*electronics*

MDPI

*Article*

# Three-Dimensional Mesh Character Pose Transfer with Neural Sparse-Softmax Skinning Blending

**Siqi Liu [1], Mengxiao Yin [1,2,*], Ming Li [1], Feng Zhan [1,2] and Bei Hua [1,2]**

[1] School of Computer and Electronics Information, Guangxi University, Nanning 530004, China; 2213301034@st.gxu.edu.cn (S.L.); 1913392026@st.gxu.edu.cn (M.L.); zf@gxu.edu.cn (F.Z.); huabei111@163.com (B.H.)
[2] Guangxi Key Laboratory of Multimedia Communications and Network Technology, Nanning 530004, China
* Correspondence: ymx@gxu.edu.cn; Tel.: +86-138-7816-0430

**Abstract:** Three-dimensional mesh pose transfer transforms the pose of a source model into the pose of a reference model while preserving the source model's identity (body detail). It has tremendous potential in computer graphics tasks. Current neural network-based methods primarily focus on extracting pose and body features, not entirely using the articulated body structure of humans and animals. We propose an end-to-end pose transfer network based on skinning deformation to address these issues. This network first extracts skinning weights and model joint features. Then, they are decoded to transfer the source model to a pose similar to the reference model while preserving the features of the source model. During feature extraction, we utilize the features of the k-nearest neighborhoods and one-ring neighborhoods to enable the network to learn the body details of the model better. Additionally, we apply skinning weights and joint features to capture the variation in the source model pose compared to the reference model pose and then use a decoding network to obtain the target model, replacing linear blend skinning. We conducted experiments on datasets such as SMPL, SMAL, FAUST, DYNA, and the MG dataset to provide empirical evidence and demonstrate that our method is the best in quantitative experiments. Our method efficiently transfers poses while better preserving the identity of the source model.

**Keywords:** pose transfer; skinning deformation; skeleton extraction; neural network



Academic Editors: Martin Černý, Antonio G. Ravelo-Garcia, Morgado Dias and Ankit Gupta

Received: 11 January 2025
Revised: 30 January 2025
Accepted: 30 January 2025
Published: 1 February 2025

**Citation:** Liu, S.; Yin, M.; Li, M.; Zhan, F.; Hua, B. Three-Dimensional Mesh Character Pose Transfer with Neural Sparse-Softmax Skinning Blending. *Electronics* **2025**, *14*, 589. https://doi.org/10.3390/electronics14030589



## 1. Introduction

Deformation transfer and pose transfer are two techniques for reusing 3D models, and they hold significant research value in the field of computer graphics. These techniques can greatly reduce the workload involved in 3D animation production and virtual world modeling. As shown in [1–3], these techniques are widely used in high-fidelity 3D mesh modeling tasks. As described in [4] (blue dashed arrow in Figure 1), given two poses for the reference model (two cat models, one of which is similar to the pose of the lion source model and the other provides the pose that needs to be transformed into), deformation transfer uses the deformation process between the two reference models to drive a similar deformation of the source model (lion model). In contrast, pose transfer only requires a reference model that provides the pose to guide the deformation of the source model as in [5] (red arrow in Figure 1). Given a source model and a reference model, pose transfer outputs a target model with the source model's body details and a pose similar to the reference model.

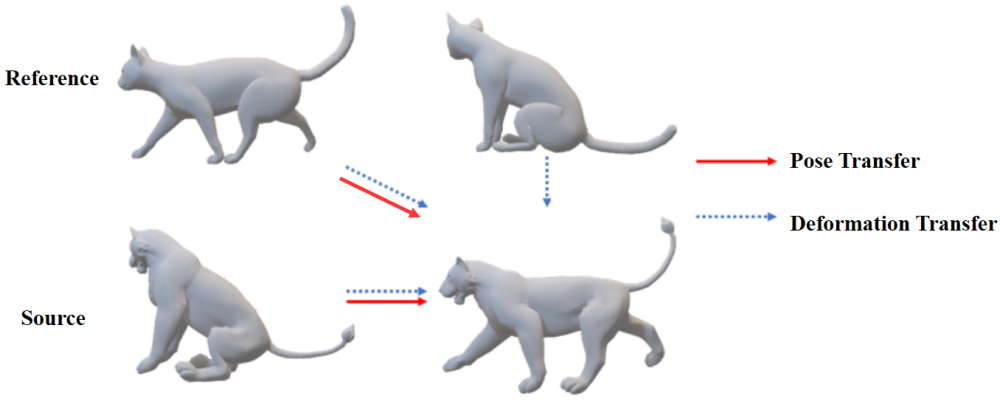


**Figure 1.** Deformation transfer and pose transfer. Deformation transfer requires two poses of the reference model to guide the deformation of the source model. In contrast, pose transfer requires only one reference model for the deformation of the source model.

In recent years, inspired by image style transfer, researchers such as Chen et al. [6] and Song et al. [7] have proposed end-to-end deep learning models for pose transfer, known as Neural Pose Transfer (NPT) [8]. This approach encodes the 3D model vertices into features in a latent space and utilizes a decoding module to generate target models with preserved details. However, as Kingma et al. show [9], this pose transfer method does not impose regularity constraints on the latent space, and the encoder–decoder architecture has a high degree of freedom. As shown in the results tested by Guo et al. [10] and Yang et al. [11], such neural networks cannot effectively capture the intrinsic connections between the vertex feature spaces of the source and reference models. The PMD metric for predictions outside the known shape and pose spaces is significantly higher than for poses fitted in the training set.

Although it is a simple and practical approach to compute transformation matrices for each grid vertex to obtain the final deformation result like LBS [12], this method can introduce issues such as "candy wrapping". Inspired by skinning deformation techniques, we propose an end-to-end pose transfer method based on neural skinning deformation, aiming to address this problem using neural networks. Unlike previous methods that directly encode the vertex's 3D coordinates into features in the latent space using an encoder, we map the vertex features to the skeleton latent space using skinning weights to obtain skeleton latent space features. Subsequently, vertex latent transformation features are obtained, and pose transfer is performed through a neural network, solving the problems faced by Yang et al. [11] as described above.

Our method consists of two stages: rigging and pose transfer. In the rigging stage, a topology-aware neural network is employed to predict the joints and corresponding skinning weights for both the source and reference mesh models. The predicted joints are used to encode the overall pose of the models, while the skinning weights map the model vertices to their respective joints. In the pose transfer stage, the vertex latent space features of the source and reference models are encoded into skeletal latent space features using skinning-based pooling. Then, an edge convolutional neural network further encodes the skeletal latent space features into skeletal latent transformations. These transformations are linearly combined with the skinning weights and then unpooled to obtain per-vertex latent transformation features of the source model. Finally, the network generates a target mesh model using these features. It has been observed that this approach encourages the network to integrate features from corresponding limb parts of the two models, resulting in a target model that exhibits both fine details and comprehensive pose learning. This paper is organized as follows: Section 2 discusses related work, Section 3 introduces the main

methods, Section 4 shares the training details, Section 5 shows the experimental results, and Section 6 gives the conclusion. Our main contributions can be summarized as follows:

(1) We propose a pose transfer method that utilizes neural networks as a replacement for linear blend skinning (LBS). By mapping the latent space features of two 3D models with no correspondence to the latent space features of a skeleton with an isomorphic relationship, we transform the pose transfer problem between models into a latent transformation learning problem between the corresponding nodes of the source model and the reference model with an isomorphic relationship. This approach results in highly accurate pose features, which are then used by a neural network to generate the target model with preserved details utilizing the concept of linear blend skinning.
(2) An improved neural weight binding method for mesh skinning was proposed by Yang et al. [11]. Sparse-Softmax is used to smooth the skinning weights, and KL divergence is employed for supervision, enabling the network to predict sparse and accurate smooth skinning weights.
(3) Experimental results demonstrate that our method achieves superior performance compared to other deep learning-based pose transfer methods on human and animal meshes from unknown datasets.

The source code is publicly available at https://github.com/lzszmp/3D-mesh-pose-transfer-based-on-deep-learning-skinning-deformation, accessed on 1 May 2024.

## 2. Related Work

This paper's method aims at the posture transfer of 3D models based on skin deformation, which mainly involves deformation technology, skin deformation technology, and digital geometry technology based on deep learning. The following briefly introduces the work closely related to this paper, such as deformation transfer and posture transfer.

### 2.1. Deformation Transfer

Given two different poses of a reference model and a source model, deformation transfer applies the geometric transformations of the reference model's different poses to the source model. Sumner et al. [4] utilized model correspondences as constraints to transfer the deformation gradients of the target model's triangular faces, achieving deformation transfer on triangular meshes. Baran et al. [13] proposed establishing semantic correspondences between non-corresponding parts of the reference model and the source model, enabling semantic deformation transfer. However, user specification is required for semantic correspondences. Kamel et al. [14] proposed a method for quantifying human body movements using three-dimensional joint coordinates and dividing joint motions into global and local movements for predicting full-body actions. Shen et al. [15] utilized a convolutional neural network-based pose estimation method to predict poses, which has great potential in deformation transfer tasks. Ahmad et al. [16] employed a combination of three compact CNN models, including the low-cost SparseNet, and a feature representation technique to extract features of three-dimensional objects. This approach can assist in achieving more accurate deformation transfer. Li et al. [17] defined rigid geometric transformations between k-nearest points of corresponding vertices as vertex deformation gradients, addressing the issue of deformation gradient calculation between two points in point cloud models, thereby extending deformation transfer to point cloud models. However, this method can result in tearing artifacts in the target model, requiring target model resampling. Deng et al. [18] employed the method of template assistance to achieve unsupervised processing of human body point clouds. Li et al. [19] applied transform to the prediction of human body poses. On the other hand, Gao et al. [20] utilized Generative Adversarial Networks (GANs) to map the deformed source shape to the deformed target

shape. Since both shapes exist in the latent space, this ensures similarity between the shape obtained from the mapping and the target shape. The authors also employed reverse mapping from the target to the source shape, incorporating cycle consistency loss. This VAE-Cycle GAN (VC-GAN) architecture establishes reliable mappings between shape spaces for high-quality transfer.

