3D Mesh Pose Transfer Based on Skeletal Deformation

Shigeng Yang, Mengxiao Yin*, Ming Li, Kan Chang, Feng Yang
School of Computer Electronics and Information, Guangxi University
Nanning, China
1065957715@qq.com, ymx@gxu.edu.cn, 403343133@qq.com, changkan0@gmail.com, yf@gxu.edu.cn

Guiqing Li
School of Computer Science and Engineering, South China University of Technology
Guangzhou, China
ligq@scut.edu.cn

Abstract

For 3D mesh pose transfer, the target model is obtained by transferring the pose of the reference mesh to the source mesh, where the shape and pose of the source are usually different from that of the reference. In this paper, pose transfer is considered as a deformation process of the source mesh, and we propose a 3D mesh pose transfer method based on skeletal deformation. First, we design a neural network based on the edge convolution operator to extract the skeleton of the 3D mesh and bind the rigid weights; then, we calculate the bone transformations between the two skeletons with different poses and use the diffusion equation to smooth the rigid weights; finally, the source mesh is deformed according to the bone transformations and the smooth weights to get the target mesh. Experiment results on different datasets show that the pose of the reference mesh can be effectively transferred to the source one while maintaining the shape and high-quality geometric details of the source mesh by using our method.

***Keywords:** Pose transfer,Deep learning,Skinning deformation,Skeleton extraction*

1. Introduction

3D mesh is an important research object in computer graphics, widely used in computer animation, film, video games, and other related industries. 3D mesh deformation and editing are one of the current research hot spots, and new meshes can be obtained by deforming and editing existing models [44].

As example-based deformation and editing methods, deformation transfer and pose transfer have many research results. Moreover, with the wide application of deep learning in graphics, pose transfer based on deep learning has begun to attract attention [35].

Inspired by image style transfer in computer vision, current pose transfer methods based on deep learning [35, 39] are mainly end-to-end networks with encoder-decoder architecture. First, the encoder encodes the source mesh's shape information and the reference mesh's pose information into the latent space. Then the decoder generates the detail-preserving target mesh. However, these methods have the following problems: firstly, due to the lack of regularity constraint, there is a high degree of freedom of encoding in the latent space [23]. As a result, the intrinsic connections of the original signal cannot be well represented by the generated feature space, so the network is prone to overfitting. For example, the model testing results of [39] show that, for datasets outside the training shape and pose space, the prediction error metric Point-wise Mesh Euclidean Distance (PMD) is much larger than the seen pose in the training set. Second, PointNet [34] is used by these networks for feature extraction. However, PointNet learns each vertex feature independently, which ignores the connection between vertices and cannot effectively capture the local features information between vertices [15]. Therefore, the predicted target mesh is distorted, even with slight changes such as an increase in the number of vertices or a change in the mesh connectivity.

Different from the above methods, to avoid the overfitting phenomenon caused by the high degree of freedom of encoding in the latent space of the above end-to-end neural networks, we design a neural network that encodes the topology and shape information. The mesh skeleton and rigid skinning weights are all obtained from the network. We use the skeleton to present the pose of the mesh and explicitly solve the transformation from the reference skeleton to the source one. Then, with the help of LBS deformation,

*Corresponding author

the detail-preserving target mesh is obtained. Our method has better generalization than the end-to-end neural network methods.

To solve the problem that PointNet cannot effectively capture local features, we get a new edge convolution operator GKEdgeConv by slightly modifying the edge convolution operator of DGCNN [41], which can effectively encode the local information of mesh vertices and skeleton joints. We use it as the basis for designing a neural network to extract the mesh's skeleton joints and rigid weights.

We calculate the rigid transformation matrix of corresponding joints between two pose skeletons in the skinning deformation phase. The skinning weights are considered a function of time variation on the mesh, and the initial state of the function is the rigid weights bound by the neural network. Next, we smooth the rigid weights using the diffusion equation based on the cotangent Laplacian to gain the weights satisfying the properties of non-negativity, linearity, smoothness, *etc*. Finally, the detail-preserving target mesh is obtained using linear blend skinning deformation.

The contributions of this paper are summarized as follows:

- We proposed a 3D mesh pose transfer method based on skeletal deformation, and the pose transfer problem is reformulated to a skin deformation problem. Experiments on the test dataset show that the target mesh predicted by this method on the unknown pose space has better performance in terms of accuracy and detail-preserving compared with other deep learning-based pose transfer methods.

- We designed a neural network for skeleton extraction and rigid weight binding of 3D mesh, which can effectively encode local information of mesh vertices and skeleton joints using the edge convolution operator proposed in this paper.

- A rigid skinning weights deformation method based on diffusion equation is posed, and the weights satisfying the geometric proxy properties are obtained by smoothing the rigid weights with cotangent Laplace. The experimental results of LBS deformation show that the target mesh obtained by deforming has satisfactory geometric details.

2. Related Work

This section briefly reviews the primary deformation transfer and pose transfer methods and then introduces the skinning deformation and automatic rigging methods that are closely relevant to this work.

2.1. Deformation transfer and pose transfer

For deformation transfer, the geometric transformation between two different poses is applied to another source model to produce similar deformations. The source model and the reference one are not required to have the same number of vertices or triangles [37], but the corresponding vertex pairs between the models should be specified as constraints to find the geometric correspondence. Baran *et al*. [4] proposed a semantical deformation transfer method, and deformation is transferred by establishing two semantically matched shape spaces. However, the user also needs to provide the corresponding semantic relationship of the models. Gao *et al*. [14] proposed a method based on generative adversarial networks. They combined cyclic consistency loss and visual similarity metric to achieve automatic deformation transfer between two non-corresponding shapes. However, recollecting data and retraining are needed whenever dealing with unseen models. More introduction to deformation transfer can be found in [14].

For pose transfer, there is only one reference pose and a source model. The final goal is to obtain a target model that keeps the source model's geometric details and a similar pose to the reference. Lévy [28] projected the vertices of two meshes to the Laplacian matrix harmonic basis and exchanged the low-frequency coefficients of two meshes with the same connectivity to realize pose transfer. However, for non-isometric meshes, there are serious distortions. Kovnatsky *et al*. [24] constructed coupled quasi-harmonic bases, which are compatible eigen basis for non-isometric meshes based on functional maps. Nonetheless, directly exchanging the low-frequency coefficients results in details loss. Yin *et al*. [43] proposed a detail-preserving hierarchical spectral pose transfer method, which is low efficiency due to the need for the user to interactively specify and partition the local meshes to be subjected to secondary pose transfer. To reduce the interaction, an automatic method to detect and segment the local meshes is proposed by Zou *et al*. [46]. However, during the low-frequency transfer, if the deformation of the local meshes is too small, that will lead to automatic detection failure, and manual operation should be introduced.

Deep learning methods have brought new ideas to de-artificialize pose transfer. There are [5, 12, 35, 39] and other methods have been proposed. For human pose transfer, an end-to-end neural network Neural Pose Transfer (NPT) is designed by Wang *et al*. [39]. They combined the PointNet [34] with the spatially adaptive instance normalization layer in image style transfer [17]. The encoder encodes both local details and global contexts of the reference mesh to obtain the pose feature, and then the pose feature is fed into the decoder to decode the output under the guidance of the source mesh. They achieve fully automated pose transfer. However, their method requires that the source mesh and the reference mesh have the same number of vertices because they simply concatenate the features without considering the correspondence between the meshes. Song *et al*. [35] es-

tablished the correspondence between the source mesh and the reference mesh by solving an optimal transport problem. They warp the reference mesh according to the dense correspondence and obtain a coarse warped mesh refined with elastic instance normalization layer. Since the above methods all treat vertex independently to maintain permutation invariance and the geometric properties of the mesh are ignored, it is difficult for the network to capture the local features of the mesh. In this paper, we design a network based on the edge convolution operator of DGCNN [41]. Our network can better encode the global and local features of the mesh, and the isomorphic skeleton of the source mesh and reference mesh can be constructed without the mesh correspondence. Instead of constructing the pose transfer results directly, we finally obtain better pose transfer results by skinning deformation.

