, d_2) is visualized, where d_1 and d_2 are from AE_0 (first-level AE) and $AE_k, 1 \leq k \leq K$ (second-level AEs) on six datasets, including Dress, Fat, Pants, SCAPE, Skirt, and CoMA. The error distributions on the heatmap are not entirely smooth in the figure, due to the gradient-descent training procedure that leads to local minima. The black (below the main diagonal) means no data, as by definition d_1 should be larger than d_2.

network can get lower errors when $\lambda_1 = 10$, which is chosen as the default value in our experiments.

5.3.2 The choice of d in $\mathbf{\Lambda_{ik}}$

The other parameter to choose is d in Λ_{ik}, which is the parameter that determines the scale of the extracted components. In our network, by default we stack two levels of autoencoders, so we need choose two parameters d_1, d_2 for both the first-level and the second-level autoencoders (AE_0 and $AE_k (1 \leq k \leq K)$) respectively. d_1, d_2 are cutoffs for normalized geodesic distances with a range from 0 to 1, and thus the original size of specific shapes in the dataset has little effect on them. These parameters only reflect the relevant size of the localized deformation components compared to the whole model. In order to extract multiscale deformation components, we need to ensure $d_1 > d_2$, then AE_k will cover a more detailed region than AE_0. As Fig. 16 shows, we test the network generalization ability (E_{rms}) on the SCAPE [82] and CoMA [66] datasets with different combinations d_1, d_2 by changing values from 0.05 to 0.5 with step 0.05. The figure illustrates that using smaller d_1 and d_2 results in larger reconstruction errors. Although lower errors can be achieved when d_1 and d_2 are large enough (close to 0.5), this will lead to extracting more global deformation components, which are not suitable for localized editing and may not be perceptually meaningful. It is therefore a trade-off to balance the generalization ability and the extracted deformation components. In our experiments, we set $d = [0.4, 0.2]$, which can make the network extract localized components while keeping lower reconstruction errors.

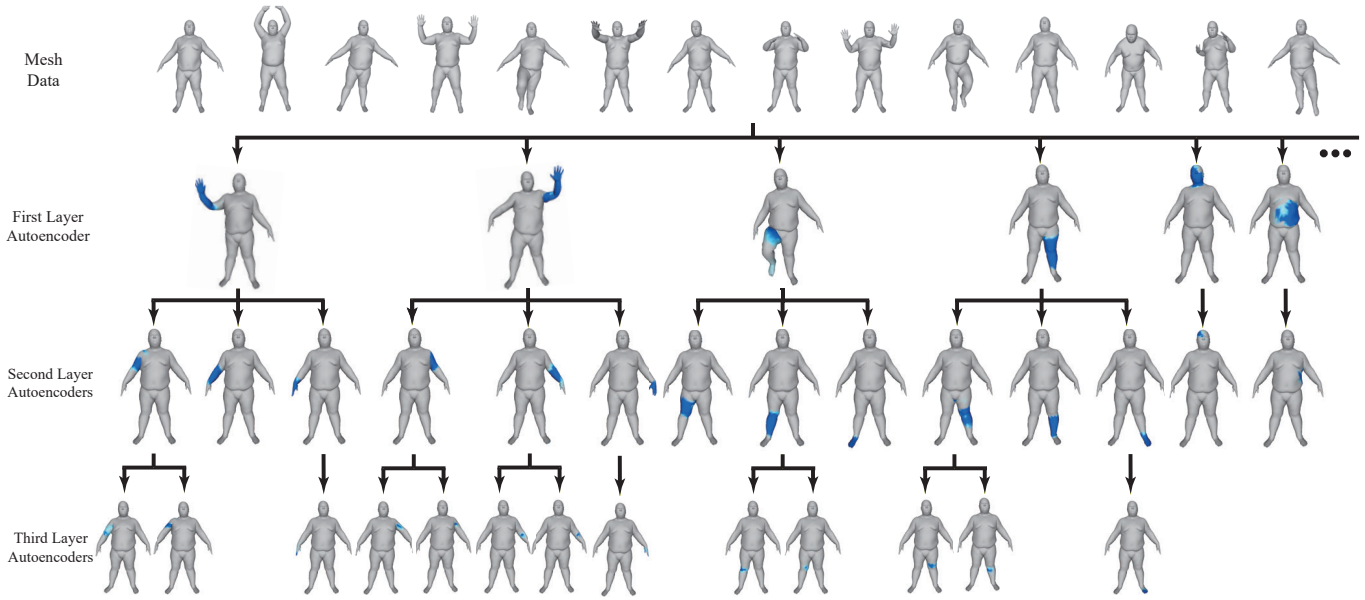


Figure 17. Three levels of visualization results on the Fat [85] dataset. In this figure, we display the extracted components visually from the three-level autoencoders. For the 3^{rd} level AEs, we can see that the extracted components have no clear semantic meanings and largely contain noise, which is not desired and not useful for shape analysis/editing.