### *2.2. Pose Transfer*

Unlike deformation transfer, pose transfer focuses solely on the relationship between the source model and the reference model, resulting in a target model that closely resembles the reference model's pose while preserving the source model's geometric details. This technique is suitable for animal and human models with limbs. Kovnatsky et al. [21] improved upon the primary pose transfer method, as in Levy et al. [22], by utilizing optimized coupled quasi-harmonic bases. It allows for approximating models' poses with different connectivity, although it may result in losing local model details. Yin et al. [23] introduced a detail-preserving multi-level spectral pose transfer method that establishes a Laplacian editing framework. It employs frequency components and combines cage-based subspace techniques to address the pose transfer problem. However, manually selecting matched point pairs is necessary to construct the functional mapping.

With the increasing integration of neural network techniques and digital geometry, some researchers have explored neural network for pose transfer of 3D mesh models. Inspired by image style transfer methods such as those of Cosmo et al. [24], Zhuo et al. [25] and Wang et al. [8] proposed an end-to-end neural pose transfer model called NPT, which utilizes PointNet [26] to encode the pose of the reference model. It employs a spatially adaptive encoding layer to learn pose-invariant features and generate target models with similar poses. NPT is a vertex-shared weight network model based on PointNet, which cannot capture local details of the model vertices. As a result, it is sensitive to the distribution of vertices and may produce distorted target models when using untrained poses as input. Building upon NPT, Song et al. [27] recognized that vertex features have certain correspondences. They proposed a vertex feature matching approach based on optimal transport theory. They constructed a decoding module using an elastic singleton normalization layer to achieve higher accuracy in generating target models. However, this method still cannot capture the local details of the model vertices, thus failing to solve the fundamental sensitivity issue to vertex distribution in NPT. Yang et al. [11] combined with Wang et al. [28] to incorporate local features into pose transfer, addressing the sensitivity problem to vertex distribution. Wang et al. [29] also chose to use DGCNN to process human body point clouds. Chen et al. [30] designed a cycle reconstruction loss for self-supervised pose transfer, eliminating the need for real datasets. Sun et al. [31] introduced global pattern loss in the latent space to decouple pose and detail features and a loss for distinguishing local semantics, aiming to separately learn the pose and features of the model for high-quality pose transfer. Liu et al. [32] utilized cross-covariance attention to perceive the correlations between points at different distances to assist in completing the task.

## 3. Method

Given a source mesh model $M_s = (V^s, F^s)$ and a reference mesh model $M_r = (V^r, F^r)$, where $V$ represents the vertex coordinates and $F$ represents the face, the goal of pose transfer is to find a target mesh model $M_t = (V^t, F^s)$ that possesses the geometric details and topological structure of the source model $M_s$, as well as a similar pose to the reference model $M_r$ (where $V^s \in \mathbb{R}^{n^s \times 3}$, $V^r \in \mathbb{R}^{n^r \times 3}$, $V^t \in \mathbb{R}^{n^s \times 3}$ represent the vertices of the model, $F^s \in \mathbb{N}^{|F^s| \times 3}$ represents the face set of the source and target models, and $F^r \in \mathbb{N}^{|F^r| \times 3}$ represents the face set of the reference model). As shown in Figure 2, our method consists

of two main stages: the rigging stage and the pose transfer stage, corresponding to the rigging branch and the pose transfer branch in the network architecture, respectively.

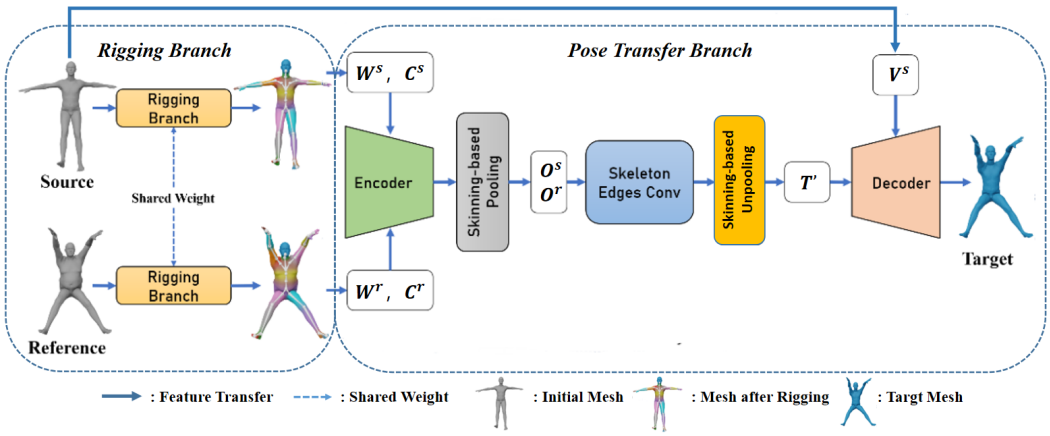


**Figure 2.** Our method overview. Our network contains two parts: the rigging branch extracts skinning weights and joint information from the input model, and the pose transfer branch performs pose transfer.

**Rigging**: As shown in the left part of Figure 2, in this stage, we were inspired by Yang et al. [11] and found that treating vertices as point clouds to extract local information is highly effective in tasks such as skeleton extraction and skin weight prediction, as this will improve the overall quality of the transformation. Using a similar feature extraction structure, the rigging branch of the network predicts the joints $C^s \in \mathbb{R}^{m\times 3}$ and $C^r \in \mathbb{R}^{m\times 3}$ with a predefined topological structure for both the source model $M_s$ and the reference model $M_r$, as well as the corresponding skinning weights $W^s \in \mathbb{R}^{|V^s|\times m}$ and $W^r \in \mathbb{R}^{|V^r|\times m}$. Based on edge convolutional neural networks, this module extracts topological and geometric k-nearest neighbor features from the mesh as local features for vertices, making it topology aware. However, unlike Yang et al. [11] who used regular Softmax to compute skin weights and then applied smoothing, we directly obtain smooth skin weights using Sparse-Softmax. Additionally, we improved the GKEdgeConv convolutional layer used by Yang et al. [11] to better predict skeleton information.

**Pose transfer**: As shown in the right part of Figure 2, in this stage, the source model $M_s$, reference model $M_r$, their skinning weights $W_s$ and $W_r$, and joints $C_s$ and $C_r$ are input into the second branch of the network for pose transfer to generate the target model $M_t$. This branch mainly includes skinning-based pooling, skeleton edge convolution, and skinning-based unpooling operations. In the pose transfer branch, the skinning-based pooling utilizes the skinning weights to map the unordered vertex features of the source and reference models to a skeleton latent space $O^s, O^r \in \mathbb{R}^{h\times m}$ with isomorphic relationships. This allows the use of skeleton edge convolution in the isomorphic skeleton latent space to extract the hidden transformation between the source model skeleton and the reference model skeleton. The skinning-based unpooling performs linear blending on the skeleton latent space to obtain the $h$-dimensional per-vertex transformation latent space $T' \in \mathbb{R}^{h\times n}$ for the source model. Then, a decoding module is used to decode and obtain the final target model $M_t$.

### *3.1. Rigging*

Given the mesh model $M = (V, F)(V \in \mathbb{R}^{n\times 3}, F \in \mathbb{N}^{|F|\times 3})$ (Since the source model and the reference model perform the same operations, $V$ and $F$ are used here to represent the vertices and faces of the two models uniformly), in this stage, the skinning binding branch is used to predict smooth skinning weights $W \in \mathbb{R}^{n\times m}$ and joints $C \in \mathbb{R}^{m\times 3}$. As

shown in Figure 3, this branch mainly consists of two parts: skinning weight binding and joint regression.

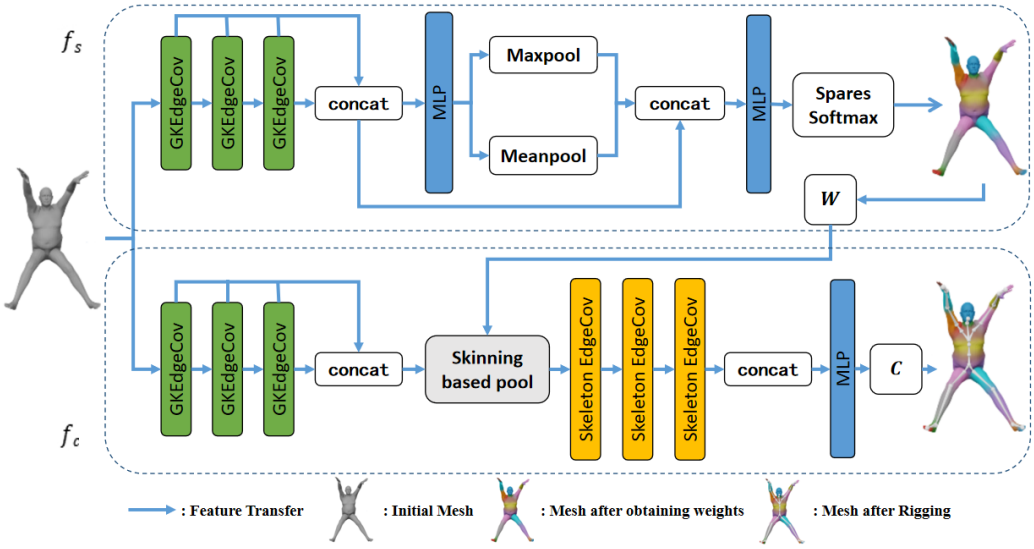


**Figure 3.** Rigging branch. The rigging branch extracts skinning weights $W$ and joint information $C$ from vertex information for subsequent pose transfer.