2.2. Skinning deformation and Automatic rigging

Skinning deformation is usually a skeleton-based mesh deformation technique. LBS [31], SBS [21] and DQS [20] are common skinning deformation methods. LBS is widely used in computer animation, video games, and other fields because of its easy implementation and fast computation. However, serious problems such as elbow collapse and candy paper would be brought from the LBS deformation process. Each vertex's transformation is represented as a nonlinear function of skeletal transformations [20, 21, 26], which achieves a better trade-off between deformation quality and computation speed. However, these methods, like LBS, require a more fine-grained weight binding between the geometric proxy and the model. In recent years, Le and Lewis [27] proposed Direct Delta Mush (DDM) based on Delta Mush (DM) [32]. Laplace smoothing [13] is applied to the mesh before and after LBS deformation, and the artifacts caused by using rigid skin weights are eliminated in DM. Furthermore, the direct computational form of DM is derived, and the computational and storage costs are optimized in DDM. In this paper, we use fast LBS to deform the source mesh.

In skinning deformation, professional people build the underlying skeleton of the mesh and bind skinning weights. The automatic generation of the skeleton and the skinning weights are challenging in computer graphics [29]. The pioneering work was presented by Baran and Popović [3]. They do skinning by solving a heat equilibrium equation, yet a predefined skeleton template is required to fit the input model. Skeletons are extracted by analyzing the geometric features of the input model in [6, 11], but there is a lack of precise control over the output skeleton topology.

Wang and Solomon [40] divided skinning weight calculation into geometry-based methods and sample-based methods. For geometry-based methods, Baran and Popović [3] solved the heat equilibrium equation, and a harmonic equation is solved in [19, 45], resulting in the harmonic weights related to the mesh resolution. Biharmonic equations can also be used for skinning weight computation [9]. Jacobson *et al*. [18] obtained bounded biharmonic weights by minimizing the biharmonic energy with constraints. In recent years, Bang and Lee [2] edited weights through interactive spline curves; Wang and Solomon [40] regarded the weight calculation as the inverse problem of recovering the best anisotropy tensor and got the quasi-harmonic weights by solving the second-order parametric elliptic partial differential equation. Although geometry-based methods can yield high-quality skinning weight, manual interaction or more complex calculations are usually required.

For sample-based methods, the spatial consistency among the samples can be used for fitting the samples to skinning models such as LBS to obtain high-quality skeletons and weights. [25, 38] are the classical methods. However, it is necessary to provide a set of deformation samples of the model for these methods, which is difficult to satisfy in practical applications. With the development of neural networks, methods whose input is only a single model are proposed. RigNet posed by Xu *et al*. [42] is an end-to-end model based on a graph neural network. Mesh shrinkage and an attention-based clustering method are applied to predict the skeleton. The user can control the sparsity of the nodes of the generated skeleton by adjusting the parameters but cannot directly control the topology of the generated skeleton. The prediction network of skinning weight is similar to that of the skeleton, besides the vertex-to-bone volumetric geodesic distance as the input feature is needed. Li *et al*. [29] considered the desired skeleton hierarchy in the network architecture and proposed the Neural blend shape (NBS). They used the edge convolution operators of MeshCNN [16] to build the network. A graph convolution operator with skeleton-aware features and a pooling operator based on mask weights are applied to predict the offset of child joints relative to the parent joint. The method assumes that the input is a T-pose, which limits the application of this method in pose transfer. Compared with these deep learning-based methods, our skeleton extraction network is free from the constraints of the T-pose or symmetric models. We predict the rigid skinning weights of the mesh bound to the skeleton and smooth the rigid weights by the diffusion equation based on the cotangent Laplacian operator. Then, high-quality LBS deformation results can be obtained.

3. Method

Given two meshes, one is the source mesh $\mathcal{S} = <V, F>$, and the other is the reference mesh $\mathcal{R} = <Q, F^r>$, where $V \in \mathbb{R}^{3\times n}$ and $Q \in \mathbb{R}^{3\times n_r}$ are the vertex set of the source mesh and the reference mesh; F and F^r are the corresponding faces. The result of pose transfer is the target mesh

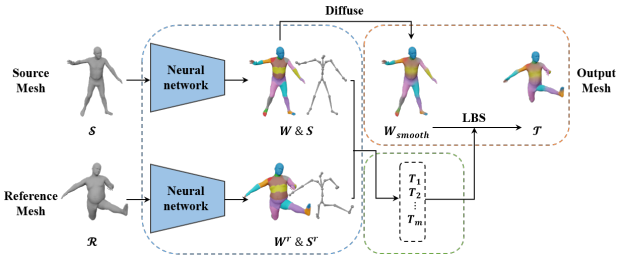

Figure 1. Method overview. Our neural network extracts the skeletons and binds the rigid weights of the source model and the reference model separately. The rigid weights are used to obtain the smooth weights by solving a diffusion equation, the skeleton is used to calculate the transformation matrix of pose deformation. Finally, the pose transfer is accomplished by deforming the source model into the target model.

$\mathcal{T} = <U, F>, U \in \mathbb{R}^{3\times n}$. The pose of the target mesh should be similar to the reference mesh, and the shape details should be the same as the source mesh.

The overview of our pose transfer method is depicted in Fig. 1. First, the source mesh and the reference mesh are fed to the neural network respectively to obtain the rigid weights $W \in \mathbb{R}^{n\times m}, W^r \in \mathbb{R}^{n^r\times m}$ and the extracted joints $C \in \mathbb{R}^{3\times m}$, $C^r \in \mathbb{R}^{3\times m}$. For convenience, this phase is called rigid weights skinning and joints regression (see the blue dotted box in Fig. 1).

Second, we construct the skeleton $S = <C, E>$ of the source mesh and the skeleton $S^r = <C^r, E>$ of the reference mesh. Pose transfer between the skeletons is simplified as the orientation consistency problem between bone vectors of the source model and the reference model. Then, we traverse the skeleton from the root joint and solve the global transformation matrix $T_1, T_2, \ldots, T_m \in \mathbb{R}^{4\times4}$ of each joint. These global transformation matrices deform the source mesh skeleton so that each bone vector is oriented in the same direction as the bone vector of the reference mesh skeleton. This phase is called skeleton pose transfer (see the green dotted box in Fig. 1).

Finally, we smooth the rigid weights W of the source mesh $\mathcal{S}$ by solving the diffusion equation. It can obtain appropriate smooth weights $W_{smooth} \in \mathbb{R}^{n\times m}$ by adjusting the parameters of the diffusion equation. With the global transformation matrix $T_1, T_2, \ldots, T_m$ and the skinning weight matrix $W_{smooth} \in \mathbb{R}^{n\times m}$, we deform the source mesh $\mathcal{S}$ via LBS (Linear Blend Skinning) to obtain target mesh $\mathcal{T}$. This phase is called LBS based on diffusion equation (see the orange dotted box in Fig. 1).