Table 4
Comparison of different training strategies and the influence of the attention mechanism for shape reconstruction. We show results comparing joint training with separate training, and whether or not the attention mechanism is used. For each setting, we test it on several data sets by computing E_{rms} of the reconstructed shapes for unseen shapes. It shows that joint training with attention mechanism gives the best results.

DataSet	Swing	SCAPE	Pants	Humanoid	Horse	Flag	Dress	Jumping	Fat	Face
With Attention Joint Training	**12.2615**	**23.6807**	**6.4083**	**4.1538**	**6.9246**	**20.0627**	**11.5744**	**16.3475**	**4.3465**	**1.4409**
Without Attention Joint Training	17.1974	46.2339	8.2945	6.0285	20.8894	25.2675	12.5284	23.2829	5.3045	2.500
With Attention Separate Training	20.2918	36.1484	15.0249	6.0469	11.5899	24.4949	14.0248	19.1332	4.4469	2.1345
Without Attention Separate Training	26.7981	43.4439	19.8732	8.0583	17.4221	28.9938	17.3392	24.9437	7.3341	3.9472

5.3.3 The number of AE levels

The architecture of this neural net supports multiple levels of deformation scales from coarse to fine. In almost all tested dataset in this paper, the two level AEs architecture is enough to represent. The error between the input ACAP feature X_i and the reconstructed ACAP feature $\hat{X}_i, 1 < i < N$ is divided by the norm of X_i to get the relative squared errors as shown in Eqn. 8.

$$\hat{E}_i = \frac{||X_i - \hat{X}_i||_F^2}{||X_i||_F^2} \tag{8}$$

where $||\cdot||_F$ is the Frobenius norm, $\hat{E}_i$ is the relative squared error on the i^{th} shape in this dataset. Then, we choose the maximum relative squared error in the whole dataset to represent the amount of deformation that has not been represented in a normalized manner. We show the results in Table 5. The experiment illustrates that the relative squared errors of two-level AEs are very low in all datasets so that our two-level AEs are sufficient, and the extra 3^{rd} level AEs cannot improve the generalization on unseen data significantly compared to the 2^{nd} AEs.

Besides that, we also evaluate it on the Fat [85] dataset qualitatively. From Figure 17, we can see that the extracted components of the third-level AEs are not meaningful and mainly correspond to noise, which are not useful for applications such as shape analysis and editing.

5.3.4 The choice of K_z

K_z specifies the dimensions of the latent spaces of two-level AEs. This hyper-parameter decides the number of deformation components in each autoencoder of each level. Since methods compared in the paper produce 50 deformation components, we let $K_{z_0} \times K_{z_1} = 50$ for fair comparison. We have the following combinations: $K_z = [1, 50], [2, 25], [5, 10], [10, 5], [25, 2], [50, 1]$. However, $[1, 50], [50, 1]$ are trivial settings, so we test the other four settings of K_z on the SCAPE dataset. Table 6 shows the results, which illustrate that it performs well when $K_z = [25, 2]$. But, there are only two deformation components for each second-level AE, which is not reasonable for most datasets. For example, in Fig. 2, our network extracts three meaningful components for some branches on the SCAPE dataset. This is consistent with