#### 3.1.1. Skinning Weights

We treat the mesh vertices as point clouds and extract features. Since the methods based on transformer used by and Zhao et al. [33] cannot effectively capture local features, we adopt a skin weight binding module similar to the structure proposed by Yang et al. [11], as shown in the upper part of Figure 3. The mesh model $M$ is inputted, and the GKEdgeConv convolutional layer [11] encodes per-vertex features. GKEdgeConv is a primary improvement of the edge convolution operator proposed in [28]. Specifically, for the input feature vectors $x_v$ of the previous layer, their k-nearest neighbors $N_k(v)$ and 1-ring neighbors $N(v)$ are constructed. A multi-layer perceptron encodes both global features $x_v$ and local features $x_u - x_v$:

$$x'_v = MLP(concat(x'_{v,k}, x'_{v,g}); w_{gk}), \tag{1}$$

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

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

where $N_k(v)$ represents the dynamic k-nearest neighbors of the vertex feature, which is reconstructed based on the feature space obtained from the previous layer before each convolution computation. $N_g(v)$ represents the static 1-ring neighbors of the vertex determined by the topological space of the mesh and is not affected by changes in the feature space. Concat operation denotes channel concatenation, where $w_k$, $w_g$, and $w_{gk}$ are learnable weights for these perceptrons. This strategy allows the network to simultaneously consider the dynamic geometric features $x'_{v,k}$ in Euclidean space and the static topological features $x'_{v,g}$ of the mesh. The final MLP layer processes the combined features, balancing the importance of topological perception features $x'_{v,g}$ relative to geometric perception features $x'_{v,k}$.

After three layers of GKEdgeConv encoding, the output features are encoded by the intermediate MLP layer and pooling layer to capture the global shape features of the mesh.

The local features of the vertices are combined with the global features and passed through the final MLP network to output per-vertex confidence matrix $A \in \mathbb{R}^{N \times m}$:

$$A = (A_{ij}) = f_s(M; w_s), \tag{4}$$

where $w_s$ represents the learnable parameters of module $f_s$, and $A_{ij}$ represents the confidence level from the $i$-th vertex to the $j$-th joint.

Yang et al. [11] use the softmax function to convert the predicted per-vertex confidence into a probability distribution, obtaining the final skinning weights. We found that the skinning weights obtained from softmax output are excessively "smooth", resulting in vertices being bound to unrelated joints and failing to achieve sparse skinning weights. The accumulation of these smaller weights significantly impacts the overall accuracy of the skinning weights, leading to a chain reaction and making it challenging for subsequent joint regression and latent space encoding. Therefore, in the prediction stage, we switch to using Sparse-Softmax to smooth the per-vertex confidence:

$$W_{ij} = Sparse\text{-}Softmax(A_{ij}) = \frac{\sigma(A_{ij})e^{A_{ij}}}{\sum_j \sigma(A_{ij})e^{A_{ij}} + \varepsilon}, \tag{5}$$

$$\sigma(A_{ij}) = \begin{cases} 1 & A_{ij} \geq q \\ 0 & others \end{cases}, \tag{6}$$

where $q$ represents the k-th (we set K = 4) largest value in $A_i$, and $\varepsilon$ is a constant used for stable numerical computation (we set it as $10^{-10}$). Through the ablation experiments conducted in Section 5, we have demonstrated that Sparse-Softmax can prevent vertices from being bound to joints with low confidence, resulting in higher precision skinning weights and better transfer effects. Additionally, we found that during loss computation, the gradient of the vertex skinning weights processed by Sparse-Softmax at $A_{ij} < q$ is zero. This allows the network to focus only on the connection between vertices and the top k most relevant joints, enabling faster network training.

#### 3.1.2. Joint Regression

The structure of the joint regression module, as shown in the lower part of Figure 3, takes the input mesh model $M$ and skinning weights $W$, and outputs the coordinates of the joint:

$$C = f_c(M, W; w_c), \tag{7}$$

where $w_c$ is the learnable parameter in the network of this module.

Specifically, we first encode the vertex coordinates into $h$-dimensional features $V' \in \mathbb{R}^{n \times h}$ using three GKEdgeConv convolutional layers. Then, we employ skinning-based pooling (detailed method in Section 5.1) with the skinning weights $W$ to map the vertex features to the joint's latent space. These joint features are then fed into our proposed Skeleton EdgeConv layers (detailed method in Section 5.2), ultimately generating the coordinates $C \in \mathbb{R}^{m \times 3}$ of the joints. It is worth noting that we assume a fixed skeletal topology, where the joints $C$ are ordered. This ensures that each joint contains unique semantic information, providing the necessary conditions for isomorphic skeletal latent space encoding of the source model and the reference model.

### *3.2. Neural Pose Transfer*

After obtaining the source mesh $M_s$ and the reference mesh $M_r$, with their respective skinning weights $W^s$ and $W^r$, as well as joint information $C^s$ and $C^r$, in this stage, we will

utilize our proposed pose transfer branch to generate the final target mesh $M_t$. The overall structure of this module is illustrated in Figure 4. The source model $M_s$ and reference model $M_r$ are passed through a shared-weight feature extractor to extract grid features $F_s$ and $F_r$. These features are then combined with weights $W^s$, $W^r$, $C^s$, and $C^r$, followed by skinning pooling to obtain hidden spatial features $O$ for the joint. Afterward, the skeletal information from both models is combined and processed through skeletal edge convolutions and skinning unpooling to obtain per-vertex hidden transformations $T'$. Finally, the final result is obtained by applying deformations through an MLP used as the decoder.

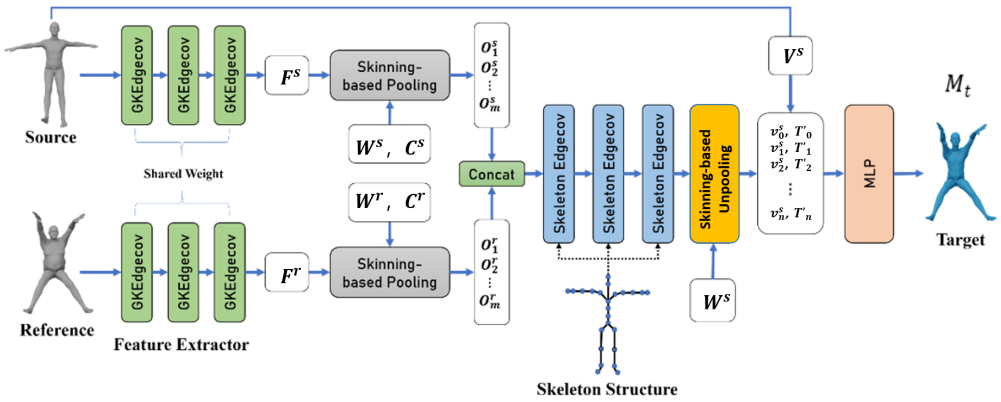


**Figure 4.** The network architecture of the pose transfer branch. The encoder in Figure 2 comprises the GKEdgeConv module with shared weights, input source model, and reference model. Then, through the GKEdgeConv layer, skinning-based pool, Skeleton EdgeConv, and skinning-based unpooling and after linear mixing operations with the source model vertices to encode the target model, the skeleton structure comes from the bone information obtained from the rigging part, which is $C$ in Figure 2.

#### 3.2.1. Skinning-Based Pooling

After encoding the vertex coordinates of the source model and the reference model through three layers of GKEdgeConv layers with shared weights, we obtain the $h_v$-dimensional latent space features $F^s, F^r \in \mathbb{R}^{n \times h_v}$:

$$F^s = f_{GKE}(M_s; w_z), \tag{8}$$

$$F^r = f_{GKE}(M_r; w_z). \tag{9}$$

As Wang et al. [28] describes, due to the unordered nature of 3D model vertices, convolutional neural networks (CNNs) for 3D models typically focus more on local geometric features of vertices rather than global features such as global pose or shape. Additionally, the vertex features output by CNNs lack a mapping relationship between 3D models and often contain redundancy. In such cases, neural networks that solely encode vertex features struggle to capture the correlation between the latent spaces of the source and reference models, posing challenges for subsequent pose learning. Further compressing and encoding of these unordered vertex features is necessary to reduce redundancy.

As shown in Figure 5, skinning weights represent the proxy relationship between vertices and joints and implicitly carry information about the body parts to which the vertices belong. In our method, the assumed skeletal topology is fixed, and thus, the joints mapped by the skinning weights are also ordered. We can map redundant and unordered vertex features to a smaller quantity of topologically isomorphic skeletal latent space by exploiting this characteristic of skinning weights. Specifically, as described in [34], skinning-based pooling is defined as follows:

$$O_j = \frac{\sum_i^n W_{ji} F_i}{\sum_i^n W_{ji}}, \tag{10}$$

where $F_i \in \mathbb{R}^{h_v}$ represents the $i$-th vertex latent space feature, and $O_j \in \mathbb{R}^{h_v}$ represents the $j$-th joint's latent space feature. As shown in Figure 5, we use different colors to represent the skinning of model vertices bound to different joints. The left lower leg of the source model and the reference model (indicated by the purple regions in $F^s$ and $F^r$) belong to the same position in the skeletal topology. Skinning-based pooling maps this region's unordered vertex latent space features to the same limb part, thereby constructing the ordered skeletal latent spaces $O^s$ and $O^r$.

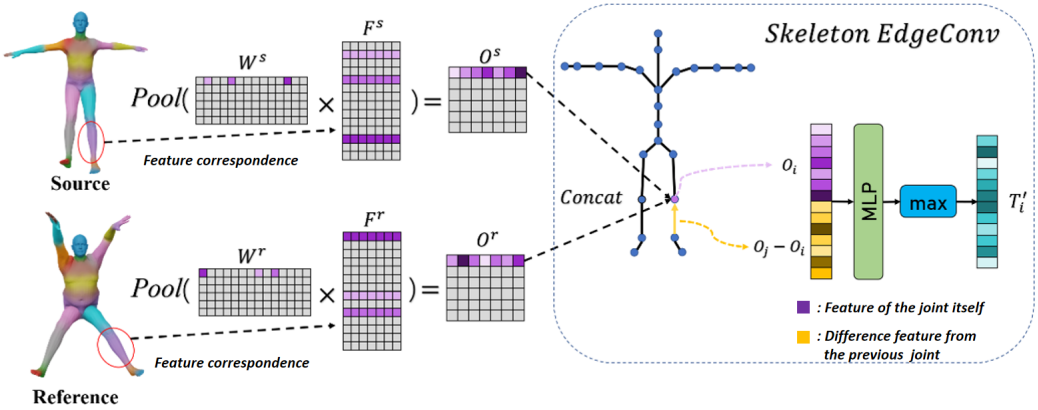


**Figure 5.** The schematic diagram of skinning-based pooling and skeleton edge convolution. Concat skeleton latent space features and joint information channels are then input into the Skeleton EdgeConv operator, which ultimately outputs the joint latent transform.