Each phase is described in detail below.

3.1. Rigid weights skinning and joints regression

We design a neural network based on a topology-aware edge convolution operator to bind rigid weights and extract skeleton joints. As shown in Fig. 2, the neural network consists of a rigid weight confidence module f_C and a joints regression module f_J. For a mesh, $M = <V, F>$, the output of the rigid weight confidence module f_C is the confidence of vertex bound to each joint, and the rigid weights $W \in \mathbb{R}^{n\times m}$ is calculated by the argmax operator. Then, the rigid weights W and the mesh M are inputted to the joints regression module f_J to predict the skeleton joints $C \in \mathbb{R}^{3\times m}$, and the skeleton $S = <C, E>$ is constructed according to the predefined skeletal hierarchy. The following subsections describe the details of the rigid weights confidence module f_C , the joints regression module f_J, and the topology-aware edge convolution operator GKEdgeConv.

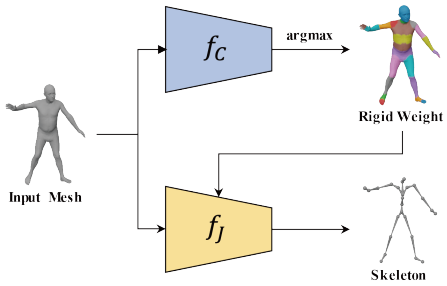

Figure 2. The overall architecture of network. The rigid weight confidence module f_C output the confidence of vertex bound to each joint. The joints regression module f_J predict the skeleton joints to construct skeleton.

3.1.1 Rigid weight confidence module

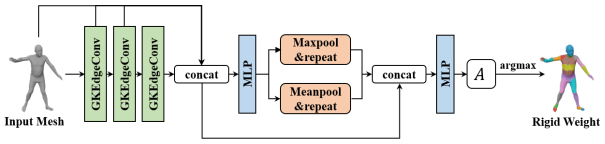

Figure 3. The structure of f_C. The task is similar to mesh segmentation, and GKEdgeConv layer can effectively encode local information of mesh vertices(It will be introduced in detail in subsection 3.1.3).

The structure of the rigid weight confidence module f_C is shown in Fig. 3. First, the mesh is fed to three-layer GKEdgeConv. Next, the global shape features of the mesh are encoded by the MLP layer and the pooling layer. Finally, combining local vertex features and global shape features, the rigid weight confidence matrix $A \in \mathbb{R}^{n\times m}$ is outputted by the last MLP layer.

$$A = (A_{ij}) = f_C\left(M; w_a\right),$$

where the learnable parameters of module f_C are denoted as w_a, and A_{ij} is the probability of vertex i bound to the joint j. According to matrix A, the rigid weight matrix $W = (W_{ij})$ can be computed as follows:

$$W_{ij} = \begin{cases} 1, & \arg\max_{\forall x \in C} A_i(x) = j \\ 0, & others \end{cases}. \quad (1)$$

Unlike the smooth weights, the value of the rigid weight is either 0 or 1, and any vertex v_i would be bound to the most relevant joint. As shown in Fig. 3, we visualize the rigid weights. In section 3.3, we will smooth the rigid weights based on the diffusion equation to obtain the smooth weights that can be used for LBS deformation.

3.1.2 Joints regression module

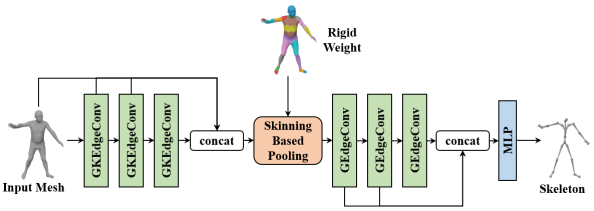

Figure 4. The structure of f_J. The skinning-based pooling is inspired by NBS [29]. The difference with NBS is that we use rigid weights instead of smooth weights to collapse vertices features into the joints' features set C'.

Fig. 4 displays the structure of the joints regression module f_J. Through the first three-layer GKEdgeConv convolution layer, the deep vertex representation matrix $V' \in \mathbb{R}^{n \times k}$ with k channels is obtained. The jth joint's high-dimensional features $C'_j \in \mathbb{R}^k$ is computed as follows:

$$C'_j = \frac{\sum_{i=1}^{n} W_{ij} V'_i}{\sum_{i=1}^{n} W_{ij}}.$$

In this way, only the vertices associated with each joint are involved in the computation of its deep features so that the network can better learn the joint positions. More details can be found in [29].

We apply the edge convolution operator GEdgeConv that only retains the topology-aware part to reduce the dimension of the deep joints features to get the 3D joints coordinate matrix C:

$$C = f_J(M, W; w_c),$$

where w_c are learnable parameters. According to the output of f_J and the fixed skeletal topology, we can construct the skeleton tree $S = <C, E>$.

3.1.3 Topology-aware edge convolution operator

To make the network better learn the local information of mesh or skeleton, we use the edge convolution in DGCNN [41] to extract the local features of the vertices and joints. For vertex v, the edge convolution x'_v encodes the global features x_v and the local features $x_u - x_v$ with a multilayer perceptron:

$$x'_v = \max_{u \in N(v)} MLP(x_v, x_u - x_v; w_{mlp}),$$

where w_{mlp} are learnable parameters of the MLP. $N(v)$ is the neighborhood of the vertex v. The key to the edge convolution operator is the construction of the neighborhood $N(v)$. DGCNN uses the k-nearest neighbor strategy to search the neighborhood $N(v)$, which means a pairwise distance matrix in feature space will be calculated, and the closest k vertices are taken as the vertex neighbors.

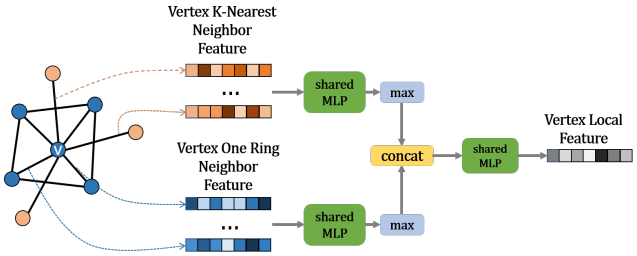

Figure 5. GKEdgeConv Layer. The edge convolution is inspired by RigNet [42], we use the k-nearest neighborhood instead of the geodesic neighborhood in RigNet.

As shown in Fig. 5, Our vertex neighborhoods are constructed by two strategies, one is the mesh topology one-ring neighborhood $N_g(v)$, and the other is the k-nearest neighborhood $N_k(v)$. We call the edge convolution operator GKEdgeConv with such two type neighborhoods. The one-ring neighborhood $N_g(v)$ is static, determined by the mesh's topology, and not affected by feature space changes. On the other hand, the k-nearest neighborhood $N_k(v)$ is a dynamic neighborhood. The vertex features encoded by the one-ring strategy are denoted as $x'_{v,g}$, the vertex features encoded by the k-nearest strategy are denoted as $x'_{v,k}$, the output features of the edge convolution operator are denoted as x'_v, then GKEdgeConv can be expressed as:

$$x'_{v,k} = \max_{u \in N_k(v)} MLP(x_v, x_u - x_v; w_k), \quad (2)$$

$$x'_{v,g} = \max_{u \in N_g(v)} MLP(x_v, x_u - x_v; w_g), \quad (3)$$

$$x'_v = MLP\left(concat\left(x'_{v,k}, x'_{v,g}\right); w_{concat}\right), \quad (4)$$

where w_k, w_g and w_{concat} are the learnable parameters. Eq. 2 and Eq. 3 represent the edge convolution operation for encoding dynamic k-nearest neighbor features and static

topology one-ring features, respectively. Eq. 4 is the MLP operation after concatenate $x'_{v,k}$ and $x'_{v,g}$. The parameters w_{concat} of the MLP layer trade off the importance of the topology-aware feature $x'_{v,k}$ and the geometry-aware feature $x'_{v,g}$.