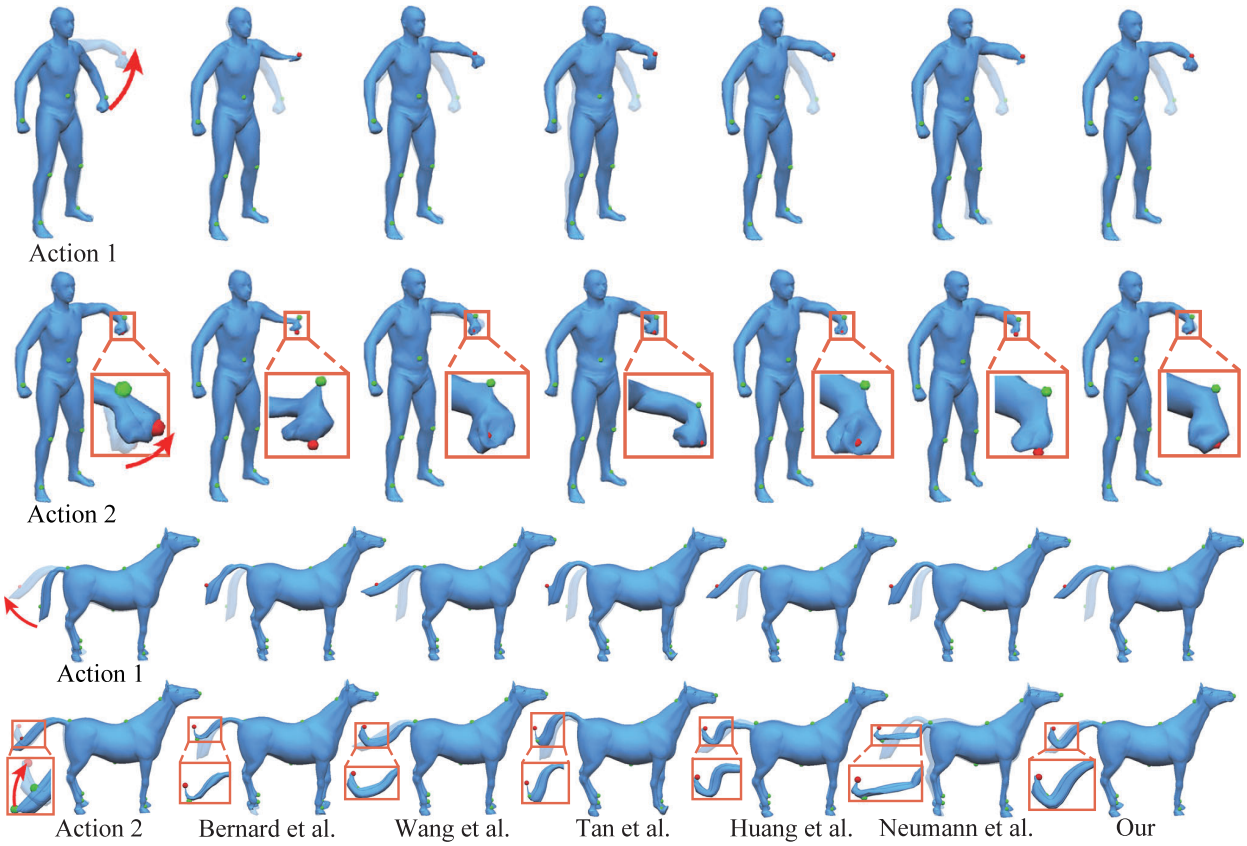


Figure 18. Comparison of multiscale shape editing. We compare the editing results with different methods on the SCAPE [82] and Horse [4] datasets. The results demonstrate that our extracted deformation components are suitable for multiscale shape editing. The first column shows the editing steps. Every row gives the deformed results of the corresponding action of various methods. The differences with other methods are highlighted in the orange rectangle with closeups to show the details. The existing methods have obvious distortions, demonstrating the superiority of our multiscale deformation components.

Table 5
The maximum of per-shape relative squared errors for every tested dataset in this paper. The errors are calculated by Eqn. 8 (Eqn. 8 in our paper). As shown in the table, the two-level AEs can represent deformations very well with tiny relative reconstruct errors, while the reconstruction errors with one-level AE are much larger for reconstructing the input features. While adding another level of AEs slightly reduces reconstruction errors, the extra 3^{rd} level AE cannot improve the generalization on unseen data significantly, compared to the 2^{nd} AE.

Dataset	Horse	Face	Jumping	Swing	Pants	Dress	Fat	Flag	Humanoid	Skirt	CoMA	SCAPE
Relative Squared Error (one-level AEs)	0.1219	0.06325	0.2312	0.4756	0.1763	0.5471	0.1322	0.1983	0.02563	0.4319	0.1171	0.3977
Relative Squared Error (two-level AEs)	0.005491	0.001065	0.007800	0.06876	0.01745	0.06296	0.005652	0.02674	0.008002	0.07159	0.01960	0.02155
Relative Squared Error (three-level AEs)	0.003567	0.0008306	0.004958	0.01065	0.009976	0.05927	0.003280	0.02595	0.003805	0.04081	0.01143	0.01225

observations from other datasets. So we choose the second high performance with the setting $K_z = [10, 5]$.

5.3.5 *Training strategies and attention mechanism*

Finally, we evaluate the effect of the attention mechanism and different training strategies. The statistics are shown in Table 4 based on the experiments on the SCAPE dataset.

For the training strategy, we compare the reconstruction error (E_{rms}) by training the network either jointly or separately. The results are shown in the first and third rows of Table 4, and jointly training the network can get better results.

We also perform the experiment to demonstrate the effect of the attention mechanism qualitatively and quantitatively. The results are shown in the first and second rows of Table 4. Training with attention mechanism gets lower errors. The reason is that each autoencoder of the second-level will focus on a different

Table 6
The reconstruction errors on the unseen data from SCAPE with different settings of K_z. For each setting, it represents the dimensions of latent spaces of two-level AEs. From the table, it shows that our method has high performance with the setting $K_z = [25, 2]$. However, there are only up to 2 deformation components for each second-level AE with the setting, which is not reasonable for most of datasets. For example, in figure 2, our network extracts three meaningful components for some branches on the SCAPE dataset. So we choose $K_z = [10, 5]$.