In the skinning binding branch, skinning-based pooling is primarily applied to the regression of joint positions, enhancing the network's perception of the relationship between the coordinates of joints and model vertices. In the neural pose transfer branch, we combine the latent spaces $O^s \in \mathbb{R}^{h_v \times m}$ and $O^r \in \mathbb{R}^{h_v \times m}$ of the source model and the reference model, which have a one-to-one mapping relationship (as shown in Figure 4). This allows the neural network to discover the hidden transformations embedded between the joint latent spaces of the source model and the reference model more easily.

### 3.2.2. Skeleton Edge Convolution

Generally, after skinning-based pooling is applied to vertex features, the resulting skeletal latent space features are independent. In skin deformation, the skeleton is commonly represented as a tree, and the skeletal topology is an essential basis for forward kinematics. The transformations of child joints are closely related to their parent joints. We aim for the neural network to learn these forward kinematics characteristics, enabling it to better extract information from the skeletal latent space as hidden transformations.

Due to the fixed topology of the skeleton in our network and the relatively small number of skeleton points, we were inspired by the GKEdgeConv block [11]. We utilize this fixed-topology construction with a locally supported edge convolution operator (proposed in [28]) to learn the relationships between a certain joint and its 1-neighborhood joints. This allows us to obtain deeper-level skeletal latent transformation information. The structure of the skeletal edge convolution is depicted in the right half of Figure 5; unlike the GKEdgesConv block, the Skeleton EdgeConv block does not extract features from k-nearest neighbors. This is because, unlike mesh vertices, the number of skeleton points is relatively small, and selecting k-nearest neighbor points would result in a significant proportion of the overall points being included.

This could lead to a decrease in the effectiveness of local feature extraction. For a given latent space feature $x_i$, the edge convolution feature $x_i'$ can be obtained by encoding its global feature $x_i$ and its local feature $x_j - x_i$ using a multi-layer perceptron:

$$x_i' = \max_{j \in N(i)} MLP(x_i, x_j - x_i, w_{mlp}), \tag{11}$$

where $w_{mlp}$ is the learnable weights of the multi-layer perceptron. $N(i)$ represents the 1-neighborhood of feature $x_i$, determined by the skeleton topology.

As shown in Figures 3 and 4, we apply skeletal edge convolution operators in both the pose transfer branch and the joint regression module. In the joint regression, the input of the skeletal edge convolution layer is the skeletal latent space of the two models, obtained by pooling the vertex depth features, and it outputs the coordinates $C_i$ of the joint. As shown in Figure 5, in the pose transfer branch, we concatenate the skeletal latent spaces $O_i^s, O_i^r$ of the source model and the reference model, obtained by pooling the skinned vertices, along with their joint $C_i^s, C_i^r$ coordinate channels:

$$x_i = Concat(O_i^s, O_i^r, C_i^s, C_i^r). \tag{12}$$

Then, it is fed into the skeletal edge convolution operator, and the final output is the latent transformation $T_i \in \mathbb{R}^{h_t}, (i = 1, 2, \cdots, m)$ of the joints in the dimensional space.

3.2.3. Skinning-Based Unpooling

As shown in Figure 5, inspired by linear blend skinning (LBS), for the latent transformation $T \in \mathbb{R}^{m \times h_t}$ obtained from the skeletal edge convolution, it is unpooled and transformed on a per-point basis using skinning weights, in a linear blending manner, to obtain the per-point latent transformation $T' \in \mathbb{R}^{n \times h_t}$:

$$T_i' = \sum_{i}^{m} W_{ji} T_j, \tag{13}$$

where $T_i' \in \mathbb{R}^{h_t}$ represents the $h_t$-dimensional latent transformation associated with the $i$th vertex. By concatenating this M-dimensional latent transformation with the coordinate channels of the source model vertices and inputting them into a multi-layer perceptron decoder, we can obtain the final coordinates $V^t$ of the target mesh $M^t$, denoted as:

$$V^t = f_d(concat(T', V^s); w_d), \tag{14}$$

where $w_d$ represents the learnable parameters of this module $f_d$. We use a combination of linear blending and multi-layer perceptron to encourage the network to decode and generate deformation effects similar to linear blend skinning while preserving fine details.

## 4. Training

Our loss function is as follows:

$$L = \alpha L_s + \beta L_c + \gamma L_v + \delta L_e, \tag{15}$$

where $L_s$ represents the skin weight loss, $L_c$ represents the joint regression loss, $L_v$ represents the reconstruction loss, $L_e$ represents the margin loss, and $\alpha, \beta, \gamma, \delta$ represent the hyperparameters for each of the losses.

**Skinning weight loss**: This is calculated by comparing the predicted skin weights $W \in \mathbb{N}^{n \times m}$ with the ground truth weights $\widehat{W} \in \mathbb{N}^{n \times m}$. We treat the per-vertex skin weights

as probability distributions and measure the difference between the predicted distribution and the ground truth distribution using KL divergence, denoted as:

$$L_s = -\frac{1}{nm}\sum_{i}^{n}\sum_{j}^{m}\widehat{W}_{ij}log\left(\frac{W_{ij}}{\widehat{W}_{ij}}\right). \tag{16}$$

Considering that we use Sparse-Softmax as the probability transformation function to calculate $W_{ij}$, to prevent gradient explosion, we only consider the top k highest predicted values $(topk(W_i))$ for each vertex in the computation, denoted as:

$$L_s = -\frac{1}{nm}\sum_{i}^{n}\sum_{j\in topk(W_i)}^{k}\widehat{W}_{ij}log\left(\frac{W_{ij}}{\widehat{W}_{ij}+\varepsilon}\right), \tag{17}$$

where $\varepsilon$ is $10^{-10}$, used for stable numerical computation. We observed that, as shown in ablation experiments , compared to using cross-entropy as the loss function, using KL divergence as the loss function leads to higher accuracy in generating skin weights.

**Joint regression loss**: The difference between the predicted values $C$ and the ground truth values $\widehat{C}$ is measured using mean squared error, denoted as:

$$L_c = \frac{1}{\mid C \mid}\sum_{i}\mid\mid C_i - \widehat{C}_i \mid\mid_2^2. \tag{18}$$

**Reconstruction loss**: The reconstruction loss is applied to measure the difference between the reconstructed vertex positions $V$ and the ground truth vertex positions $\widehat{V}$, aiming to guide the vertices to regress to their correct positions:

$$L_V = \frac{1}{\mid V \mid}\sum_{i}\mid\mid V_i - \widehat{V}_i \mid\mid_2^2. \tag{19}$$

**Edge length loss**: When solely using the L2 norm between vertices for regression, there may be issues with insufficient preservation of mesh details. Inspired by [17], we introduce an edge length regularization term as a penalty, encouraging the network to generate smoother target models:

$$L_e = \frac{1}{\mid E \mid}\sum_{e}^{|E|}\mid\mid e - \widehat{e} \mid\mid_2^2, \tag{20}$$

where $e$ represents the predicted edge lengths of the mesh model, while *widehate* represents the corresponding edge lengths of the ground truth mesh model.

To accelerate convergence and achieve better training results, we adopt a staged training strategy for the network model's parameters $w_s, w_c, w_z, w_d$. In the first stage, we set $\alpha = 1, \beta = 1, \gamma = 0, \delta = 0$ and primarily update the parameters $w_s, w_c$ in the skin binding branch. In the second stage, we set $\alpha = 0, \beta = 0, \gamma = 1, \delta = 5 \times 10^{-4}$, fix the parameters $w_s, w_c$, and update the parameters $w_z, w_d$ in the pose transfer branch. In the final stage, we set $\alpha = 0.1, \beta = 0.1, \gamma = 1, \delta = 5 \times 10^{-4}$ and update all parameters to obtain the final results. The parameters are set in this way because we found that riggingnet has been fully trained in the previous training, and the focus of the overall training should be on correcting the subsequent posetransfernet. Therefore, the sizes of $\alpha$ and $\beta$ are reduced to reduce the training proportion of riggingnet parameters.

## 5. Experiments

In this section, we first introduce the experimental setup, dataset, and other experimental details. Then, we evaluate the results of skin weight binding and skeleton extraction, pose transfer of the SMPL model, and pose transfer of non-SMPL models separately. Finally, we conduct ablation experiments to demonstrate the effectiveness of each module in our proposed method.

### *5.1. Experimental Details*

**Experimental environment**: The experiments in this study were conducted in a Python environment. The experimental platform is a processor from Intel, the Intel Core i7-11700K 3.60 GHz, with 16GB of RAM, sourced from Intel in Santa Clara, CA, USA, The GPU used was an Nvidia RTX 3090 from Nvidia in Santa Clara, CA, USA. with a VRAM capacity of 24 GB. The operating system used was the 64-bit version of Windows 10. PyTorch, NumPy, and other library functions were utilized for implementation.

**Dataset**: To train and test the models in this study, we utilized a popular dataset constructed using the SMPL model [35]. Specifically, during the training phase, random pose and shape parameters were generated in each epoch, and the resulting 3D coordinates of 5000 SMPL models, their corresponding model skeletons, and skinning weights were used as ground truth values. The vertex positions in the sequence were randomly permuted (correspondingly, the vertex skinning weights underwent the same permutation, and the model's face indices were reconstructed). For the test set, we employed the same method to generate 50,000 pairs of reference models and source models randomly. Then, we generated the corresponding target model using the pose parameters from the source model and the shape parameters from the reference model. In addition to SMPL, we evaluated the performance of our model on other datasets such as SMAL [36], FAUST [37], DYNA [38], and the MG dataset [39].