In the joints regression module, since the number of joints is small, only the one-ring features are sufficient to get good results. We use the topology one-ring edge convolution operation expressed in Eq. 3 (that is GEdgeConv in Fig. 4) to learn the features of the joints.

3.1.4 Training detail

We first pre-train modules f_C and f_J separately. Then two modules are trained simultaneously to finetune the parameters w_a and w_c. More detailed information about the dataset will be introduced in section 4.

The binding of rigid skinning weights is similar to the mesh segmentation problem, so we train the module f_C under the supervision of classical softmax cross-entropy loss. First, denote the ground truth of rigid skinning weights as $\hat{W} \in \mathbb{R}^{n\times m}$. The probability matrix $A \in \mathbb{R}^{n\times m}$ is the output of the module f_C and is transformed into a probability distribution by the softmax operator. The loss function L_a is the cross-entropy loss as follows:

$$L_a = \frac{1}{n}\sum_{i}^{n}\sum_{j}^{m}\hat{W}_{ij}\log\left(softmax\left(A_{ij}\right)\right). \tag{5}$$

For the joints regression module f_J, the pre-training loss is the mean square error L_c:

$$L_c = \left\|\hat{C} - C\right\|^2, \tag{6}$$

where $\hat{C} \in \mathbb{R}^{3\times m}$ is the ground truth joints matrix. Since the quality of the rigid weights used for skinning-based pooling would greatly influence the prediction accuracy of the module f_J, we pre-train the joints regression module with ground truth weights $\hat{W}$.

After pre-training, the rigid weight matrix W obtained from the module f_C is fed to the module f_J, and the two modules f_C and f_J are trained at the same time to finetune the parameters learned in the pre-training phase. Eq. 5 and Eq. 6 are the supervision of the network.

3.2. Skeleton pose transfer

After predicting the source joints C and the reference joints C^r from the network, we construct the source skeleton tree S and the reference skeleton tree S^r according to the predefined skeletal topology. In this way, the pose of the mesh is represented by the corresponding skeleton. Finally, pose transfer is converted into solving the global transformation matrices $T_1, T_2, \ldots, T_m$. These transformation matrices transform the source skeleton S to the target skeleton; meanwhile, the orientation of each bone in the target skeleton should be the same as the skeleton S^r.

Algorithm 1 skeleton pose transfer

Input: S, S^r, W, W^r, V, Q
Output: $T_1, T_2, \ldots, T_m$

1: Calculate the rotation matrix $R_1, R_2, \ldots, R_m$ according to the number of child joints by SVD or Rodriguez formula.
2: Initialize the global transformation matrix of the root joint as $\begin{bmatrix} R_{root} & 0 \\ 0 & 1 \end{bmatrix}$
3: Traverse the skeleton according to its topology and calculate the translation vectors $t_1, t_2, \ldots, t_m$ by Eq. 9
4: Construct the transformation matrixes $T_1, T_2, \ldots, T_m$ of each joint.
5: **return** $T_1, T_2, \ldots, T_m$;

The global transformation matrix $T_i \in \mathbb{R}^{4\times4}$, corresponding to the joint C_i, includes a rotation matrix $R_i \in \mathbb{R}^{3\times3}$ and a translation vector $t_i \in \mathbb{R}^{3\times1}$. The rotation matrix guarantees that the orientation of each bone is consistent with the reference bone, and the translation vector guarantees that the starting joint of each bone is the end joint of its parent bone after deformation.

We first solve the rotation matrix R_i. The rotation matrix can be obtained by computing the rotation of the bone vectors between the source and reference skeletons. The bone vector points from joint C_i to its child. Let the child(i) denote the number of children of C_i, and there are the following cases in our predefined skeletal topology:

(1) When C_i is a root joint or chest joint, child$(i) = 3$. There are three bone vectors in both source and reference, so we apply SVD to solve the rotation. The details of SVD can be found in [36].

(2) When C_i is a general joint, $child\left(i\right) = 1$. We solve the rotation between two vectors according to the Rodriguez formula.

(3) When C_i is a leaf joint, child$(i) = 0$. Since no child can form a vector with C_i, we add a virtual child $C_{virtual} \in \mathbb{R}^{3\times1}$ additionally. The virtual child joint is calculated by the vertices bound to C_i as follows:

$$C_{virtual} = \frac{VW_i}{n(i)}, \tag{7}$$

where $W_i \in \mathbb{R}^{n\times1}$ is the rigid weights of all vertices bound to C_i, which can be obtained from the rigid weight matrix W directly. The denominator $n(i)$ is the number of vertices bound to C_i. The next step is as same as case (2).

Then we can solve the translation vector t_i. First, the

transformed joint C_i' is computed:

$$\begin{bmatrix} C_i' \\ 1 \end{bmatrix} = T_{p(i)} \begin{bmatrix} C_i \\ 1 \end{bmatrix}, \tag{8}$$

where $p(i)$ is the parent of the joint C_i. Starting from the root joint, according to the hierarchy of the skeleton, the translation vector of each joint is:

$$t_i = C_i' - R_i C_i. \tag{9}$$

In particular, the translation vector of the root joint is $\mathbf{0}$, which represents the model as a whole is not translated.

The whole process of skeleton pose transfer is shown in Alg. 1.

3.3. LBS based on diffusion equation

Given the skinning weight matrix W and the transformation matrixes $T_1, T_2, \ldots, T_m$, we deform the source model $\mathcal{S}$ to obtain the target model $\mathcal{T}$ by LBS:

$$\begin{bmatrix} u_i \\ 1 \end{bmatrix} = \sum_{j=1}^{m} W_{ij} T_j \begin{bmatrix} v_i \\ 1 \end{bmatrix}, \quad i = 1, \ldots, n \tag{10}$$

The deformed mesh will be torn if LBS with rigid weights is directly applied. So, we smooth the rigid weights by solving the diffusion equation. The diffusion equation, a second-order linear partial differential equation, can smooth the time-dependent process for a given signal value [10]. Desbrun *et al*. [13] gave the representation of the diffusion equation based on the backward Euler method:

$$(\mathbb{I} - dt\lambda L) f(t + dt) = f(t),$$

where L is the Laplace-Beltrami operator, λ is the diffusion coefficient, and $f(t)$ is the function about time t to be smoothed. The rigid weights W are regarded as a function to be smoothed over time on the mesh, and the smoothed weights W_{smooth} can be expressed as:

$$W_{smooth} = (\mathbb{I} - \lambda L)^{-1} W.$$

W_{smooth} can be seen as the blending of the weights W according to the heat diffusion principle [3], and the blending coefficients depend on the discrete Laplace operator L. The cotangent Laplace operator is an approximation of the mean curvature normal to the mesh vertices, which can better reflect the mesh geometry [13]. In this paper, we use the cotangent Laplace operator.

