Pose-aware Attention Network for Flexible Motion Retargeting by Body Part

Lei Hu†, Zihao Zhang†, Chongyang Zhong, Boyuan Jiang, Shihong Xia*,

Abstract—Motion retargeting is a fundamental problem in computer graphics and computer vision. Existing approaches usually have many strict requirements, such as the source-target skeletons needing to have the same number of joints or share the same topology. To tackle this problem, we note that skeletons with different structure may have some common body parts despite the differences in joint numbers. Following this observation, we propose a novel, flexible motion retargeting framework. The key idea of our method is to regard the body part as the basic retargeting unit rather than directly retargeting the whole body motion. To enhance the spatial modeling capability of the motion encoder, we introduce a pose-aware attention network (PAN) in the motion encoding phase. The PAN is pose-aware since it can dynamically predict the joint weights within each body part based on the input pose, and then construct a shared latent space for each body part by feature pooling. Extensive experiments show that our approach can generate better motion retargeting results both qualitatively and quantitatively than state-of-the-art methods. Moreover, we also show that our framework can generate reasonable results even for a more challenging retargeting scenario, like retargeting between bipedal and quadrupedal skeletons because of the body part retargeting strategy and PAN. Our code is publicly available[1].

Index Terms—Deep Learning, Motion Processing, Motion retargeting

1 Introduction

Articulated motion data plays a crucial role in computer animation, virtual reality and the game industry, since most of the virtual characters are driven by the articulated skeletons. To get the motion data, current methods are mainly two-fold. The first type is to capture human motions using a motion capture system, the other one is to create animations by artists using key-frame technology. However, both methods require a major expenditure of time and effort, and the obtained motion data cannot be directly applied to each other between skeletons because of the differences in structure. Motion retargeting, as one of the motion-reusing technology, has been regarded as a promising way that enables the transfer of a character's motion to another skeleton. It is widely used as a motion pre-processing tool since it can integrate various motion datasets [1] for neural network training. In physical simulation and control [2], [3], [4], motion retargeting also plays an important role in transferring the input signals from the real environment into the simulation setting.

Though motion retargeting is a long-standing problem that has been studied for decades, there are still two main issues that restrict the large-scale application of this technology. First, motion retargeting essentially requires flexible source-target correspondence. However, we observe that previous motion retargeting methods [5], [6], [7] have some strict requirements, such as the source-target skeletons needing to have the same number of joints or share the same topology. These requirements will reduce the flexibility of the motion retargeting model and limit the usage scenarios. Another issue is the balance between the accuracy and automation of motion retargeting. Traditional works [8], [9], [10] regard motion retargeting as a space-time optimization problem and employ the inverse kinematic technology to precisely satisfy the user-given constraints. However, users need to design the energy functions manually, making the whole process semi-automated. Recent works [5], [6], [7], [11] achieve automatic motion retargeting using the deep learning methods. However, the retargeting accuracy is still limited due to the lack of spatial modeling of articulated motion.

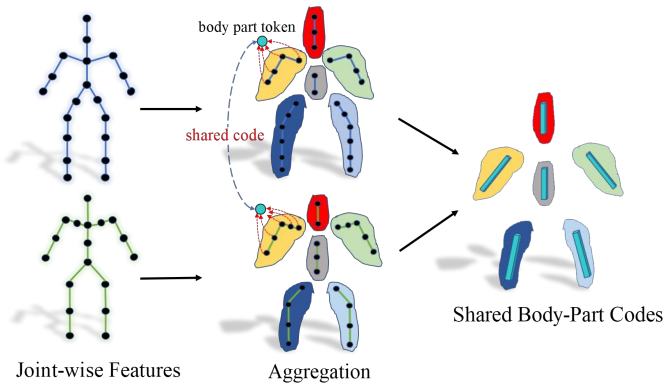

Fig. 1. Feature Pooling at body part level. We pool the joint-wise features (black dots) into body part tokens (cyan dots) to build the shared latent codes in terms of body parts.

- † *Equal contributions*
- * *Corresponding author*
- *Lei Hu, Chongyang Zhong, Boyuan Jiang and Shihong Xia are with Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 100190, China.*
  E-mail: hulei19z, zhongchongyang, jiangboyuan20s, xsh@ict.ac.cn
- *Zihao Zhang is with Institute of Computing Technology, Chineses Academy of Sciences, Beijing 100190, China.*
  E-mail: zhangzihao@ict.ac.cn

[1] *https://github.com/hlcdyy/pan-motion-retargeting*

To tackle the above issues, it first needs to define a flexible correspondence between the source and target skeletons. Therefore, we propose to treat the body part as the basic retargeting unit to extract shared latent codes cross structure (see Figure 1). Our key observation supporting this strategy is that skeletons with different structure still have some common body parts sharing the same semantic meaning. Using the body part as the retargeting unit not only helps us improve the flexibility of retargeting, but also gives the neural network a geometric prior. To enhance the spatial modeling capability of the model, we further introduce a novel pose-aware attention network (PAN). Inspired by PFNN [12] and its follow-up works [13], [14], [15], we find that the dynamic motion modeling based on the state/pose is beneficial for generating high-quality motions. Therefore, we use the proposed PAN to dynamically predict the weights of each joint for feature blending and pooling. Moreover, the dynamic spatial modeling meets our intuition that the contribution of a fixed joint to its corresponding body part varies with the body part's motion.