**Training**: During the training process, the vertex order of the source model and reference model in the sequence will be randomly permuted (correspondingly, the target model undergoes the same vertex permutation). Additionally, random vertex noise is added to the source model and reference model. For the first stage of training, the batch size is set to 16, the base learning rate is set to $10^{-4}$, and the number of training epochs is set to 100. In the second stage, the batch size is set to 8, the base learning rate is set to $5 \times 10^{-5}$, and the number of training epochs is set to 200. In the third stage, the batch size is set to 4, the base learning rate is set to $5 \times 10^{-5}$, and the number of training epochs is set to 50. For all stages, the Adam optimization algorithm is used to update the network parameters, and the learning rate is updated using the cosine annealing learning rate schedule.

### *5.2. Experimental Result*

This subsection presents the results of various experiments. In the rigging experiment, we compare our method with commonly used rigging approaches such as RigNet [40] regarding skeleton prediction and vertex binding. In the pose transfer experiment, we compare our method with Yang et al. [11], NPT [8], 3D-CoreNet [7], and MAPConNet [31] with supervised training methods. In the pose transfer experiment, in addition to comparing with the standard SMPL model, we also demonstrate the performance of our method on non-SMPL models and test its robustness against noise.

We first tested our rigging module, as shown in Figure 6. We compared our method with commonly used rigging methods such as RigNet, NBS, and Yang et al. [11] It can be observed that RigNet struggles to achieve the same bone topology as SMPL due to varying predicted bone counts, leading to significant differences in skinning weight coloring compared to GT. NBS exhibits substantial differences from GT in both skeleton prediction

and skinning weights. On the other hand, Yang et al. [11] achieve more accurate weights and skeletons than GT but still show some differences compared to our method, such as the joint position at the right wrist. Our process, however, consistently produces accurate skeletons and skinning weights across various poses.

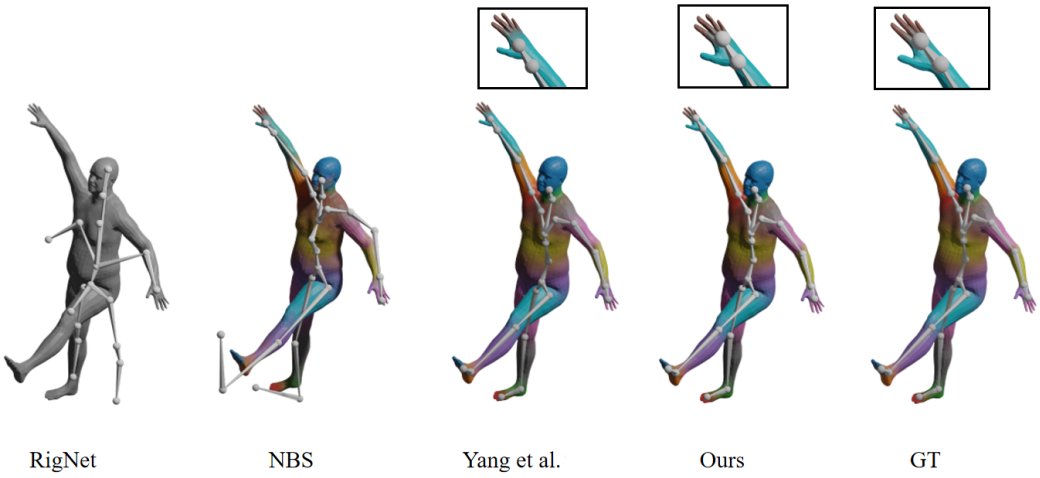


**Figure 6.** Comparison testing of rigging [11].

In the SMPL mesh model, Figure 7 shows the results of pose transfer using NPT, Yang et al. [11], 3D-CoreNet, MAPConNet, and our method. It can be observed that NPT results in significant distortions. 3D-CoreNet performs better than NPT, but the body features still need to be more accurate, such as the less prominent abdominal muscles compared to the in the results of the first row. MAPConNet shows better feature preservation than 3D-CoreNet, but there are still some issues handling details, such as the less smooth right thigh in the first and second rows. In Yang et al. [11], the angle of the left foot is not accurate. Yang et al. [11] directly calculate the skeletal transformation matrix from the source model to the reference model without accurately calculating the skeletal twist. In contrast, our method directly predicts the hidden skeletal pose transformation using a neural network, which implicitly contains the twist information, resulting in more realistic effects, as shown in Figure 8.

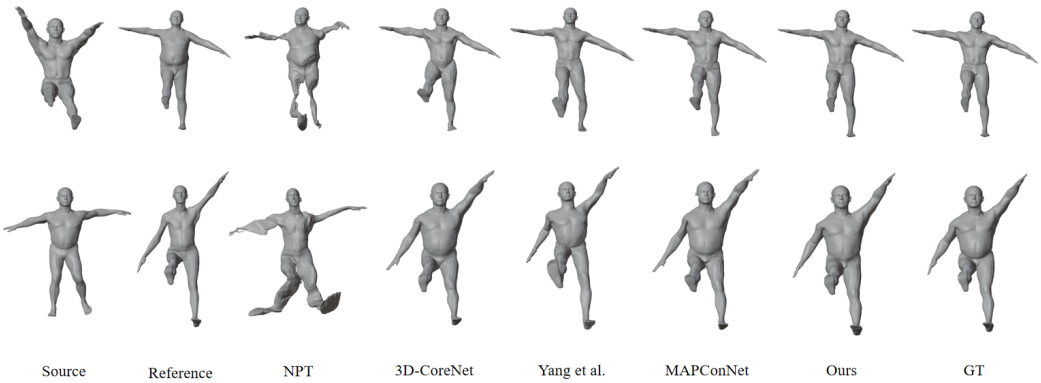


**Figure 7.** Comparison testing on the SMPL model. On the SMPL model, our method is compared with the results of NPT, 3D-CorNet, and MAPConNet, focusing on whether the pose transfer is in place, whether the detailed features are preserved, and whether the model surface is smooth or not [11].

The quantitative metrics are shown in Table 1. PMD (Equation (19)) and EL (Equation (20)) were defined earlier, and we also use CD (Chamfer distance) and MaxDist to measure the quality of the generated models. A minor CD indicates closer proximity between the predicted point set and the ground truth point set. At the same time, MaxDist

represents the maximum distance between corresponding vertices, reflecting whether there are unnatural protrusions or indentations on the mesh surface. Table 1 shows that both the PMD and CD metrics outperform the compared methods, indicating that our obtained mesh poses are more accurate. The EL and MaxDist metrics also demonstrate that our method achieves better results in local details, and the mesh is smoother without unnatural concavities or convexities. Figure 7 illustrates the real example application results presented in Table 1.

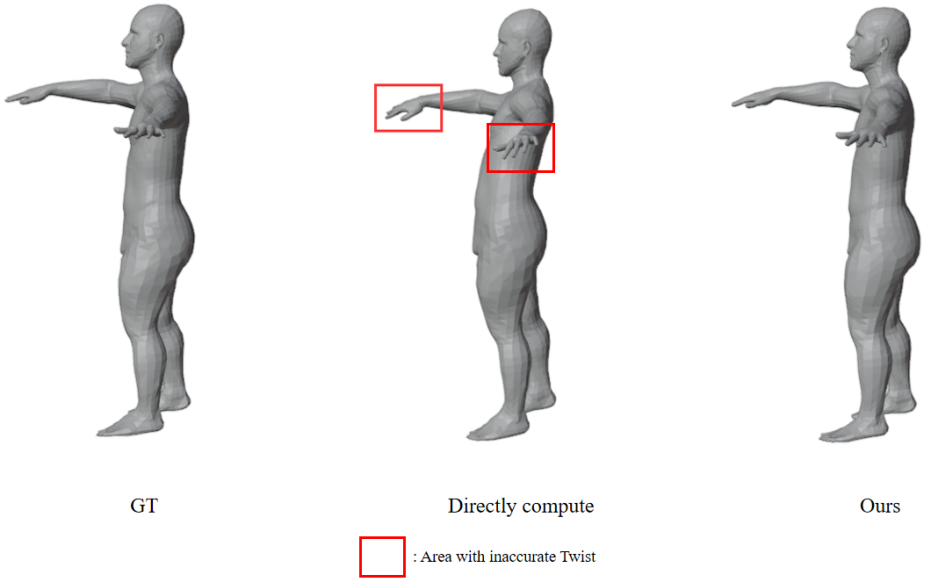


**Figure 8.** Comparison testing of twist computing

In non-SMPL mesh models, we conducted tests on FAUST, DYNA, the MG dataset, and SMAL. In FAUST and DYNA, the human body meshes were obtained by aligning 3D scanning sequences with templates. The vertex count of these two datasets is the same as the SMPL model, with 6890 vertices. The pose transfer results on FAUST and DYNA are shown in Figure 9. Both the reference and source models in these datasets represent real human poses, which differ significantly from SMPL and pose a significant challenge for our network. Although some models may have minor feature loss, our method performs well in both motion transfer and feature preservation tasks.