K_z	$[2, 25]$	$[5, 10]$	$[10, 5]$	$[25, 2]$
E_{rms}	34.9763	29.1127	23.6807	21.7749

sub-region to minimize the loss.

For the qualitative evaluation, we show the results in Fig. 9. If we train our network without the attention mechanism, our network architecture degenerates into the version of Tan et al. [10] with two-level autoencoders. In this case, a single second-level AE is sufficient as there would be no differences between them if multiple AEs were used. Despite this, different level autoencoders are also able to extract the different scale deformation components as shown in the right part of Fig. 9. In the left part of Fig. 9, the second and third rows represent the deformation components extracted by first-level and second-level AEs respectively, with redundant components removed. The first row shows some sample meshes in the Dress dataset. The results illustrate that without the attention mechanism, the deformation components no longer have a multiscale structure when the second level AEs do not focus on sub-regions to extract the deformations components.

5.3.6 Comparison with DEMEA on Shape Reconstruction

For fair and convincing comparison, we compare our baseline architecture with DEMEA [67] on the four datasets (COMA [66], SynHand5M [93], TextureLess Cloth [94], and DFaust [95]) with the same setting as used in [67]. We also use the same training set and test set split. With the same dimensions (8 and 32) of latent space, we also set our network with only one-level AE to make the architectures more comparable on the same metric (average per-vertex errors). In Table 7, we present the quantitative evaluation on the test datasets with different dimensions of latent space, and we can see that our method outperforms DEMEA on the four datasets for the shape reconstruction task. This also demonstrates that the shape representation and our autoencoder architecture are better choices than DEMEA [67]. Such benefits can be more substantial when shapes undergo more substantial deformations, such as various human body poses, so we build our network architecture based on this.

Table 7
In this table, we evaluate the generalization ability on the test data for 4 datasets (CoMA [66], SynHand5M [93], TextureLess Cloth [94], and DFaust [95]) and compare our baseline with DEMEA [67] with different dimensions of latent space. For fair comparison, we also set our network with only one-level AE to make the architectures more comparable and use the same metrics (average per-vertex errors).

Methods	CoMA		SynHand5M		Cloth		Dfaust	
Dimension of latent space	8	32	8	32	8	32	8	32
DEMEA [67]	1.49	1.05	8.97	4.67	13.40	8.30	6.60	2.90
Ours (baseline)	**1.17**	**0.82**	**6.10**	**3.30**	**10.03**	**4.62**	**6.05**	**2.17**

5.4 Multiscale Shape Editing

Multiscale shape editing is an important application in computer graphics. Users usually start with editing of the overall shape, and then focus on adjusting the details. With existing methods, the extracted deformation components either contain some global information [7], [28] thus making the components unsuitable for local editing, or focus too much on large-scale deformations and fail to capture essential small-scale deformations for faithful reconstruction, leading to distortions like [10], which would affect the users' editing efficiency for 3D animations. Given a shape deformation dataset that contains diverse deformations, our method can produce multiscale localized deformation components which are visually semantically meaningful, corresponding to typical deformation behaviour. Along with data-driven deformation [19], this allows users to edit shapes efficiently and intuitively under the constraints of the control points and subspace spanned by extracted deformation components. Please refer to the work [19] for implementation details of data-driven deformation.

Fig. 18 shows some examples. For the SCAPE dataset, we design two actions: raising the left arm (Action 1) and then turning the wrist (Action 2). All the compared methods perform well in Action 1. However, in Action 2, only our method can naturally twist wrist with the help of our extracted multi-scale deformation components. In contrast, all other methods have various distortions. For the Horse dataset, we also design two actions: raising the whole tail (Action 1) and then twisting the end of the tail (Action 2). Our method can bend the end of the tail naturally after raising the whole tail. But other methods have more distortions and even lead to changes on the entire tail, especially the methods [7], [28] based on the Euclidean coordinate representation. In summary, our extracted multiscale deformation components can perform better than existing methods in multiscale shape editing. See the accompanying video for more results.

6 Limitation and Conclusion

In this paper, we propose a novel autoencoder with the attention mechanism to extract multiscale localized deformation components. We use stacked AEs to extract multiscale deformation components. This helps capture richer information for better shape editing, with better generalization ability (see Table 2). Moreover, the first-level AE extracts some coarse level components and learns attention masks to help the second-level AEs focus on relevant sub-regions to extract fine-level components. Extensive quantitative and qualitative evaluations show that our method is effective, outperforming state-of-the-art methods. The extracted deformation components by our method can be used on multiscale shape editing for computer animation, which demonstrates that the extracted multiscale localized deformation components are effective and meaningful for reducing the user's efforts. In the future, this work can give more solutions and explorations on applying the attention mechanism on 3D shape analysis and synthesis.