Given an articulated motion of the source skeleton, we first process the motion frame-by-frame using our PAN to extract the spatial features at the body part level. Specifically, we introduce a trainable parameter for each body part called "body part token" (see cyan dots in Figure 1) and compute the dot-product attention weights together with the hidden features of the inner-part joints. Through several attention layers, the joint-wise features will be blended into the "body part tokens" by weighted summation. For feature pooling, we keep the parameters of the body parts, i.e. "body part tokens" and discard the hidden features of joints in the last layer, so that these "body part tokens" can be shared among skeletons with different structure. Then, we further compress the "body part tokens" along the temporal dimension by 1D convolution to extract the shared motion code. Finally, we combine the shared motion code and the deep representation of the target skeleton offsets to generate the retargeted motion by the motion decoder of the target structure.

Since paired motion data is hard to acquire in this task, we train our architecture in an unsupervised manner and use a motion discriminator for each articulated structure to ensure the retargeted motion falls into the corresponding motion manifold. Experiments on Maximo [16], Human3.6M [17], lafan1 [18], and quadruped [13] datasets show that our method can achieve state-of-the-art performance.

Our approach can effectively address the issues mentioned above, enabling automatic, accurate, and flexible retargeting. The main contributions of this work can be summarised as follows:

1. We propose a novel pose-aware attention network (PAN) that can dynamically extract the spatial features of motion, which is beneficial for improving the accuracy of automatic retargeting.

2. We propose a powerful motion retargeting framework that uses body parts as retargeting units, improving the flexibility in processing different source-target structure pairs.

3. Through extensive experiments, we show that our method can generate high-quality results for human motion retargeting task. Because of our body-part retargeting strategy and pose-aware attention network, we find that our method can even fulfill the retargeting task between bipedal and quadrupedal skeletons. To the best of our knowledge, we are the first to solve such motion retargeting problems without manual effort.

2 Related Work

In this section, we will review related research on motion retargeting, motion processing with body parts, and dynamical motion modeling methods that are similar to our thinking.

2.1 Motion Retargeting

One of the earliest motion retargeting methods was proposed by Gleicher [8], which regards retargeting as a space-time constraint problem. As most of the constraints in retargeting are kinematic related, inverse kinematics solver [9], [10], [19] is widely adopted for this task. Besides, there is some literature [20], [21] that takes dynamics into consideration for preserving essential physical properties of the motion during retargeting. When the source and target skeletons differ greatly, the key-framing technique plays an important role in many works [19], [22], [23]. Yamana et al. [22] combine the key-frame selection with a GPLVM-based model to learn the static and dynamic mapping between the source and target poses for anthropomorphic characters' retargeting. Yeongho Seol et al. [23] ask the users to manually specify several features representing the source motion and build a paired data library through key-framing. However, most traditional motion retargeting approaches rely on hand-crafted constraints and iterative solvers, making the workflow semi-automatic and tedious.

Recently, with the fast development of deep learning and articulated motion datasets, end-to-end motion retargeting methods have achieved remarkable success. Jang et al. [24] use convolutional encoder-decoder architecture to model the source motion and limb ratios for retargeting between skeletons with different bone lengths. Villegas et al. [5] train an RNN-based encoder-decoder network to alternate inverse kinematics module. Aberman et al. [11] propose the skeleton-aware convolution and pooling operations to extract a primal skeleton for handling topologically equivalent retargeting. In order to reduce interpenetration and preserve self-contacts, Villegas et al. [7] combine iterative optimization and geometry-conditioned RNN to achieve real-time retargeting between different mesh geometries. Though the deep learning-based methods achieve automatic motion retargeting, the lack of spatial modeling of the skeletal structure makes the retargeting accuracy limited. Skeleton-aware network [11] can handle different structure to some extent, but the flexibility of their correspondence strategy is limited because all its operations are based on the neighborhoods in the kinematic chain. These operations do not work well for constructing correspondence in some cases, such as retargeting between quadrupeds and bipeds.

2.2 Motion Processing with Body Parts

Subdividing the whole body motion into several partial movements is widely used in motion splicing and style