The diffusion coefficient λ determines the degree of weight smoothing. As shown in the first row of Fig. 6. The first column of the second row shows the result of LBS deformation using rigid weights, and there is a noticeable tear at the elbow. We replace the rigid weight in Eq. 10 with the smoothed weights W_{smooth}. For $\lambda = 5$, there are still some artifacts compared with GT at the elbow. For $\lambda = 20$, the deformation of the elbow is more similar to GT. The volume of the elbow is shrunken significantly for $\lambda = 35$.

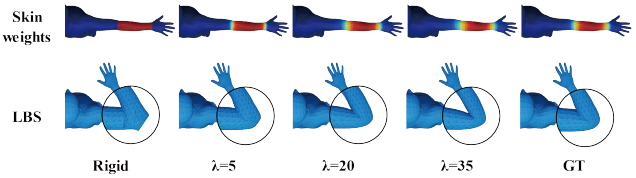

Figure 6. Qualitative evaluation of LBS deformation under different λ smooth skin weights. Red indicates the weight is 1, blue indicates the weight is 0, and the GT skinning weights are the standard weights of the SMPL [30] model. The skinning weights at the boundary gradually become smooth as λ increases

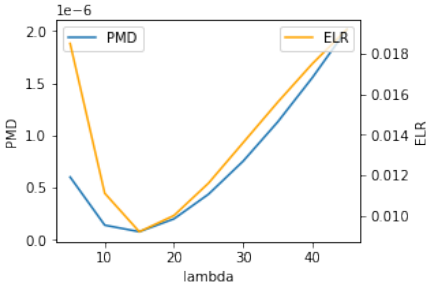

Figure 7. Quantitative evaluation of LBS deformation under different λ smooth skin weights. The evaluation metrics are Pointwise Mesh Euclidean Distance (PMD) and the Edges Length Radio (ELR), see subsection 4.1.

The hyper-parameter λ is unfixed. There are different optimal values for the different poses of the LBS deformation process. Fig. 7 shows the quantitative evaluation of Fig. 6. It can be seen that the error between the deformation and GT is minimum when λ is between 10 and 20. For convenience, the value of λ is set as 20 in all experiments.

4. Experiments

In this section, we first introduce the implementation details, such as the experimental environment and dataset. Then, the results of skinning weight binding, skeleton extraction, and pose transfer on SMPL and non-SMPL mesh are evaluated. Finally, the effectiveness of each module is demonstrated through an ablation study.

4.1. Implementation details

Environment: The experiments are conducted on NVIDIA Geforce RTX 3090 GPU(24 GB) and Intel Core i7-11700K/3.6GHz, CPU(32GB RAM). We implemented our network model with Pytorch, Numpy and other libraries.

Dataset: The currently popular SMPL models [30] are used to construct the training and testing datasets. Specifically, for the training phase, 5000 pairs of pose and shape parameters of SMPL are randomly generated in each epoch. The vertices coordinates, skeletons, and skinning weights are taken as ground truth. The order of vertices is randomly shuffled (correspondingly, the skinning weights and the face indexes are reconstructed). In particular, since the skinning weights of SMPL are smooth, we convert the smooth weights into rigid weights using Eq. 1. The same method is applied for the test dataset to generate 50,000 pairs of reference and source meshes randomly. The ground truth of pose transfer is generated through the source shape parameters and the reference pose parameters. In addition to SMPL, we also tested our method in SMAL [47], FAUST [8], DYNA [33], and MG-dataset [7].

Training: For the pre-training, the batch size is set to 8, the learning rate is 1e-4, and the epochs are 100. In the fine-tuning phase, the batch size is 4, the learning rate is reduced to 5e-5, and the epochs are set to 50. The cosine annealing learning rate updates the learning rates of both phases, and the network parameters are updated using Adam [22].

Metrics: We use PMD and ELR for evaluation. Let the predicted mesh be $M = (V, F)$ and GT is $\hat{M} = \left(\hat{V}, F\right)$:

$$PMD = \frac{1}{|V|} \sum_{i} \left\| V_i - \hat{V}_i \right\|_2^2,$$

$$ELR = \frac{1}{|V|} \sum_{i} \sum_{j \in N(i)} \left| 1 - \frac{\|V_i - V_j\|_2^2}{\left\|\hat{V}_i - \hat{V}_j\right\|_2^2} \right|,$$

where PMD measures the corresponding vertex coordinates difference between the predicted mesh and the GT, and ELR measures local details. If PMD is small and ELR is large, it indicates that the global pose of the mesh is sufficiently learned but without enough details.

4.2. Experimental results

This section shows the results of skeleton extraction and skinning weight binding, pose transfer, ablation study, and time efficiency. The experiments of skeleton extraction and skinning weight binding mainly compared with NBS [29]. We compare our pose transfer results with that of NPT [39] and 3D-CoreNet [35]. For NPT and 3D-CoreNet, we use the methods provided by the authors for training.

4.2.1 Skeleton extraction and skinning weight binding

The latest automatic rigging and skinning methods are RigNet [42] and NBS [29]. However, RigNet cannot guarantee the generation of the same skeleton hierarchy for different meshes, so we mainly compare our results with NBS.

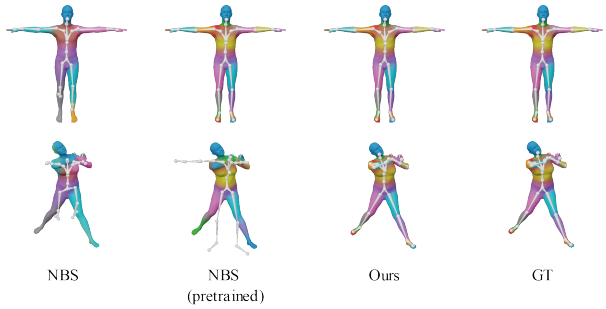

Figure 8. Comparison of rigging and skinning weights under different pose. We visualized the predicted rigging and skinning weights, the first row is T-pose model, the second row is non-T-pose model.

Avg Length	Max Std	Avg Std
0.2008	0.0106	0.0061

Table 1. Evaluation of bone length under different poses of the same model. The average length of the bone can be seen as a baseline. The maximum and average standard deviation of bone length, much smaller than the baseline, indicates that our method obtained stable bones.

Fig. 8 shows the skeleton extraction and weight-binding results of NBS and our method. The different colors on the mesh indicate the binding weights of different joints. The pose in the first row is a common T-pose in mesh skinning, and the results of our method and NBS are both satisfying. The pose of the second row, which is an uncertain pose in the pose transfer problem, is irregular. It can be seen that there is an apparent difference between the result of NBS and the GT. The first of the second row is the result that we add the random pose meshes to re-train, and it is not better than that of the pre-trained model(the second of the second row). The network of NBS is trained by indirect supervision of the LBS deformation results, and LBS is usually deformed from T-pose. When the initial pose is unfixed, it is difficult for the model to converge. By comparison, our result is more similar to the GT. We train the skeleton extraction and rigid skin weight network by direct supervision and smooth the rigid weights by the diffusion equation. The results of our method are both satisfying for T-pose and non-T-pose.