Although our method can analyze the shape dataset in a multiscale manner to extract deformation components of different scales for easy shape editing, there are some limitations. Our method can only handle datasets containing meshes with the same connectivity. Although such datasets are common for deformable shapes, it would be useful to extend our method cope with general 3D shapes, e.g. from ShapeNet. In addition, our current method uses a fixed network architecture and attention mechanism to analyze shapes in a multiscale manner, which does not take into account unique characteristics of shape deformations in individual datasets. In the future we would like exploit analyzing and capturing the variation of the multiscale structures automatically. Furthermore, this method could also be implemented on other deep learning frameworks such as Jittor [53] – a just-in-time (JIT) compiled deep learning framework with higher efficiency. Finally, it would also be useful to improve the pipeline to merge he post-processing (Sec. 4.3) to the neural network which can predict the number of sub-components automatically.

ACKNOWLEDGMENTS

This work was supported by the the National Natural Science Foundation of China (No. 62061136007 and No. 61872440), the Beijing Municipal Natural Science Foundation (No. L182016), the Science and Technology Service Network Initiative, Chinese Academy of Sciences (No. KFJ-STS-QYZD-2021-11-001), Royal Society Newton Advanced Fellowship (No. NAF\R2\192151), the Youth Innovation Promotion Association CAS and the Open Research Projects of Zhejiang Lab (No. 2021KE0AB06).

REFERENCES

[1] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling," in *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*, 2007, pp. 109–116.

[2] T. Igarashi, T. Moscovich, and J. F. Hughes, "As-rigid-as-possible shape manipulation," *ACM transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 1134–1141, 2005.

[3] S.-Y. Chen, L. Gao, Y.-K. Lai, and S. Xia, "Rigidity controllable as-rigid-as-possible shape deformation," *Graphical Models*, vol. 91, pp. 13–21, 2017.

[4] R. W. Sumner and J. Popović, "Deformation transfer for triangle meshes," *ACM Transactions on graphics (TOG)*, vol. 23, no. 3, pp. 399–405, 2004.

[5] J. Yang, L. Gao, Y.-K. Lai, P. L. Rosin, and S. Xia, "Biharmonic deformation transfer with automatic key point selection," *Graphical Models*, vol. 98, pp. 1–13, 2018.

[6] L. Gao, J. Yang, Y.-L. Qiao, Y.-K. Lai, P. L. Rosin, W. Xu, and S. Xia, "Automatic unpaired shape deformation transfer," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–15, 2018.

[7] T. Neumann, K. Varanasi, S. Wenger, M. Wacker, M. Magnor, and C. Theobalt, "Sparse localized deformation components," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 179, 2013.

[8] Z. Huang, J. Yao, Z. Zhong, Y. Liu, and X. Guo, "Sparse localized decomposition of deformation gradients," in *Computer Graphics Forum (CGF)*, vol. 33, no. 7. Wiley Online Library, 2014, pp. 239–248.

[9] Y. Wang, G. Li, Z. Zeng, and H. He, "Articulated-motion-aware sparse localized decomposition," in *Computer Graphics Forum (CGF)*, vol. 36, no. 8. Wiley Online Library, 2017, pp. 247–259.

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

[11] C. N. Alleman, J. W. Foulk, A. Mota, H. Lim, and D. J. Littlewood, "Concurrent multiscale modeling of microstructural effects on localization behavior in finite deformation solid mechanics," *Computational Mechanics*, vol. 61, no. 1-2, pp. 207–218, 2018.

[12] M. Mathew, A. Ellenberg, S. Esola, M. McCarthy, I. Bartoli, and A. Kontsos, "Multiscale deformation measurements using multispectral optical metrology," *Structural Control and Health Monitoring*, vol. 25, no. 6, p. e2166, 2018.

[13] M. K. Abeyratne, W. Freeden, and C. Mayer, "Multiscale deformation analysis by cauchy-navier wavelets," *Journal of Applied Mathematics*, vol. 2003, 2002.

[14] K. C. Lam, T. C. Ng, and L. M. Lui, "Multiscale representation of deformation via beltrami coefficients," *Multiscale Modeling & Simulation*, vol. 15, no. 2, pp. 864–891, 2017.

[15] Y. Yang, W. Xu, X. Guo, K. Zhou, and B. Guo, "Boundary-aware multidomain subspace deformation," *IEEE transactions on visualization and computer graphics (TVCG)*, vol. 19, no. 10, pp. 1633–1645, 2013.