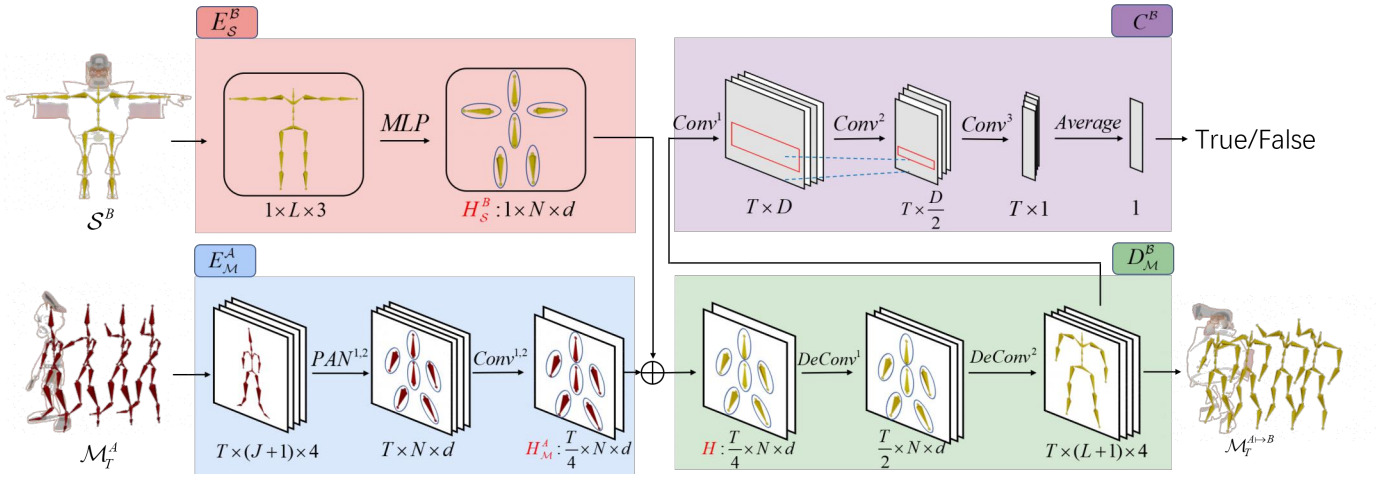

Fig. 2. The overall architecture of our motion retargeting framework. The transparent meshes are overlaid on the first frame of source motion and the canonical pose of the target skeleton to highlight the differences between skeleton A and B. The motion code $H_{\mathcal{M}}^A$ produced by $E_{\mathcal{M}}^A$ will be modified by skeleton code $H_{\mathcal{S}}^B$ in an additive manner, i.e, $H = H_{\mathcal{M}}^A + H_{\mathcal{S}}^B$ ($H_{\mathcal{S}}^B$ will first be replicated along the temporal axis.)

transfer. Based on the source of the partial movements, we can divide the body part-based methods into two categories. 1. The movement of each body part comes from different motions. 2. The movement of each body part comes from a single motion. Our body part-based retargeting method falls into the second category.

For the first category, the difficulty of splicing lies in choosing the appropriate combination of partial motions, since the movements of the individual body parts are usually coupled with each other. To tackle this problem, traditional methods use segmentation, clustering [25], and example-based techniques [26], [27] to measure the similarity of partial motions to ensure the naturalness of the splicing motion. However, these splices are often performed in the original space or linear embedding space reduced by PCA, thus limiting the splicing to analogous motions. Thanks to the non-linear modeling of deep neural networks, recent methods have tried to splice the motion features of different body parts in the hidden space. Ye et al. [4] use three points of the VR device as input and predict the movement of the upper and lower body respectively, and then splice them together. Motion Puzzle [28] transfers the style of each body part from different motion clips to the content motion by graph convolutional network. Lee et al. [29] propose a novel part assembler layer for splicing the part motions from the different creatures. This assembler can search for the spatial alignment among body parts after the temporal alignment. These works show that it is a good choice to combine the motion features of different body parts in deep motion space.

There is no ambiguity in the splicing phase for the second category because all the partial movements come from the same motion clip. However, if the target skeleton is different from the source motion skeleton, we still need to modify the motion features of each body part to match the target skeleton. Abdul-Massih et al. [30] decompose motion style into a set of features present in distinct groups of body parts and use optimization methods to solve the retargeting between skeletons with different morphologies. However, this approach requires manual feature design and spatial-temporal alignment, making the whole process semi-automatic. Liao et al. [31] propose a skeleton-free pose transfer network to automatically learn the skinning weights and transformation in terms of deformation parts. But, it can only process a single pose rather than motion. We regard the body parts as retargeting units and employ deep neural networks to automatically construct the shared latent spaces between skeletons with different structure, but have common body parts.

2.3 Dynamical Motion Modeling

In motion retargeting, we need to encode the source motion features, which is actually a spatio-temporal modeling process. Dynamic modeling of articulated motion has been included in many studies of various tasks. In this part, we will briefly review some of the methods related to our methodological ideas. Dynamical motion modeling is crucial for many applications, including style transfer, motion synthesis, prediction, etc. We divide dynamic motion modeling into two categories, one for dynamically changing the network's weights such as PFNN [12], and the other is dynamically changing the feature vectors such as Transfomer [32].

For dynamically changing the control weights, Xia [33] proposed an online mixture-of-autoregressive model for real-time motion style interpolation and control by blending the parameters of distinctive styles. PFNN [12] has achieved remarkable success in data-driven motion synthesis thanks to its phase-functioned model. It works by generating the weights of a regression network at each frame as a function of the phase, thus enabling smooth transitions between different motion states with good stability and producing high-quality motion. Follow-up works [13], [14], [34], [35] employ a gating network to automatically learn the blending coefficients of different experts which allows the network to cluster the poses based on different states or phases. The graph convolutional neural network is also a common tool for modeling human body motion, but the adjacency matrix of the vanilla network is fixed. To make the graph connections more flexible to portray the relationships

between joints, Lei Shi et al. [36] add a learnable matrix and a data-dependent graph to change the adjacency matrix based on the input data. Recently, Zhong et al. [15] introduce the gating adjacency matrix into the graph convolution network for motion prediction, and the core idea is also to dynamically change the graph convolutional weights to improve the expression of the model. Although the mixture-of-experts model has excellent motion modeling capabilities, the size of the model increases proportionally with the number of experts.