**Table 1.** Quantitative comparison of pose transfer results on the SMPL dataset. Our method performs best on the SMPL dataset in terms of the quantitative indicators PMD (Equation (19)), EL (Equation (20)), CD (Chamfer distance), and MaxDist, ↓ indicates that smaller values are better.

| Method | PMD ($10^{-4}$) ↓ | EL ($10^{-4}$) ↓ | CD ($10^{-4}$) ↓ | MaxDist ($10^{-1}$) ↓ |
|---|---|---|---|---|
| NPT | 8.01 | 16.63 | 2.63 | 8.89 |
| Yang et al. [11] | 0.62 | 1.03 | 0.32 | 0.52 |
| 3D-CoreNet | 0.58 | 0.92 | 0.53 | 1.02 |
| MAPConNet | 0.53 | 0.88 | 0.15 | 0.36 |
| Ours | 0.44 | 0.83 | 0.14 | 0.32 |

The standard model in the MG dataset, based on SMPL, has 27,554 vertices, with the same topology as the SMPL model for the first 6890 vertices. The presence of models with more vertices and complex clothing significantly increases the difficulty of motion transfer and feature preservation. We used the first 6890 vertices for the test, as shown in Figure 10; we tested MAPConNet and our method on the MG dataset. In the first and second rows, when using the clothed MG dataset model as the reference, MAPConNet fails to identify the pose, resulting in unnatural distortions. In contrast, our method demonstrates more accurate

pose recognition. In the second and fourth rows, it can be observed that our method is less sensitive to the features of the source model in the MG dataset. There are distortions in the right foot and finger positions. This is because our method uses the skinning weights from the SMPL model as the ground truth for training, and the vertex weights for clothing differ significantly from those of SMPL, leading to some feature loss.

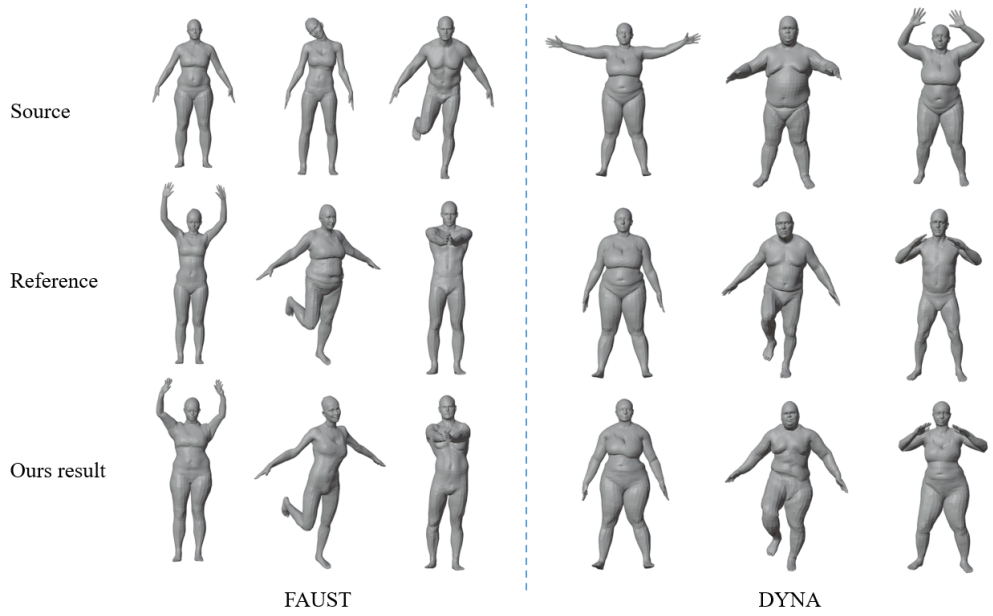


**Figure 9.** The performance on the FAUST and DYNA datasets. On the non-SMPL model, due to the large difference between the model skin weights and SMPL, our method lacks detail preservation but still performs the pose transfer task well.

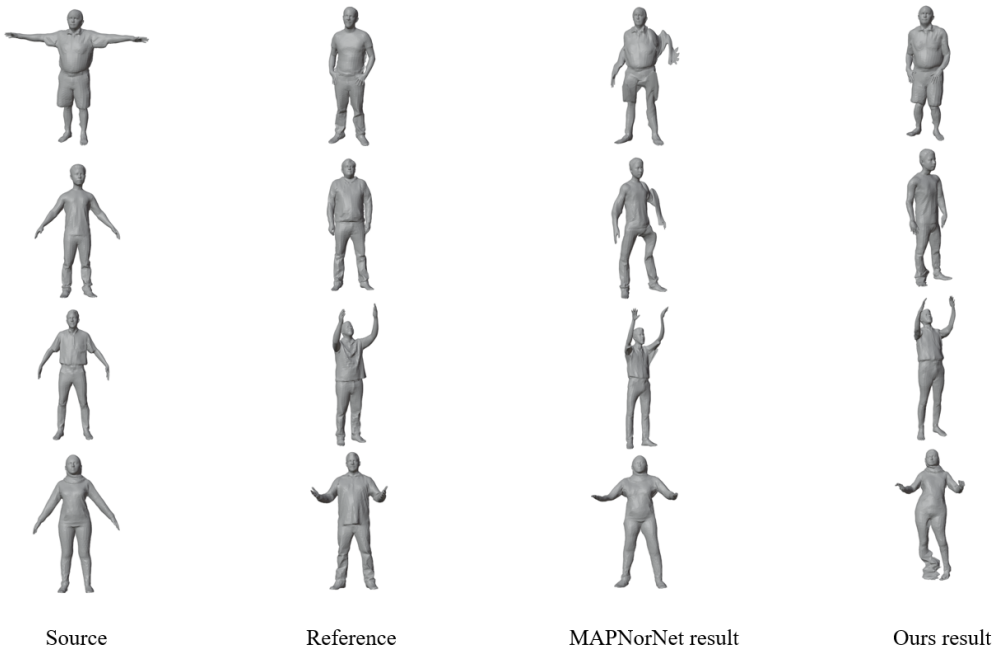


**Figure 10.** The test results on the MG dataset. We compared our method with MAPConNet on MG dataset data and found that our method is slightly inferior in body detail extraction but very accurate in pose extraction.

Figure 11 demonstrates the experimental results of our method on SMAL. Feature extraction is more difficult due to the denser number of vertices on the face. As a result, in the second column, there might be some non-smoothness in the head vertices when

the horse opens its mouth. However, overall, our model performs remarkably well in accomplishing pose transfer tasks on the SMAL model.

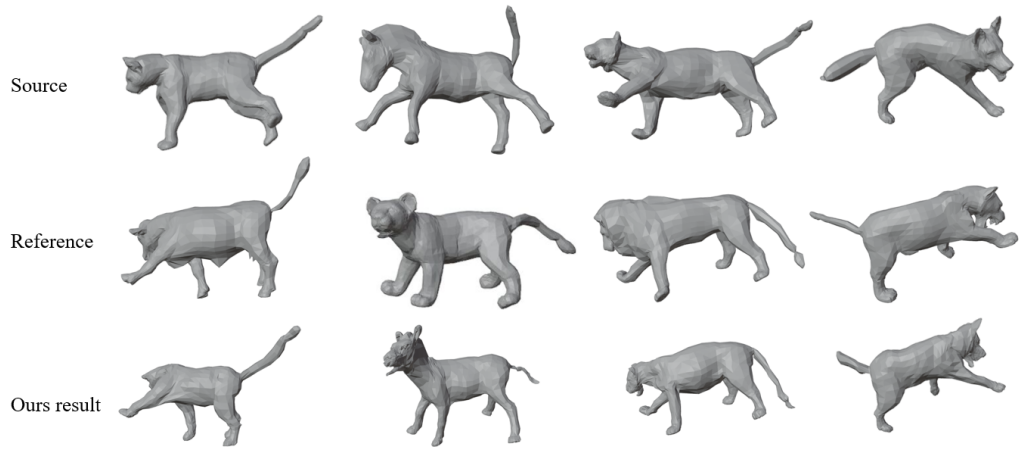


**Figure 11.** The results of pose transfer on the SMAL dataset. In the SMAL model, our method accomplishes the task well, except for some parts of the face that have a high density of vertices and are drastically deformed and slightly bumpy.

To test the robustness of our model to noise, we conducted noise testing. We added random noise to the vertices of the reference models and performed pose transfer. The results are shown in Figure 12. The second column represents the reference model without noise, while the fourth and fifth columns show the reference models with added noise and the corresponding pose transfer results. The results show that the generated models have motions that are very similar to the reference models without noise. This indicates that our method maintains a certain level of robustness even in noise interference. Despite the added noise, our approach still produces accurate and visually coherent pose transfer results, demonstrating its ability to handle noisy input and maintain the overall quality of the transferred motions.

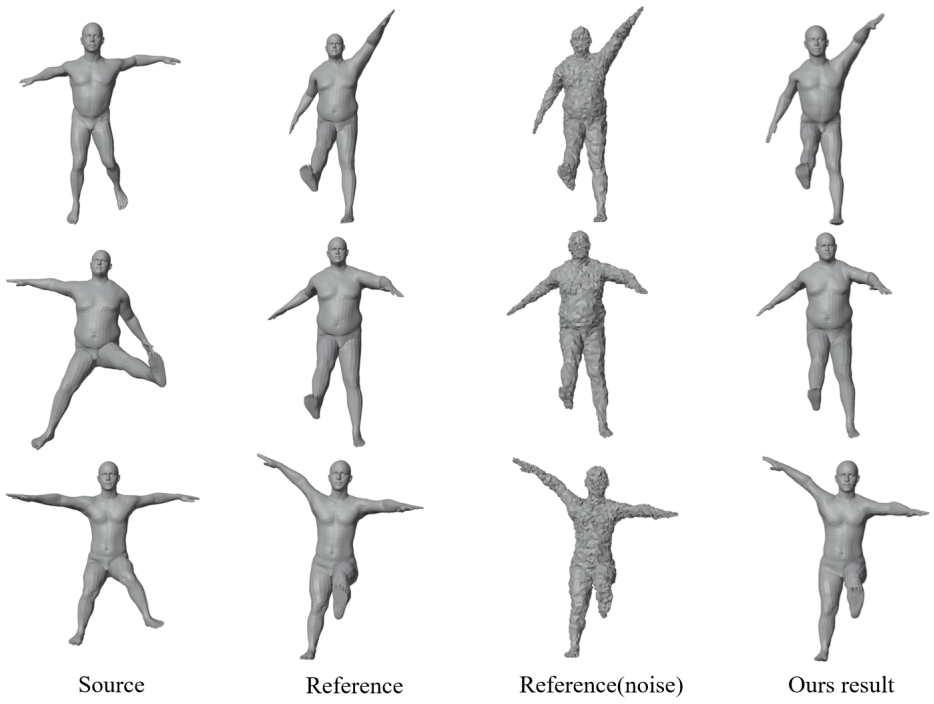


**Figure 12.** The results of the noise testing. We input the random-noise-processed reference model to test the robustness of our method to noise. It can be seen that our method copes excellently with noise.