[16] H. Hamidian, Z. Zhong, F. Fotouhi, and J. Hua, "Surface registration with eigenvalues and eigenvectors," *IEEE transactions on visualization and computer graphics (TVCG)*, 2019.

[17] S.-L. Liu, Y. Liu, L.-F. Dong, and X. Tong, "Ras: A data-driven rigidity-aware skinning model for 3d facial animation," in *Computer Graphics Forum (CGF)*, vol. 39, no. 1. Wiley Online Library, 2020, pp. 581–594.

[18] J. Fu, H. Zheng, and T. Mei, "Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2017, p. 3.

[19] L. Gao, Y.-K. Lai, J. Yang, Z. Ling-Xiao, S. Xia, and L. Kobbelt, "Sparse data driven mesh deformation," *IEEE transactions on visualization and computer graphics (TVCG)*, 2019.

[20] W. Xu, J. Wang, K. Yin, K. Zhou, M. Van De Panne, F. Chen, and B. Guo, "Joint-aware manipulation of deformable models," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3, pp. 1–9, 2009.

[21] M. E. Yumer and L. B. Kara, "Co-constrained handles for deformation in shape collections," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 6, p. 187, 2014.

[22] M. E. Yumer, S. Chaudhuri, J. K. Hodgins, and L. B. Kara, "Semantic shape editing using deformation handles," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 86, 2015.

[23] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. Shum, "Large mesh deformation using the volumetric graph laplacian," in *ACM SIGGRAPH 2005 Papers*, 2005, pp. 496–503.

[24] L. Gao, Y.-K. Lai, D. Liang, S.-Y. Chen, and S. Xia, "Efficient and flexible deformation representation for data-driven surface modeling," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 5, p. 158, 2016.

[25] M. Alexa and W. Müller, "Representing animations by principal components," in *Computer Graphics Forum (CGF)*, vol. 19, no. 3. Wiley Online Library, 2000, pp. 411–418.

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

[27] 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.

[28] F. Bernard, P. Gemmar, F. Hertel, J. Goncalves, and J. Thunberg, "Linear shape deformation models with local support using graph-based structured matrix factorisation," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5629–5638.

[29] J. R. Tena, F. De la Torre, and I. Matthews, "Interactive region-based linear 3D face models," in *ACM Transactions on Graphics (TOG)*, 2011, vol. 30, no. 4, pp. 1–10.

[30] T. Neumann, K. Varanasi, N. Hasler, M. Wacker, M. Magnor, and C. Theobalt, "Capture and statistical modeling of arm-muscle deformations," in *Computer Graphics Forum (CGF)*, vol. 32, no. 2pt3. Wiley Online Library, 2013, pp. 285–294.

[31] S. Fröhlich and M. Botsch, "Example-driven deformations based on discrete shells," in *Computer graphics forum (CGF)*, vol. 30, no. 8. Wiley Online Library, 2011, pp. 2246–2257.

[32] M. Botsch and O. Sorkine, "On linear variational surface deformation methods," *IEEE transactions on visualization and computer graphics (TVCG)*, vol. 14, no. 1, pp. 213–230, 2007.

[33] Y.-J. Yuan, Y.-K. Lai, T. Wu, L. Gao, and L. Liu, "A revisit of shape editing techniques: From the geometric to the neural viewpoint," *Journal of Computer Science and Technology*, vol. 36, no. 3, pp. 520–554, 2021.

[34] M. Chen, C. Wang, and L. Liu, "Cross-domain retrieving sketch and shape using cycle cnns," *Computers & Graphics*, 2020.

[35] C. Jiang, D. Wang, J. Huang, P. Marcus, M. Nießner *et al.*, "Convolutional neural networks on non-uniform geometrical signals using euclidean spectral transformation," 2019.

[36] K. Sarkar, K. Varanasi, and D. Stricker, "3D shape processing by convolutional denoising autoencoders on local patches," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 1925–1934.

[37] C. Jiang, D. Lansigan, P. Marcus, and M. Nießner, "DDSL: Deep differentiable simplex layer for learning geometric signals," in *Proceedings of the IEEE International Conference on Computer Vision(ICCV)*, 2019, pp. 8769–8778.

[38] D. Maturana and S. Scherer, "VoxNet: a 3D convolutional neural network for real-time object recognition," in *IEEE Conference on Intelligent Robots and Systems*, 2015, pp. 922–928.

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

[40] M. E. Yumer and N. J. Mitra, "Learning semantic deformation flows with 3D convolutional networks," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 294–311.