The idea of the Transformer [32] is similar to the mixture-of-experts model, except that the dynamic modeling is achieved by generating attention weights through computing the dot products of the query with key vectors. The advantage of this model is that the network size can be controlled to stack more layers. This architecture is currently widely used for pose estimation [37], [38], [39], action-gesture recognition [40] and motion synthesis [41]. In motion style transfer, Jang et al. [28] use attention network transfers the locally semantic style features into the decoded content features. In this work, we employ the self-attention mechanism in source motion encoding process and incorporate the body part strategy to achieve pose-aware spatial feature extraction for motion retargeting. The proposed pose-aware attention mechanism can dynamically generate the attention weights based on the input poses to aggregate the joint-wise features at the body part level. Because of the dynamic spatial modeling, we are able to achieve more accurate motion retargeting.

2.4 Other Related Works

There are some loosely related problems including mesh deformation transfer and 2D motion retargeting. Deformation transfer [42], [43], [44], [45] aims to adapt the deformation from a source mesh model to another mesh. These methods often require building dense correspondences between the meshes. Recently, Gao [45] proposes a method that can transfer the deformation between two unpaired mesh models without defining the correspondences. The cycle consistency loss used in their work is similar to ours, except that we impose consistency constraints on both the latent space and the original rotation space and focus on the articulated skeleton.

There are also some works [46], [47] that bypass the explicit 3D motion representation to directly implement 2D motion retargeting, but these often incorporate viewpoint and texture generation. In contrast, we mainly focus on 3D motion retargeting in this work.

3 Data Representation and Overview

In this section, we will first describe the representation of articulated motion data, skeleton representation, and the symbols used in our paper. Then, we will further introduce the overview architecture of our method.

3.1 Data representation

We denote an articulated motion of length T as $\mathcal{M}_T = [M_1, M_2, ..., M_T]$ and suppose that the skeleton A belonging to structure $\mathcal{A}$ has J joints. The motion attributes used in our representation include local joint rotations of each joint represented by unit quaternions denoted as $Q^{T\times J\times 4}$, global motion velocity $V^{T\times 1\times 3}$ of the root in x,y, and z directions and additional angular velocity $R^{T\times 1\times 1}$ representing the rotation velocity around the axis perpendicular to the ground (y-axis in our setup). We concatenate the velocity V and angular velocity R in the last dimension to represent a new combined velocity vector $\bar{V}^{T\times 1\times 4}$, and the motion $\mathcal{M}_T$ can be written as $\mathcal{M}_T = [Q, \bar{V}] \in \mathbb{R}^{T\times(J+1)\times 4}$ if we regard the root velocity vector $\bar{V}$ as an additional "joint". In articulated skeleton representation, the skeleton topology and bone lengths are usually represented by a set of offsets $S \in \mathbb{R}^{J\times 3}$, the offset of each joint in the skeleton is the 3D vector relative to its parent's coordinate frame in the kinematic chain. The task of motion retargeting is adapting a motion $\mathcal{M}_T^A$ from structure $\mathcal{A}$ with offsets $\mathcal{S}^A \in \mathbb{R}^{J\times 3}$ to skeleton $B \in \mathcal{B}$ with offsets $\mathcal{S}^B \in \mathbb{R}^{L\times 3}$. The retargeted motion is $\mathcal{M}_T^{A\mapsto B} \in \mathbb{R}^{T\times(L+1)\times 4}$ where the joint number L may not be equal to J, but has the same time length T.

In humanoid motion retargeting, we divide the whole body into N=6 body parts based on the main limbs, which are the head, the spine, the left/right arms, and the left/right legs. Specially, we attribute the velocity of root joint to all body parts as special "joint" because its important role in distinguishing the whole-body motion types. To achieve flexible source-target correspondence, we will only construct the common body part motion spaces of the source and target skeletons in practice. For example, when we retarget the motion from the structure of a normal person to the skeleton of a disabled person with only one arm, we construct only N=5 shared body parts. In addition to this, we allow semantic-level correspondence when facing more complex retargeting tasks such as bipeds to quadrupeds, which will be discussed in Sec 7.

3.2 Overview

Figure 2 shows the overall architecture of our approach. Our pipeline is similar to that of the motion Puzzle which is designed for motion style transfer at the body part level. However, our task differs from style transfer in that we take the target skeleton representation rather than the target motions as input. As shown on the left side of the figure, the input of the framework is the source motion $\mathcal{M}_T^A$ performed by skeleton $A \in \mathcal{A}$ and targeted skeleton offsets $\mathcal{S}^B$. The output of our framework is the motion $\mathcal{M}_T^{A\mapsto B}$ that is performed by skeleton $B \in \mathcal{B}$. Our architecture can be divided into four modules, namely, skeleton encoder $E_{\mathcal{S}}^{\mathcal{B}}$, motion encoder $E_{\mathcal{M}}^{\mathcal{A}}$, motion decoder $D_{\mathcal{M}}^{\mathcal{B}}$ and motion discriminator $C^{\mathcal{B}}$. The skeleton encoder $E_{\mathcal{S}}^{\mathcal{B}}$ receives the target skeleton offsets as input and encodes them by a multi-layer perception to generate the skeleton code $H_{\mathcal{S}}^B$. The target skeleton code is composed of the hidden features of N body parts, each with d dimensions. The motion encoder $E_{\mathcal{M}}^{\mathcal{A}}$ is mainly in charge of mapping the source motion $\mathcal{M}_T^A$ into the source motion code $H_{\mathcal{M}}^A$. It includes the pose-aware attention network (PAN) for spatial modeling and the convolution layers for temporal compression. The motion code $H_{\mathcal{M}}^A$ and the skeleton code $H_{\mathcal{S}}^B$ will be fused in an additive manner and pass through the motion decoder $D_{\mathcal{M}}^{\mathcal{B}}$ to generate the retargeted motion $\mathcal{M}_T^{A\mapsto B}$. Since our