Besides, we extract the skeletons of the meshes that are the same shape with different poses and compare the length changes of the same bone. As shown in Table 1, when the average length of the bone is 0.2008, the maximum standard deviation is 0.0106 among all bones, and the average standard deviation is only 0.0061, which indicates that the bones obtained by our method are almost the same for the meshes with different poses and the same shape. It is also

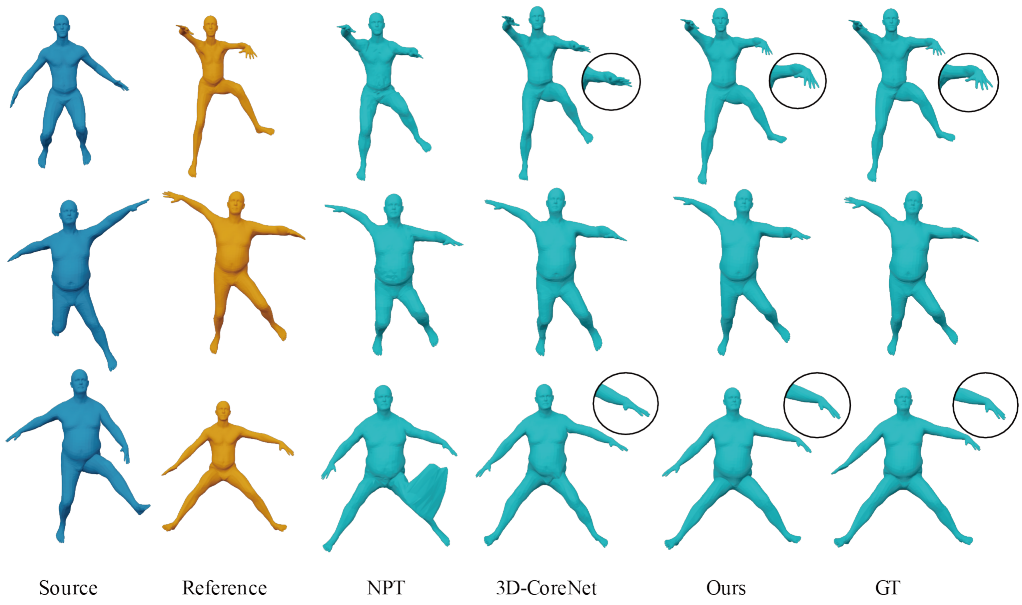

Figure 9. Qualitative comparison of pose transfer results on the SMPL dataset. We add zoomed wrist figures in the first and third rows. Note the orientation of the hand.

consistent with the property that the mesh pose does not affect the length of the bones.

4.2.2 Pose transfer results of SMPL meshes

Fig. 9 displays the pose transfer results of NPT, 3D-CoreNet, and our method. There are distorted results of NPT, such as the abdomen in the first two rows and the left leg in the third row. The details of 3D-CoreNet's results are better than that of NPT, but some poses are insufficient, such as the wrist in the first and third rows, and in the third row, there are adhesions between the left leg and the abdomen.

The quantitative results are shown in Table 2. One can see that our method's similarity metrics PMD and CD are better than that of NPT and 3D-CoreNet, which means the pose of our target mesh is more accurate. Our method's local detail metrics ELR and Max Dist are also much smaller than other methods, which indicates that the mesh details can be kept better, and there are almost no unnatural bumps or depressions on the mesh surface.

4.2.3 Pose transfer results of non-SMPL meshes

In this section, we test our model in different 3D model datasets DYNA, SMAL, FAUST, and MG-dataset, to evaluate the robustness of our method.

The human body meshes in DYNA and FAUST are obtained by matching 3D scan sequences with templates. The number of mesh vertices in these two datasets is the same as that of the SMPL model, which is 6890. The MG-dataset

	PMD(10^{-4}) ↓	ELR(10^{-1}) ↓	CD(10^{-1}) ↓	Max Dist(10^{-1}) ↓
NPT	5.41	3.62	2.36	9.94
3D-CoreNet	5.86	2.68	1.73	4.42
Ours	**4.58**	**1.40**	**0.17**	**0.93**

Table 2. Quantitative comparison of pose transfer results on the SMPL dataset. PMD and ELR were defined previously. The small Chamfer Distance (CD) indicates that the two point sets are similar. Max Dist is the maximum value of the distance between the corresponding vertices, which reflects whether there are unnatural bumps or depressions on the mesh surface.

is a dressing model based on SMPL. The number of MG-dataset standard model vertices is 27554, and the topological relationship of the first 6890 vertices is the same as that of the SMPL model. The skinning weights of the SMPL model are mapped to the remaining vertices by a weight mapping matrix. We take only the first 6890 vertices to predict joints and rigid weights for MG-dataset and map them to the remaining vertices using the same method.

The pose transfer results on DYNA and FAUST are shown in Fig. 10. The source mesh and the reference mesh are both natural human poses. Though the pose and the shape of these models are quite different from that of the SMPL model, the results are still satisfying.

Fig. 11 shows the results of pose transfer on MG-dataset. It can be seen that the details of the clothing are kept well. Moreover, in the fourth column, the source mesh's hands and legs overlap, but our result is still good.

SMAL is a parametric animal model with 3889 vertices

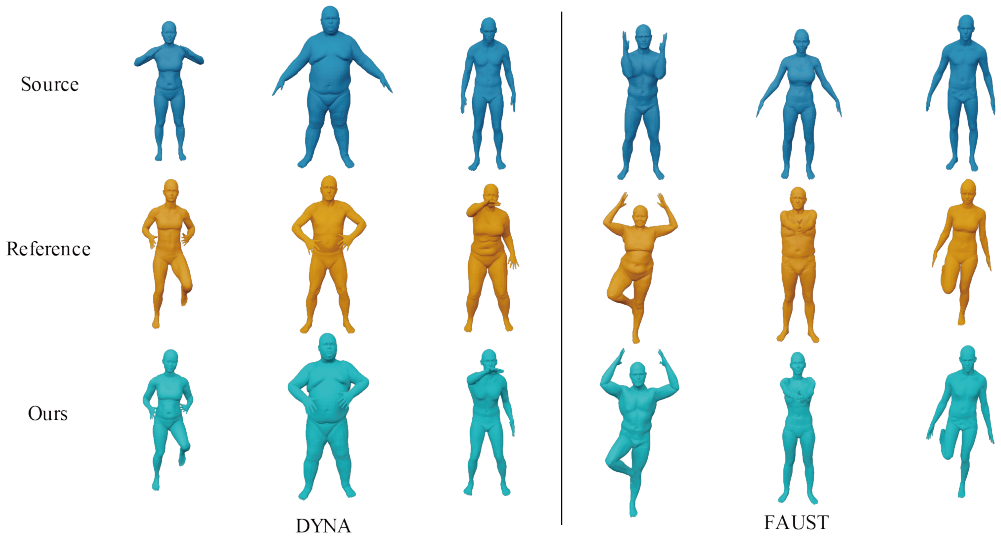

Figure 10. Pose transfer results on the DYNA and FAUST.

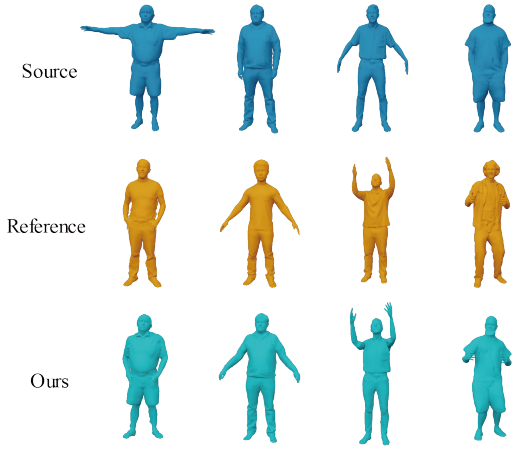

Figure 11. Pose transfer results on MG-dataset.