[41] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-CNN: Octree-based convolutional neural networks for 3D shape analysis," *ACM Transactions On Graphics (TOG)*, vol. 36, no. 4, pp. 1–11, 2017.

[42] P.-S. Wang, C.-Y. Sun, Y. Liu, and X. Tong, "Adaptive o-cnn: A patch-based deep representation of 3d shapes," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–11, 2018.

[43] P.-S. Wang, Y. Liu, and X. Tong, "Deep octree-based cnns with output-guided skip connections for 3d shape and scene completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 266–267.

[44] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 820–830, 2018.

[45] H. Lei, N. Akhtar, and A. Mian, "Spherical convolutional neural network for 3D point clouds," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9631–9640.

[46] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (TOG)*, vol. 38, no. 5, pp. 1–12, 2019.

[47] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "Pct: Point cloud transformer," *Computational Visual Media*, vol. 7, no. 2, pp. 187–199, 2021.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems(NeurIPS)*, 2017, pp. 5998–6008.

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

[50] K. Sarkar, B. Hampiholi, K. Varanasi, and D. Stricker, "Learning 3d shapes as multi-layered height-maps using 2d convolutional networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 71–86.

[51] H. Huang, E. Kalegorakis, S. Chaudhuri, D. Ceylan, V. Kim, and E. Yumer, "Learning local shape descriptors with view based convolutional neural networks," *ACM Transactions on Graphics (TOG)*, vol. 2, 2018.

[52] S.-M. Hu, Z.-N. Liu, M.-H. Guo, J.-X. Cai, J. Huang, T.-J. Mu, and R. R. Martin, "Subdivision-based mesh convolution networks," *arXiv preprint arXiv:2106.02285*, 2021.

[53] S.-M. Hu, D. Liang, G.-Y. Yang, G.-W. Yang, and W.-Y. Zhou, "Jittor: a novel deep learning framework with meta-operators and unified graph execution," *Science China Information Sciences*, vol. 63, no. 12, pp. 1–21, 2020.

[54] Y.-J. Yuan, Y.-K. Lai, J. Yang, Q. Duan, H. Fu, and L. Gao, "Mesh variational autoencoders with edge contraction pooling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 274–275.

[55] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015, pp. 2224–2232.

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

[57] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. [Online]. Available: https://arxiv.org/abs/1606.09375

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

[59] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016, pp. 3844–3852.

[60] T. Qingyang, L.-X. Zhang, J. Yang, Y.-K. Lai, and L. Gao, "Mesh-based variational autoencoders for localized deformation component analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence(PAMI)*, 2021.

[61] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[62] A. Ioannidou, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris, "Deep learning advances in computer vision with 3d data: A survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–38, 2017.

[63] Y. Jin, D. Jiang, and M. Cai, "3d reconstruction using deep learning: a survey," *Communications in Information and Systems*, vol. 20, no. 4, pp. 389–413, 2020.

[64] Y.-P. Xiao, Y.-K. Lai, F.-L. Zhang, C. Li, and L. Gao, "A survey on deep geometry learning: From a representation perspective," *Computational Visual Media*, vol. 6, no. 2, pp. 113–133, 2020.

[65] Q. Tan, L. Gao, Y. Lai, and S. Xia, "Variational autoencoders for deforming 3D mesh models," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 5841–5850.

[66] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black, "Generating 3D faces using convolutional mesh autoencoders," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 725–741.

[67] E. Tretschk, A. Tewari, M. Zollhöfer, V. Golyanik, and C. Theobalt, "DEMEA: Deep Mesh Autoencoders for Non-Rigidly Deforming Objects," *arXiv:1905.10290v1*, 2020.

[68] O. Litany, A. Bronstein, M. Bronstein, and A. Makadia, "Deformable shape completion with graph convolutional autoencoders," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2018, pp. 1886–1895.

[69] L. Fulton, V. Modi, D. Duvenaud, D. I. Levin, and A. Jacobson, "Latent-space dynamics for reduced deformable simulation," in *Computer Graphics Forum (CGF)*, vol. 38, no. 2. Wiley Online Library, 2019, pp. 379–391.

[70] S. Ma, J. Fu, C. W. Chen, and T. Mei, "DA-GAN: Instance-level image translation by deep attention generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 5657–5666.

[71] J. Si, H. Zhang, C.-G. Li, J. Kuen, X. Kong, A. C. Kot, and G. Wang, "Dual attention matching network for context-aware feature sequence based person re-identification," *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[72] J. Xu, R. Zhao, F. Zhu, H. Wang, and W. Ouyang, "Attention-aware compositional network for person re-identification," *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[73] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1480–1489.

[74] X. Zhang, T. Wang, J. Qi, H. Lu, and G. Wang, "Progressive attention guided recurrent network for salient object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 714–722.