networks are trained in an unsupervised manner, we build a discriminator ($C^{\mathcal{B}}$ in Figure 2) for each structure to ensure the retargeted motion falls onto the correspondence motion manifold.

As stated above, in order to enhance the spatial modeling capability of the model, we introduce the pose-aware attention network (PAN) in the motion encoder $E^{\mathcal{A}}_{\mathcal{M}}$ to extract the source motion features in terms of body parts. In the following sections, we will first elaborate on this network.

4 Pose-Aware Attention Network

In this section, we will introduce the most important component of the motion encoder, i.e, the pose-aware attention network(PAN). We will first describe the joint embedding and positional encoding, and finally illustrate our pose-aware attention mechanism.

The choice of using attention networks for spatial feature extraction is based on the idea that the importance of each joint in the kinematic chain is different and varies dynamically with the pose (The visualization shown in Sec 6 will confirm this assumption). This prompts us to employ a neural network to automatically learn the attention weights, aggregating the motion characteristics at the body part level. Transformer [32] achieves remarkable success in natural language processing due to the capacity of the attention mechanism which could automatically depict the association of separate words in a sentence. Inspired by that, we try to model the spatial relationships between joints using the proposed pose-aware attention network.

4.1 Joint Embedding and Positional Encoding

The PAN will process the motion frame-by-frame. Given the input pose represented as $M_t \in \mathbb{R}^{(J+1)\times 4}$ at the timestep t, we need to first map the input to the hidden embedding space through the joint embedding layer, which is shown in Figure 3 (a). This step can be formulated as follows:

$$\Phi(M_t;\alpha) = Relu(Relu(M_t W_0 + b_0)W_1 + b_1)W_2 + b_2 \quad (1)$$

Where the parameters of the network α are defined by $\alpha = \{W_0 \in \mathbb{R}^{4\times h}, W_1 \in \mathbb{R}^{h\times h}, W_2 \in \mathbb{R}^{h\times d}, b_0 \in \mathbb{R}^h, b_1 \in \mathbb{R}^h, b_2 \in \mathbb{R}^d\}$. Here h is the number of hidden units used in non-linear mapping which is 256 in our implementation and the d is the dimensionality of the joint embedding space.

However, directly using the above joint feature is not feasible since the network lacks the ability to know where each joint is located in the kinematic chain. Inspired by the positional encoding in the vanilla attention network (Transformer), we use the joint-level positional encoding to label each joint's location in an additive manner. In our case, we modify the mathematical formulation of positional encodings and use the joint indices in the kinematic chain as the position. The formula is represented as follows:

$$\begin{aligned} PE_{j,2i} &= sin(\frac{j}{basis^{2i/d}}) \\ PE_{j,2i+1} &= cos(\frac{j}{basis^{2i/d}}) \end{aligned} \quad (2)$$

where j indicates the index of the joint in the kinematic chain, d equals to the dimensionality of the embeddings in equation 1 and $i \in [0, ..., d/2]$. The $basis$ which can control the sinusoid's frequency is often set to 10,000 as in most transformer implementations. The joint-level positional encoding is bounded, smooth and dense compared to the one-hot form and the vector dimension of PE is not correlated with the number of skeleton joints, which makes it very suitable for encoding in the case of skeleton structure changes.

After calculating the PE_j for each joint, we further combine the joint embeddings with the positional encodings in an additive manner, which is given by the following equation:

$$X_{t,j} = \Phi(M_{t,j};\alpha) + PE_j \qquad \forall j \in J \quad (3)$$

where $M_{t,j}$ represents the input feature of joint j in pose M_t. Through the non-linear joint embedding and positional encoding, the hidden representation X_t contains information about each joint rotation and location in the kinematic chain.

4.2 Pose-aware Attention at Body Part Level

As our retargeting strategy is to use body parts as shared units, we need to integrate and pool joint-wise features into body part-wise codes. Previous literature [41] proposes to use the "distribution tokens" to pool arbitrary-length motion sequences into one latent space. Inspired by this work, we similarly prepend the joint embeddings with learnable tokens and only use the corresponding attention outputs as a way to pool the joint-wise features into body part level. Specifically, we introduce a learnable parameter with the same dimension d as the joint embedding for each body part, thus yielding a vector $Y \in \mathbb{R}^{N\times d}$ which we called "body part tokens", where the N represent the body part number we have defined. Figure 3(b) shows the operations with an arm part as an example. For each timestep t, the attention function first takes as input a combination of the body part tokens and the joint-level features $Z_t = [Y, X_t] \in \mathbb{R}^{(N+J+1)\times d}$ and outputs the Query vector Q_t and Key-Value pair (K_t and V_t). the formulation can be described as follow:

$$Q_t = Z_t W_Q + b_Q \quad K_t = Z_t W_K + b_K \quad V_t = Z_t W_V + b_V \quad (4)$$

where $W_{[Q,K,V]} \in \mathbb{R}^{d\times d}$ and $b_{[Q,K,V]} \in \mathbb{R}^d$ are learnable matrix and bias vectors, respectively.