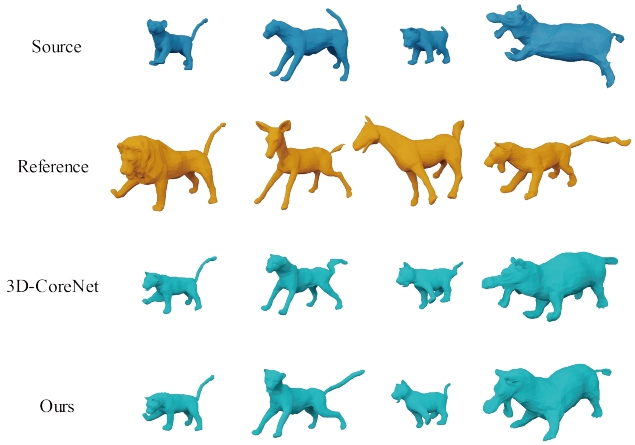

Figure 12. Pose transfer results on SMAL dataset.

and 33 joints. It includes five major animal classes: cat, dog, horse, cow, and hippopotamus. We use the same method to generate SMAL meshes for training and testing. The results of 3D-CoreNet are obtained from the pre-trained model provided by the authors, as shown in the third row of Fig. 12. One can see that the poses of the tails are incorrect from the second to the fourth, and there is an unnatural stretch at the root of the right front leg in the first. The models obtained by our method are better than that of 3D-CoreNet.

To test the robustness to noise, we add random noise to the vertices of the reference model, as shown in the fourth column of Fig. 13. In the third and fifth columns, we can see whether there is noise or not, and the generated target meshes obtained by our method are mostly the same.

In summary, although our model is trained on the SMPL dataset, we can also get the target mesh with accurate pose and detail on the non-SMPL human dataset. In addition, the model can be extended to non-human meshes if there is a suitable dataset like SMAL.

4.2.4 Ablation study

To demonstrate the effectiveness of the several components of our pose transfer method, we tested the performance without the network topology-aware module (one-ring neighbor features extraction module) and the performance without the diffusion equation smoothing rigid

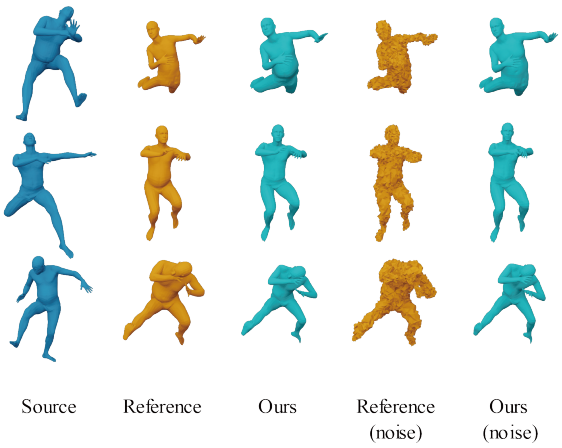

Figure 13. Robustness to noise.

weights separately. Table 3 displays the experiment results. Full and w/o denote the results with all modules and the results without some modules, respectively. $N_g(v)$ and diff denote the one-ring neighbor features extraction module and the smoothing rigid weight module based on the diffusion equation. From Table 3, the missing one-ring neighbor features extraction module impacts the PMD loss obviously. On the other hand, ELR loss would increase spectacularly if the smoothing rigid weights module is missed.

	PMD(10^{-4}) ↓	ELR(10^{-1}) ↓
w/o $N_g(v)$	4.92	1.54
w/o diff	4.88	2.81
Full	**4.58**	**1.40**

Table 3. Ablation experiments.

4.2.5 Time efficiency

Table 4 shows the time efficiencies of different methods. The source and reference meshes used for testing are SMPL models with 6890 vertices. Our method is not based on an end-to-end neural network model, and we give the time of the neural network part (network only) for extracting the skeleton and the rigid weights and the time of the whole pose transfer process (full). Including the source mesh and the reference mesh, the time of the neural network part (network only) is 589.9ms, which accounts for about 52% of the full time, and the inference time of the individual mesh is about 294.95ms. The main time-consuming is that GKEdgeConv dynamically computes the k-nearest neighbors of each vertex at each layer. In addition, we use a dense matrix in our experiments to solve the large sparse linear diffusion equation, which is also time-consuming. Compared with NPT and 3D-CoreNet, our method sacrifices some efficiency. However, it improves the quality of the pose transfer results, and the number of trainable parameters of the network is much smaller than the former two.

	Parameters(M)↓	Time(ms)↓
NPT	6.06	**42.02**
3D-CoreNet	24.46	128.18
Our(network only)	**3.38**	589.90
Our(full)	**3.38**	1130.26

Table 4. Time efficiency.

4.2.6 Generalization extension

Although the number of mesh vertices is not limited in our network, our performance would be significantly affected if there is too much difference from the number of vertices in the training dataset. This's because the number of vertices affects the computation of the k-nearest neighborhood in DGCNN [41]. The use of neighborhood information produces better performance in our method, but it is also influenced by the number of vertices and vertex density.

To reduce the effect of vertex number and vertex density, we simplify the vertices of SMPL to 2048 points to construct a dataset and retrain the network. The mapping between the simplified mesh and the original mesh is recorded, and the network output will be mapped back to the original mesh for subsequent processing. The process is similar to normalizing the input data.

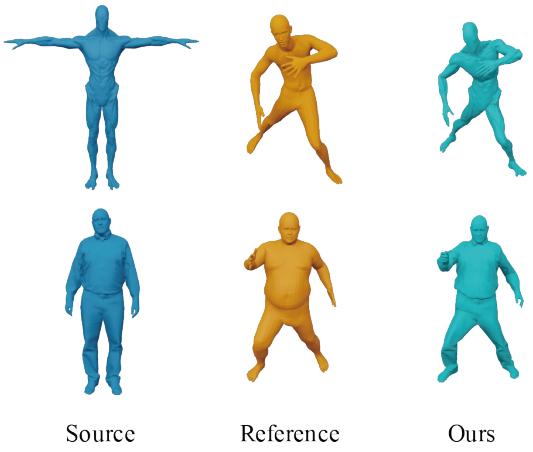

Figure 14. Pose transfer on meshes with large difference in the number of vertices.

Fig. 14 displays the results. The reference poses are from the SMPL dataset. The source mesh in the first row

is from the Mixamo dataset [1] (18453 vertices), an artist-made mesh of a monster that differs dramatically from the meshes of SMPL. Our result is satisfying, and the reference pose is transferred to the source. The model in the second row is from MG-dataset (27554 vertices). This result is also satisfying.

5. Conclusion

In this paper, we propose a 3D mesh pose transfer method based on skeletal deformation. We predict the source and reference mesh's rigid weights and skeleton joints through GKEdgeConv-based neural network. Then, pose transfer is reformulated as a deformation problem of the source mesh. We further smooth the rigid weights by the diffusion equation to obtain high-quality LBS results, and fully automatic pose transfer without human interaction is achieved. Compared with the existing deep learning-based pose transfer methods, the details and the poses of our experiment results on different datasets are better.

However, the neural network requires many datasets with skeletons and weights for training. We have trained only on the human dataset SMPL and the tetrapod dataset SMAL, respectively. The lack of suitable training data may limit the use of the method. In addition, only the corresponding bones of the source and reference models are used as two vectors to calculate the rotation, resulting in the twisted rotation along the skeleton axis being lost, which may lead to insufficient pose transfer. As shown in Fig. 15, the poses of the heads and the left hand are not exactly the same as the reference meshes.