[75] B. Zhuang, Q. Wu, C. Shen, I. Reid, and A. van den Hengel, "Parallel attention: A unified framework for visual object discovery through dialogs and queries," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2018, pp. 4252–4261.

[76] X. Long, C. Gan, G. de Melo, J. Wu, X. Liu, and S. Wen, "Attention clusters: Purely attention based local feature integration for video classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7834–7843.

[77] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[78] J. Lu, J. Yang, D. Batra, and D. Parikh, "Hierarchical question-image co-attention for visual question answering," in *Advances In Neural Information Processing Systems (NeurIPS)*, 2016, pp. 289–297.

[79] P. Wang, Q. Wu, C. Shen, and A. van den Hengel, "The VQA-machine: Learning how to use existing vision algorithms to answer new questions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 4, 2017, pp. 1173–1182.

[80] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3156–3164, 2017.

[81] K. Crane, C. Weischedel, and M. Wardetzky, "Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow," *ACM Transactions on Graphics (TOG)*, vol. 32, pp. 152:1–152:11, 2013.

[82] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "SCAPE: shape completion and animation of people," in *ACM transactions on graphics (TOG)*, vol. 24, no. 3, 2005, pp. 408–416.

[83] L. Zhang, N. Snavely, B. Curless, and S. M. Seitz, "Spacetime faces: High-resolution capture for modeling and animation," in *ACM Transactions on Graphics (TOG)*, 2004, pp. 548–558.

[84] R. White, K. Crane, and D. A. Forsyth, "Capturing and animating occluded cloth," in *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3. ACM, 2007, p. 34.

[85] G. Pons-Moll, J. Romero, N. Mahmood, and M. J. Black, "Dyna: A model of dynamic human shape in motion," *ACM Transactions on Graphics(TOG)*, vol. 34, no. 4, pp. 120:1–120:14, 2015.

[86] D. Vlasic, I. Baran, W. Matusik, and J. Popović, "Articulated mesh animation from multi-view silhouettes," in *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3. ACM, 2008, pp. 97:1–97:9.

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

[88] L. Kavan, P.-P. Sloan, and C. O'Sullivan, "Fast and efficient skinning of animated meshes," in *Computer Graphics Forum (CGF)*, vol. 29, no. 2. Wiley Online Library, 2010, pp. 327–336.

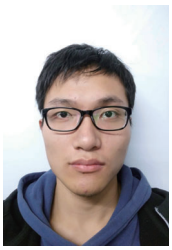
[89] L. Vasa and V. Skala, "A perception correlated comparison method for dynamic meshes," *IEEE transactions on visualization and computer graphics (TVCG)*, vol. 17, no. 2, pp. 220–230, 2011.

[90] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi, "The farthest point strategy for progressive image sampling," *IEEE Transactions on Image Processing*, vol. 6, no. 9, pp. 1305–1315, 1997.
[91] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008.
[92] R. W. Sumner, M. Zwicker, C. Gotsman, and J. Popović, "Mesh-based inverse kinematics," in *ACM transactions on graphics (TOG)*, vol. 24, no. 3. ACM, 2005, pp. 488–495.
[93] J. Malik, A. Elhayek, F. Nunnari, K. Varanasi, K. Tamaddon, A. Heloir, and D. Stricker, "Deephps: End-to-end estimation of 3d hand pose and shape by learning from synthetic depth," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 110–119.
[94] J. Bednarik, P. Fua, and M. Salzmann, "Learning to reconstruct texture-less deformable surfaces from a single view," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 606–615.
[95] F. Bogo, J. Romero, G. Pons-Moll, and M. J. Black, "Dynamic FAUST: Registering human bodies in motion," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.

Jie Yang received a bachelor's 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.

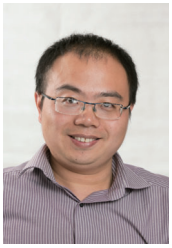
Lin Gao received the bachelor's degree in mathematics from Sichuan University and the PhD degree in computer science from Tsinghua University. He is currently an Associate Professor at the Institute of Computing Technology, Chinese Academy of Sciences. He has been awarded Royal Society Newton Advanced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

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.



Yi-Hua Huang obtained his bachelor degree from the University of Chinese Academy of Sciences. He is currently a graduate candidate in the Institute of Computation Technology, Chinese Academy of Sciences. His research interests include computer graphics and visions.

Shihong Xia is a professor associated with the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences. He received a bachelor's degree in mathematics from Sichuan Normal University, China, in 1996 and a PhD degree in computer software and theory from the University of Chinese Academy of Sciences in 2002. His research interests include computer graphics, virtual reality and artificial intelligence.



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*.