Since we regard the body part as a retargeting unit, we will use a mask matrix $U \in \mathbb{R}^{(N+J+1)\times(N+J+1)}$ to isolate the body parts so that the final body-part features are generated purely by pooling the inner-part joints' features. The mask matrix U is symmetric, where $U_{i,j} = 0$ means that joint i and j are in the same body part and vice verse $U_{i,j} = -\infty$. The final output of the attention is computed as a weighted sum of the values:

$$Attention(Q_t, K_t, V_t, U) = softmax(\frac{Q_t K_t^{'} + U}{\sqrt{d}})V_t \quad (5)$$

where $K_t^{'}$ represents the transpose of K_t and $\sqrt{d}$ is used for scaling to prevent pushing the softmax function into regions where it has extremely small gradients [32]. The production of Query Q and the transpose of Key K actually depict the

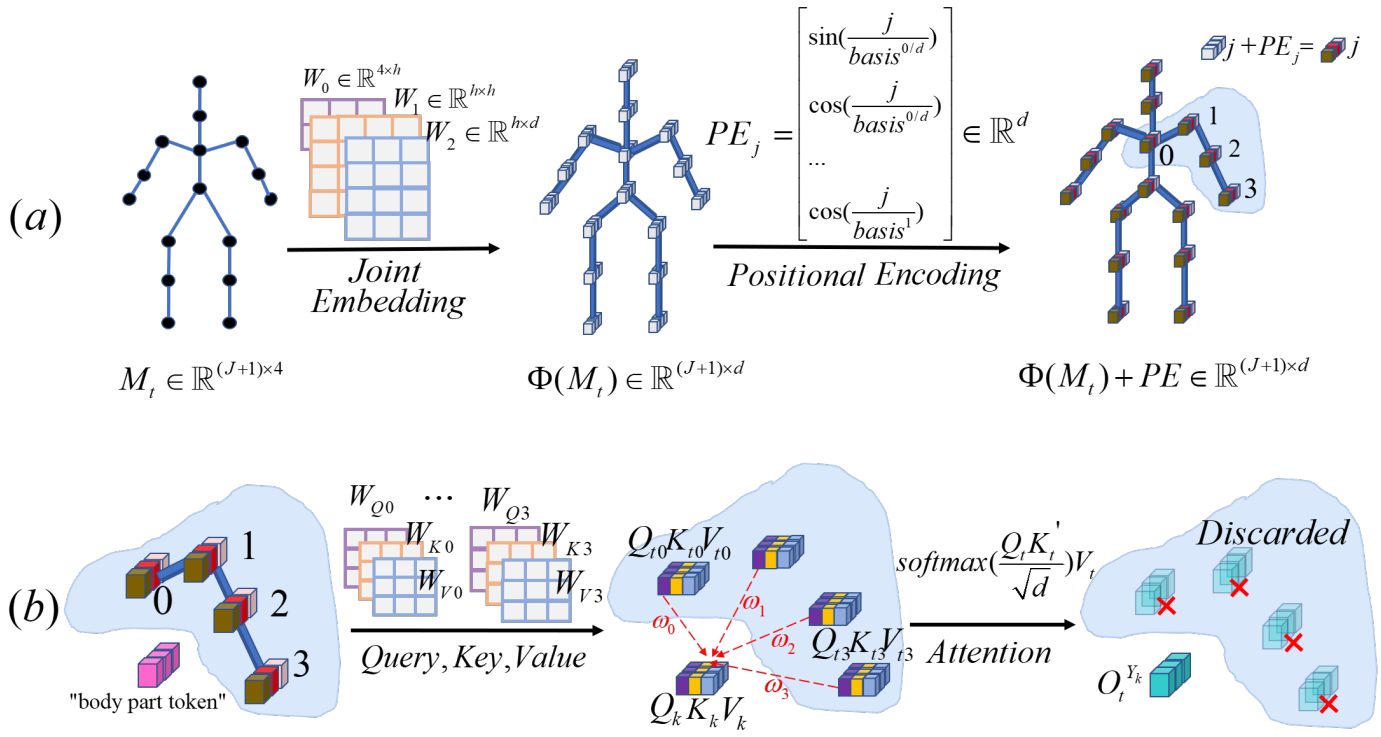

Fig. 3. (a) Joint embedding and positional encoding. The raw joint features will first be embedded by MLP and then modified by positional encodings. (b) Pose-aware attention at the body part level. We use the arm part as a sample to demonstrate the attention process. In the last layer of the attention, the output features $O_t^{Y_k}$ corresponding to the "body part token" will be retained, while other nodes will be discarded.

relationship of the joints. Specifically, the larger the dot-production value of two joints reveals the stronger their relationship. This attention operation can be stacked into several layers to form the PAN, in our case we use 2 layers. In the last layer, we discard the inner-part joint values and only retain the "body part token" values (see Figure 3(b)).