We also conducted ablation experiments to determine the effectiveness of our Sparse-Softmax and the application of skinning weights in the transfer network, as shown in Table 2. $N(S)$ indicates replacing Sparse-Softmax with a regular Softmax function, and it can be observed that both the final PMD and EL metrics increase. Similarly, $N(K)$ represents replacing the KL divergence loss function with cross-entropy loss during weight loss computation, and $N(W)$ means replacing the added skinning weight matrix in the transfer network with a matrix of equal shape and all values set to 1. In these cases, the pose transfer performance of the entire network significantly deteriorates. PMD and EL metrics both experience a notable increase. This also demonstrates the crucial role played by skinning weights in the transfer stage. $N(D)$ represents replacing the module that calculates the hidden transformation between skeletal poses with Yang et al.'s [11] method of directly computing the transformation matrix between two skeletons. It was found that although the neural network can partially compensate for inaccuracies in skeletal twist calculation, it leads to a decrease in accuracy.

**Table 2.** Ablation experiments. The quantitative indicators PMD and EL can be used to analyze and prove that each module used in our method positively affects excellent results, ↓ indicates that smaller values are better.

| Module | PMD ($10^{-4}$) ↓ | EL ($10^{-4}$) ↓ |
|---|---|---|
| $N(W)$ | 9.99 | 3.86 |
| $N(K)$ | 0.61 | 0.92 |
| $N(S)$ | 0.51 | 0.89 |
| $N(D)$ | 0.46 | 0.84 |
| FULL | 0.44 | 0.83 |

### *5.3. Limitations and Future Perspectives*

Although Sparse-Softmax, which we used, allows for more accurate vertex binding to skeletons and achieves better transfer results, its implementation involves setting the values of predicted skinning weights to zero or a minimal value for all but the top k weights. This approach can lead to issues such as the random initial position of the maximum value and the vanishing gradient problem for smaller values, resulting in slow initial descent and potential convergence to a local optimum during training. To address this problem, we attempted a solution by using the indices of the top k weights from ground truth instead of predicted weights during training. This ensures that the network starts learning in the correct positions from the beginning. As shown in Table 3, using GTidx (the GT index) leads to a faster descent of the skinning weight loss (SWL) and reduces its final value. However, it does not contribute to the final transfer performance, as indicated by the PMD metric, which increases instead of decreasing. This limitation can be gradually overcome as GT provides better and more accurate skinning weights in the future.

**Table 3.** Attempts to improve the skinning weight loss. We found that using GTidx (the GT index), although it causes the weighted tare loss (SWL) to decrease faster and reduce its final value, hurts the final migration results, ↓ indicates that smaller values are better.

| Test | SWL ($10^{-3}$) ↓ | PMD ($10^{-4}$) ↓ |
|---|---|---|
| GTidx | 1.22 | 0.56 |
| Original | 2.09 | 0.44 |

In addition, we utilize the edge convolution module of DGCNN, where the number of mesh vertices influences the computation of k-nearest neighbors. While incorporating neighborhood information in the network yields good results, it also increases the network's

sensitivity to the vertices. Suppose we directly train the network using SMPL to handle pose transfer. In that case, the effectiveness of pose transfer will gradually deteriorate with the change in the number of vertices, as shown on the left side of Figure 13. It can be observed that the pose transfer results are already poor when the number of vertices is halved. This leads to unsatisfactory pose transfer results when using the SMAL model with fewer vertices, and the same applies when using manually modeled models, as shown on the right side of Figure 13. Excellent results can be obtained when the model has a vertex count close to 6890, as seen in the first column, but the performance declines when the vertex count reaches 8972 in the second column. To address this issue, we simplify the vertex count of the SMPL model to construct the dataset and retrain the network. We also record the mapping relationship between the simplified and original mesh, allowing us to map the network's output back to the original mesh for further processing.

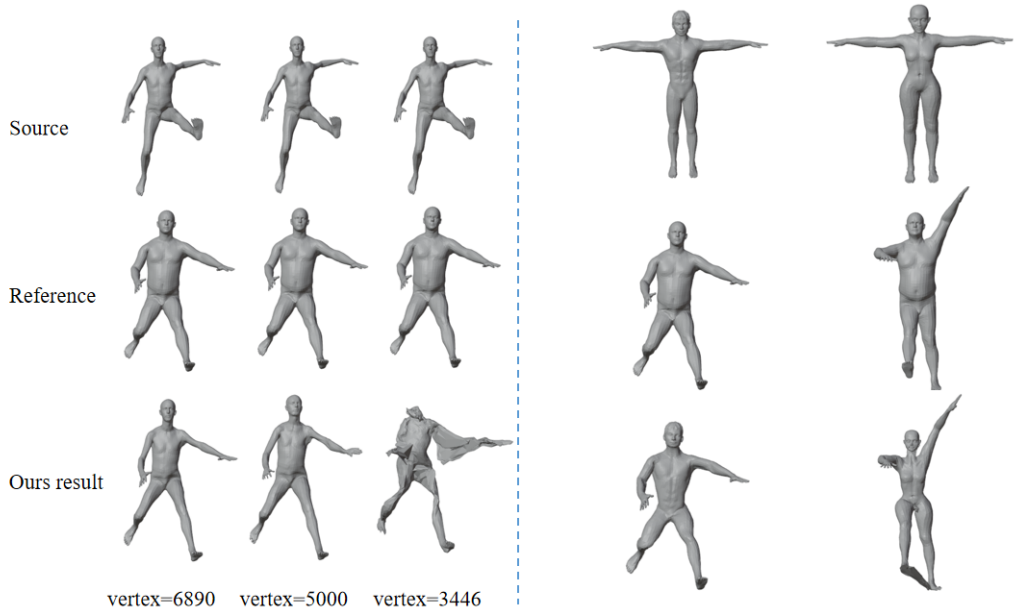


**Figure 13.** Using reference and source models with different vertex counts (**left** side) and results using manually modeled models (**right** side).

Additionally, the random generation of parameters to control the generation of GT models in our dataset results in unnatural representations of specific actions in the GT models. This, in turn, leads to the suboptimal performance of our model in transferring specific actions sampled from real scenarios. For example, in the first row of Figure 14, the source model exhibits a single-leg jumping pose, where our model fails to accurately recognize the posture of the right leg, resulting in distortion across the entire right side of the body. Similarly, in the second row, the source model displays a thumbs-up gesture, which undergoes significant deformation after the transfer process. At the same time, using the method of Ding et al. [41] can reduce the use of SMPL-generated datasets and the negative impact of fixed weights, which is also a method worth discussing in future development.

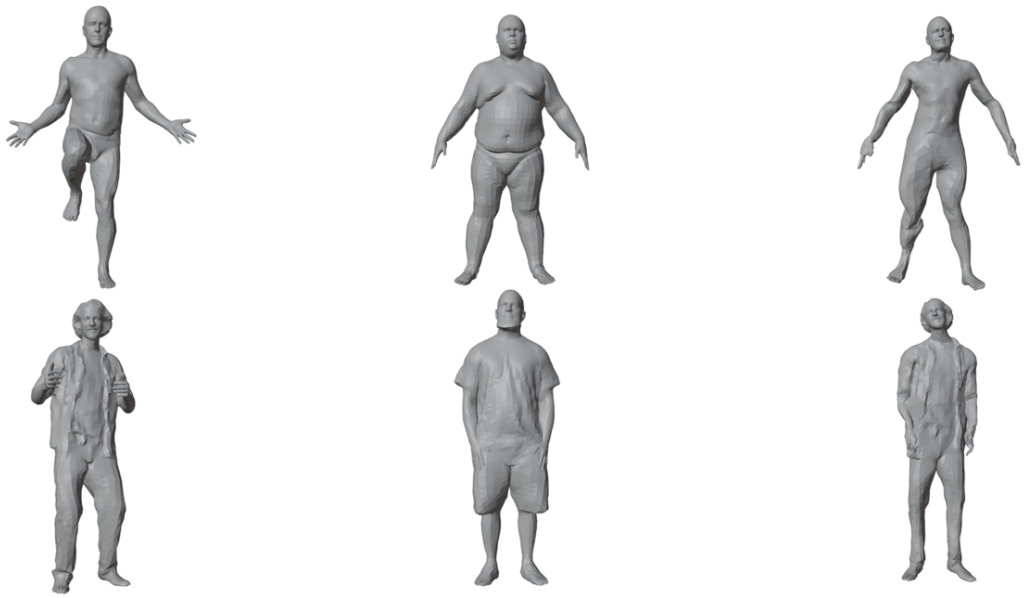


**Figure 14.** Some inaccurate results in the transfer. Our model does not extract features well for some specific poses, such as front high leg raise and thumbs up.

## 6. Conclusions

We propose an end-to-end pose transfer method based on neural network and skin deformation. We employ Sparse-Softmax to smooth the skinning weights and supervise them with KL divergence. We obtain skeletal latent features by mapping vertex features to the skeletal latent space using the skinning weights. We further derive vertex latent transformation features and guide the neural network in pose transfer utilizing the concept of skin deformation. This transforms the pose transfer problem between models into a hidden transformation problem between corresponding nodes in the source and reference models with isomorphic relationships. As a result, we generate the target model with preserved details using a skinning approach entirely through the neural network, achieving high-quality deformation effects. Compared to existing neural network-based pose transfer methods, our experimental results demonstrate superior details and pose fidelity performance.

However, as the current datasets are still relatively limited, we use datasets generated by SMPL and SMAL. There is still significant room for improvement in terms of reality. Additionally, our method still needs to improve gradient issues and sensitivity to the number of vertices. We are incorporating improvements from unsupervised methods like Chen et al. and Sun et al. [30,31], which would be a promising direction to enhance the quality of our approach further.