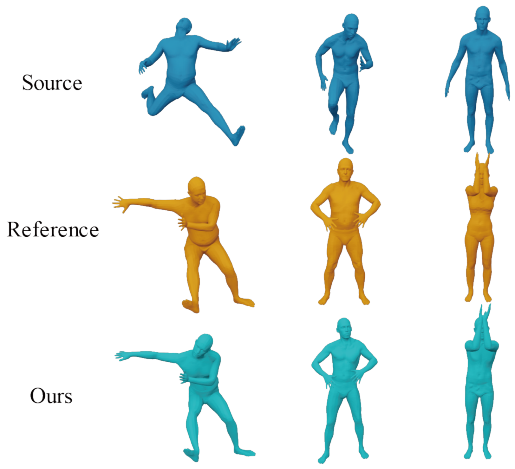

Figure 15. The results with limitations.

References

[1] Adobe Systems Inc. Mixamo. https://www.mixamo.com, 2018. 12

[2] S. Bang and S.-H. Lee. Spline interface for intuitive skinning weight editing. *ACM Transactions on Graphics (TOG)*, 37(5):1–14, 2018. 3

[3] I. Baran and J. Popović. Automatic rigging and animation of 3d characters. *ACM Transactions on graphics (TOG)*, 26(3):72–es, 2007. 3, 7

[4] I. Baran, D. Vlasic, E. Grinspun, and J. Popović. Semantic deformation transfer. In *ACM SIGGRAPH 2009 papers*, pages 1–6. 2009. 2

[5] J. Basset, A. Boukhayma, S. Wuhrer, F. Multon, and E. Boyer. Neural human deformation transfer. In *2021 International Conference on 3D Vision (3DV)*, pages 545–554. IEEE, 2021. 2

[6] G. Bharaj, T. Thormählen, H.-P. Seidel, and C. Theobalt. Automatically rigging multi-component characters. In *Computer Graphics Forum*, volume 31, pages 755–764. Wiley Online Library, 2012. 3

[7] B. L. Bhatnagar, G. Tiwari, C. Theobalt, and G. Pons-Moll. Multi-garment net: Learning to dress 3d people from images. In *proceedings of the IEEE/CVF international conference on computer vision*, pages 5420–5430, 2019. 8

[8] F. Bogo, J. Romero, M. Loper, and M. J. Black. Faust: Dataset and evaluation for 3d mesh registration. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3794–3801, 2014. 8

[9] M. Botsch and L. Kobbelt. An intuitive framework for real-time freeform modeling. *ACM Transactions on Graphics (TOG)*, 23(3):630–634, 2004. 3

[10] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy. *Polygon mesh processing*. CRC press, 2010. 7

[11] J. Cao, A. Tagliasacchi, M. Olson, H. Zhang, and Z. Su. Point cloud skeletons via laplacian based contraction. In *2010 Shape Modeling International Conference*, pages 187–197. IEEE, 2010. 3

[12] H. Chen, H. Tang, H. Shi, W. Peng, N. Sebe, and G. Zhao. Intrinsic-extrinsic preserved gans for unsupervised 3d pose transfer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8630–8639, 2021. 2

[13] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324, 1999. 3, 7

[14] 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)*, 37(6):1–15, 2018. 2

[15] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(12):4338–4364, 2020. 1

[16] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. 3

[17] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017. 2

[18] A. Jacobson, I. Baran, J. Popovic, and O. Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 30(4):78, 2011. 3

[19] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki. Harmonic coordinates for character articulation. *ACM Transactions on Graphics (TOG)*, 26(3):71–es, 2007. 3

[20] L. Kavan, S. Collins, J. Žára, and C. O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG)*, 27(4):1–23, 2008. 3

[21] L. Kavan and J. Žára. Spherical blend skinning: a real-time deformation of articulated models. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 9–16, 2005. 3

[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 8

[23] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

[24] A. Kovnatsky, M. M. Bronstein, A. M. Bronstein, K. Glashoff, and R. Kimmel. Coupled quasi-harmonic bases. In *Computer Graphics Forum*, volume 32, pages 439–448. Wiley Online Library, 2013. 2

[25] B. H. Le and Z. Deng. Robust and accurate skeletal rigging from mesh sequences. *ACM Transactions on Graphics (TOG)*, 33(4):1–10, 2014. 3

[26] B. H. Le and J. K. Hodgins. Real-time skeletal skinning with optimized centers of rotation. *ACM Transactions on Graphics (TOG)*, 35(4):1–10, 2016. 3

[27] B. H. Le and J. Lewis. Direct delta mush skinning and variants. *ACM Trans. Graph.*, 38(4):113–1, 2019. 3

[28] B. Lévy. Laplace-beltrami eigenfunctions towards an algorithm that" understands" geometry. In *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, pages 13–13. IEEE, 2006. 2

[29] P. Li, K. Aberman, R. Hanocka, L. Liu, O. Sorkine-Hornung, and B. Chen. Learning skeletal articulations with neural blend shapes. *ACM Transactions on Graphics (TOG)*, 40(4):1–15, 2021. 3, 5, 8

[30] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015. 7, 8

[31] N. Magnenat-Thalmann, R. Laperrire, and D. Thalmann. Joint-dependent local deformations for hand animation and object grasping. In *In Proceedings on Graphics interface'88*. Citeseer, 1988. 3

[32] J. Mancewicz, M. L. Derksen, H. Rijpkema, and C. A. Wilson. Delta mush: smoothing deformations while preserving detail. In *Proceedings of the Fourth Symposium on Digital Production*, pages 7–11, 2014. 3

[33] 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)*, 34(4):1–14, 2015. 8

[34] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1, 2

[35] C. Song, J. Wei, R. Li, F. Liu, and G. Lin. 3d pose transfer with correspondence learning and mesh refinement. *Advances in Neural Information Processing Systems*, 34:3108–3120, 2021. 1, 2, 8

[36] O. Sorkine-Hornung and M. Rabinovich. Least-squares rigid motion using svd. *Computing*, 1(1):1–5, 2017. 6

[37] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23(3):399–405, 2004. 2

[38] K. Wampler. Fast and reliable example-based mesh ik for stylized deformations. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016. 3

[39] J. Wang, C. Wen, Y. Fu, H. Lin, T. Zou, X. Xue, and Y. Zhang. Neural pose transfer by spatially adaptive instance normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5831–5839, 2020. 1, 2, 8

[40] Y. Wang and J. Solomon. Fast quasi-harmonic weights for geometric data interpolation. *ACM Transactions on Graphics (TOG)*, 40(4):1–15, 2021. 3

[41] 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)*, 38(5):1–12, 2019. 2, 3, 5, 11

[42] Z. Xu, Y. Zhou, E. Kalogerakis, C. Landreth, and K. Singh. Rignet: Neural rigging for articulated characters. *arXiv preprint arXiv:2005.00559*, 2020. 3, 5, 8

[43] M. Yin, G. Li, H. Lu, Y. Ouyang, Z. Zhang, and C. Xian. Spectral pose transfer. *Computer Aided Geometric Design*, 35:82–94, 2015. 2

[44] 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*, 36(3):520–554, 2021. 1

[45] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003. 3

[46] X. Zou, G. Li, M. Yin, Y. Liu, and Y. Wang. Deformation-graph-driven and deformation aware spectral pose transfer. *Journal of Computer-aided Design & Computer Graphics*, 33:1234–1245, 2021. 2

[47] S. Zuffi, A. Kanazawa, D. W. Jacobs, and M. J. Black. 3d menagerie: Modeling the 3d shape and pose of animals. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6365–6373, 2017. 8