Substitute equation 4 in equation 5, we can easily get a new formulation $\Psi(Z_t;\beta)$, where the parameters are defined by $\beta = \{W_Q, W_K, W_V, b_Q, b_K, b_V, U\}$ (all except U are trainable). Our attention network is pose-aware since the attention weights calculated by $Q_tK_t^{'}$ are dependent on the variable Z_t, which is related to the input pose M_t according to equation 3. Therefore, the proposed PAN can dynamically extract the spatial features of the articulated motion.

As stated above, we only use the output corresponding to the "body part token". Therefore, denote the complete output of the $\Psi(Z_t;\beta)$ as $O_t = [O_t^Y, O_t^X] \in \mathbb{R}^{(N+J+1)\times d}$, we only keep the $O_t^Y \in \mathbb{R}^{N\times d}$ for pooling purpose.

5 Architecture Modules and Training Process

In this section, we will describe in detail the modules of our architecture including the motion encoder, skeleton encoder, motion decoder, and discriminator. Then we will show the training/testing process and the loss functions we used.

5.1 Motion Encoder

The motion encoder is designed to extract the source motion code $H_{\mathcal{M}}^A$ from spatio-temporal dimension. The process of motion encoding is shown in the blue part ($E_{\mathcal{M}}^{\mathcal{A}}$) of Figure 2.

The motion encoder consists of two blocks, the pose-aware attention PAN$^i, i \in \{1,2\}$ and the temporal convolutional $Conv^i, i \in \{1,2\}$. Given a source motion $\mathcal{M}_T^A$, the pose-aware attention network generate the hidden features of N body parts, i.e, $O^Y = [O^{Y_1}, O^{Y_2}, ..., O^{Y_N}], O^{Y_k} \in \mathbb{R}^{T\times d}$. The PAN is mainly in charge of extracting spatial information. As for the temporal compression, we use the multi-level temporal convolution $Conv$, which is proven to be influential in building motion manifold [48], [49] as well as retargeting [11]. In our setting, the convolution block $Conv$ takes each body part features O^{Y_k} as input, and the temporal modeling process can be described as follows:

$$Conv_k = Relu(O^{Y_k} * W_k^{conv} + b_k^{conv}) \qquad \forall k \in 1, 2, ..., N \tag{6}$$

where $*$ means the convolution operation, $W_k^{conv} \in \mathbb{R}^{h\times d\times w}$ is the weights matrix with temporal filter width of w, and $b_k^{conv} \in \mathbb{R}^h$ is the bias with h hidden units in the convolutional layer. The filter width w is set to 15 so that the receptive field of the filter can roughly cover a half second of motion, which is proven to be efficient in work [49]. We set the number of hidden units to 32 for each body part because we experimentally find it can produce good reconstruction and retargeting results. The stride of all the convolution kernels is set to 2 for temporal compression purpose. Through the $Conv^i, i \in \{1,2\}$ we eventually obtain the shared motion code $H_{\mathcal{M}}^A \in \mathbb{R}^{\frac{T}{4}\times N\times d}$.

5.2 Skeleton Encoder

The skeleton encoder (see the pink part ($E_{\mathcal{S}}^B$) of Figure 2) aims to transform the target skeleton offsets into latent

codes in terms of body parts. Since the offset vectors, which express the skeleton topology and bone lengths, can be regarded as a single canonical pose, we only spatially encode the offset vectors. Given the target skeleton offsets S^B with L joints, we utilize a multi-layer perceptron (three layers in our work) to map the raw vectors to the skeleton code $H_S^B \in \mathbb{R}^{1\times N\times d}$. Specifically, for each body part, we concatenate the corresponding joints' offsets to form a vector $S_k \in \mathbb{R}^{3L_k}$, $k \in [1, 2, ..., N]$ and pass through the following equation:

$$\omega(S_k;\gamma_k) = Relu(Relu(S_kW_{k0} + b_{k0})W_{k1} + b_{k1})W_{k2} + b_{k2} \tag{7}$$

Where the parameters of MLP are defined by $\gamma_k = \{W_{k0} \in \mathbb{R}^{3L_k\times h}, W_{k1} \in \mathbb{R}^{h\times h}, W_{k2} \in \mathbb{R}^{h\times d}, b_{k0} \in \mathbb{R}^h, b_{k1} \in \mathbb{R}^h, b_{k2} \in \mathbb{R}^d\}$. Here h is the number of hidden units which is 64 in our implementation, and L_k is the number of joints in kth body part. We stack the latent codes belonging to different body parts to generate the final skeleton code $H_S^B = [\omega(S_1;\gamma_1), ..., \omega(S_N;\gamma_N)] \in \mathbb{R}^{1\times N\times d}$