**Author Contributions:** Conceptualization, S.L.; methodology, S.L. and M.L.; software, S.L. and M.L.; validation, S.L. and M.Y.; resources, M.Y., F.Z. and B.H.; writing—original draft, S.L.; writing—review and editing, M.Y.; visualization, S.L.; supervision, M.Y., M.L., F.Z. and B.H.; project administration, S.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was partially supported by the Natural Science Foundation of China (No. 61762007), Natural Science Foundation of Guangxi Province, China (No. 2024GXNSFAA170255).

**Data Availability Statement:** The data that support the findings of this study are available in NeuralSkeletonPoseTransfer at https://github.com/lzszmp/3D-mesh-pose-transfer-based-on-deep-learning-skinning-deformation, accessed on 1 May 2024. These data were derived from the following resources available in the public domain: SMPL, https://smpl.is.tue.mpg.de/accessed on 1 May 2024.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References


1. Botsch, M.; Sumner, R.; Pauly, M.; Gross, M. Deformation transfer for detail-preserving surface editing. In Proceedings of the Vision, Modeling & Visualization, Citeseer, Aachen, Germany, 22–24 November 2006; pp. 357–364.
2. Chen, H.; Tang, H.; Shi, H.; Peng, W.; Sebe, N.; Zhao, G. Intrinsic-extrinsic preserved gans for unsupervised 3d pose transfer. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Virtual, 11–17 October 2021; pp. 8630–8639.
3. Shi, X.; Zhou, K.; Tong, Y.; Desbrun, M.; Bao, H.; Guo, B. Mesh puppetry: Cascading optimization of mesh deformation with inverse kinematics. In *ACM SIGGRAPH 2007 Papers*; Association for Computing Machinery: New York, NY, USA, 2007; pp. 81–es.
4. Sumner, R.W.; Popović, J. Deformation transfer for triangle meshes. *ACM Trans. Graph. (TOG)* **2004**, *23*, 399–405.
5. Zou, X.; Li, G.; Yin, M.; Liu, Y.; Wang, Y. Deformation-Graph-Driven and Deformation Aware Spectral Pose Transfer. *J. -Comput.-Aided Des. Comput. Graph./Jisuanji Fuzhu Sheji Yu Tuxingxue Xuebao* **2021**, *33, 1234–1245.*.
6. Chen, H.; Tang, H.; Yu, Z.; Sebe, N.; Zhao, G. Geometry-contrastive transformer for generalized 3d pose transfer. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 22 February–1 March 2022; Volume 36, pp. 258–266.
7. Song, C.; Wei, J.; Li, R.; Liu, F.; Lin, G. 3d pose transfer with correspondence learning and mesh refinement. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 3108–3120.
8. Wang, J.; Wen, C.; Fu, Y.; Lin, H.; Zou, T.; Xue, X.; Zhang, Y. Neural pose transfer by spatially adaptive instance normalization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2020; pp. 5831–5839.
9. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
10. Guo, Y.; Wang, H.; Hu, Q.; Liu, H.; Liu, L.; Bennamoun, M. Deep learning for 3D point clouds: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *43*, 4338–4364.
11. Yang, S.; Yin, M.; Li, M.; Li, G.; Chang, K.; Yang, F. 3D mesh pose transfer based on skeletal deformation. *Comput. Animat. Virtual Worlds* **2023**, *34*, e2156.
12. Magnenat, T.; Laperrière, R.; Thalmann, D. Joint-dependent local deformations for hand animation and object grasping. In Proceedings of the Graphics Interface'88, Edmonton, AB, Canada, 6–10 June 1988.
13. Baran, I.; Popović, J. Automatic rigging and animation of 3d characters. *ACM Trans. Graph. (TOG)* **2007**, *26*, 72–es
14. Aouaidjia, K.; Sheng, B.; Li, P.; Kim, J.; Feng, D.D. Efficient body motion quantification and similarity evaluation using 3-D joints skeleton coordinates. *IEEE Trans. Syst. Man, Cybern. Syst.* **2019**, *51*, 2774–2788.
15. Kamel, A.; Liu, B.; Li, P.; Sheng, B. An investigation of 3D human pose estimation for learning Tai Chi: A human factor perspective. *Int. J. Hum. Comput. Interact.* **2019**, *35*, 427–439.
16. Karambakhsh, A.; Sheng, B.; Li, P.; Li, H.; Kim, J.; Jung, Y.; Chen, C.P. SparseVoxNet: 3-D object recognition with sparsely aggregation of 3-D dense blocks. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *35*, 532–546.
17. Li, M.; Yin, M.; Li, G.; Zhao, M.; Yang, F. Point-Cloud Self-Adaptive Pose Transfer Based on Skinning Deformation. *J. -Comput.-Aided Des. Comput. Graph./Jisuanji Fuzhu Sheji Yu Tuxingxue Xuebao* **2022**, *34, 1673–1683*.
18. Deng, J.; Lu, J.; Zhang, T. Unsupervised Template-assisted Point Cloud Shape Correspondence Network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville TN, USA, 11–15 June 2024; pp. 5250–5259.
19. Li, W.; Liu, M.; Liu, H.; Wang, P.; Cai, J.; Sebe, N. Hourglass Tokenizer for Efficient Transformer-Based 3D Human Pose Estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville TN, USA, 11–15 June 2024; pp. 604–613.
20. Gao, L.; Yang, J.; Qiao, Y.L.; Lai, Y.K.; Rosin, P.L.; Xu, W.; Xia, S. Automatic unpaired shape deformation transfer. *ACM Trans. Graph. (TOG)* **2018**, *37*, 237.
21. Kovnatsky, A.; Bronstein, M.M.; Bronstein, A.M.; Glashoff, K.; Kimmel, R. Coupled quasi-harmonic bases. In *Computer Graphics Forum*; Wiley Online Library: Hoboken, NJ, USA, 2013; Volume 32, pp. 439–448.
22. Lévy, B. Laplace-beltrami eigenfunctions towards an algorithm that "understands" geometry. In Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06), Washington, DC, USA, 14–16 June 2006; IEEE: Piscataway, NJ, USA; 2006; p. 13.
23. Yin, M.; Li, G.; Lu, H.; Ouyang, Y.; Zhang, Z.; Xian, C. Spectral pose transfer. *Comput. Aided Geom. Des.* **2015**, *35*, 82–94.
24. Cosmo, L.; Norelli, A.; Halimi, O.; Kimmel, R.; Rodola, E. Limp: Learning latent shape representations with metric preservation priors. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Proceedings, Part III 16; Springer: Berlin/Heidelberg, Germany, 2020; pp. 19–35.
25. Zhou, K.; Bhatnagar, B.L.; Pons-Moll, G. Unsupervised shape and pose disentanglement for 3d meshes. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Proceedings, Part XXII 16; Springer: Berlin/Heidelberg, Germany, 2020; pp. 341–357.

26. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 652–660.
27. Song, C.; Wei, J.; Li, R.; Liu, F.; Lin, G. Unsupervised 3d pose transfer with cross consistency and dual reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 10488–10499.
28. Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.E.; Bronstein, M.M.; Solomon, J.M. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph. (TOG)* **2019**, *38*, 146.
29. Wang, A.; Chen, H.; Lin, Z.; Han, J.; Ding, G. Repvit: Revisiting mobile cnn from vit perspective. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville TN, USA, 11–15 June 2024; pp. 15909–15920.
30. Chen, J.; Li, C.; Lee, G.H. Weakly-supervised 3D Pose Transfer with Keypoints. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Vancouver, BC, Canada, 17–24 June 2023; pp. 15156–15165.
31. Sun, J.; Chen, Z.; Kim, T.K. MAPConNet: Self-supervised 3D Pose Transfer with Mesh and Point Contrastive Learning. *arXiv* **2023**, arXiv:2304.13819.
32. Liu, S.; Gai, S.; Da, F.; Waris, F. Geometry-aware 3D pose transfer using transformer autoencoder. *Comput. Vis. Media* **2024**, *10*, 1063–1078.
33. Zhao, T.; Zeng, H.; Zhang, B.; Fan, B.; Li, C. Point-voxel dual stream transformer for 3d point cloud learning. *Vis. Comput.* **2023**, *40*, 5323–5339.
34. Li, P.; Aberman, K.; Hanocka, R.; Liu, L.; Sorkine-Hornung, O.; Chen, B. Learning skeletal articulations with neural blend shapes. *ACM Trans. Graph. (TOG)* **2021**, *40*, 130.
35. Loper, M.; Mahmood, N.; Romero, J.; Pons-Moll, G.; Black, M.J. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graph.* **2015**, *34*, 851–866. https://doi.org/10.1145/2816795.2818013.
36. Zuffi, S.; Kanazawa, A.; Jacobs, D.W.; Black, M.J. 3D menagerie: Modeling the 3D shape and pose of animals. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6365–6373.
37. Bogo, F.; Romero, J.; Pons-Moll, G.; Black, M.J. Dynamic FAUST: Registering human bodies in motion. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6233–6242.
38. Pons-Moll, G.; Romero, J.; Mahmood, N.; Black, M.J. Dyna: A model of dynamic human shape in motion. *ACM Trans. Graph. (TOG)* **2015**, *34*, 120.
39. Bhatnagar, B.L.; Tiwari, G.; Theobalt, C.; Pons-Moll, G. Multi-garment net: Learning to dress 3d people from images. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Long Beach, CA, USA, 15–20 June 2019; pp. 5420–5430.
40. Xu, Z.; Zhou, Y.; Kalogerakis, E.; Landreth, C.; Singh, K. Rignet: Neural rigging for articulated characters. *arXiv* **2020**, arXiv:2005.00559.
41. Ding, P.; Cui, Q.; Wang, H.; Zhang, M.; Liu, M.; Wang, D. Expressive forecasting of 3d whole-body human motions. In Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 20–27 February 2024; Volume 38, pp. 1537–1545.