5.3 Motion Decoder

Through the motion encoder $E_{\mathcal{M}}^{\mathcal{A}}$ and skeleton encoder $E_{\mathcal{S}}^{\mathcal{B}}$, we obtain the source motion code $H_{\mathcal{M}}^A$ and target skeleton code H_S^B, respectively. To integrate these two codes, we first repeat the target skeleton code H_S^B along the temporal axis to make it consistent with the shape of $H_{\mathcal{M}}^A$ and then add it to $H_{\mathcal{M}}^A$. The reason we directly add them is our motion and skeleton codes are both composed in the form of body parts. Direct summation enables body-part codes to correspond to each other (e.g., arm motion and arm skeleton) The motion decoder (see green part ($D_{\mathcal{M}}^{\mathcal{B}}$) in Figure 2) receives the fused motion code H, and generate the retargeted motion $\mathcal{M}_T^{A\mapsto B}$ by $Deconv$ blocks. The $Deconv$ block consists of up-sampling and deconvolution as follows:

$$Deconv = Relu(\uparrow H * W^{dec} + b^{dec}) \tag{8}$$

Where the weights matrix $W^{dec} \in \mathbb{R}^{h\times Nd\times w}$ and bias vector $b^{dec} \in \mathbb{R}^h$ are defined similar with formulation 6. The difference is that we integrate all the body-part features to synthesize the retargeted motion of the whole body i.e., the deconvolutional kernel W^{dec} is not body part-wise and will act on the fused features H to consider the relationship between different body parts. The number of $Deconv$ blocks is 2 and we use the same kernel width and hidden size with $Conv$ blocks (see equation 6), but the stride of the kernel is 1. We set the up-sampling factor as 2 in order to recover the temporal dimension layer by layer, and in the last layer we restore the spatial dimension of the target motion, i.e.$Nd \to (L+1)\times 4$. It should be noted that the attention weights computed in the motion encoder will not be used for decoding. In the last layer, the $Relu$ activation will not be used and the convolution operation will synthesize the retargeted motion performed by skeleton B with the same sequence length T.

5.4 Motion Discriminator

The retargeted motion generated by the decoder needs to fall onto the motion manifold of the target structure. Therefore, we build a motion discriminator for each skeletal structure as shown in the purple part ($C^{\mathcal{B}}$) of Figure 2. The discriminator $C^{\mathcal{B}}$ consists of several convolution layers and receives the motion $\mathcal{M}_T^{A\mapsto B}$ as input. Each layer of $C^{\mathcal{B}}$ aims to progressively compress the feature size from $4(L+1)$ to $D/2$, here the D is set to 256. The convolutional operation is similar with equation 6, but not body part-related. The final output of the motion discriminator is a one-dimension feature in the range of (0, 1) by using the sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{9}$$

The average score indicates the naturalness of the retargeted sequence. We train the discriminator and motion generator alternatively to encourage the motion decoder to synthesize natural-looking motions, the detailed process is demonstrated in section 5.5

5.5 Training and Testing Process

One of the main difficulties our method faces is acquiring paired motion data in real-world scenarios. Even if the same actor repeatedly performs the same motion clip, there will still be slight differences. We follow the training strategy of SAN [11], which uses cyclic adversarial learning to train the encoder-decoder pairs of different structure for the purpose of self-supervision. This strategy can overcome the data deficiency problem. Specifically, we take the motion encoder, skeleton encoder, and motion decoder together as a set called generator $G = \{E_{\mathcal{M}}, E_{\mathcal{S}}, D_{\mathcal{M}}\}$, we alternatively update the parameters between generator G and discriminator C while fixing the parameters of the other network. In our implementation, we employ the one-to-one ratio (1 discriminator iteration per generator update) which we empirically found can maintain the balance between the generator and discriminator.

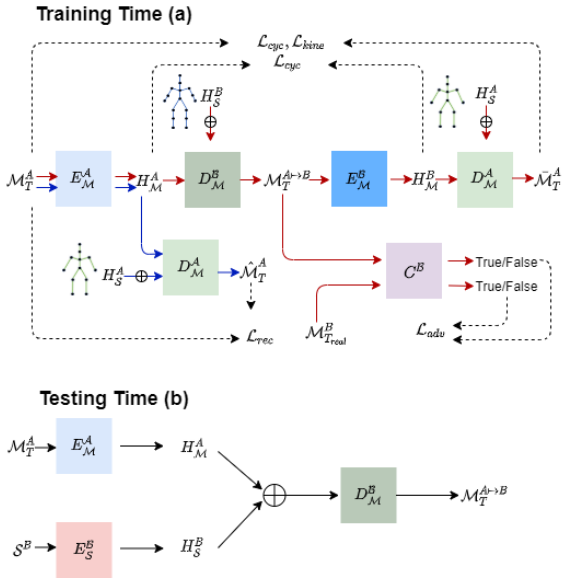

Fig. 4. (a) Training process. The blue route shows the reconstruction process, which is used to train the encoder-decoder pairs. The red route shows the retargeting and cyclic retargeting process. (b) Testing Process. We discard the discriminator when inference.

Figure 4 (a) shows the training process (we take the retargeting $A \to B$ as an example, $B \to A$ is symmetrically

equivalent). We use the motion encoder $E^{\mathcal{A}}_{\mathcal{M}}$ to encode the source motion $\mathcal{M}^A_T$ of skeleton A and then there are two branches (blue and red route in Figure 4). One is to reconstruct the motion $\hat{\mathcal{M}}^A_T$ by motion decoder $D^A_{\mathcal{M}}$ for training the encoder-decoder pair (blue route in Figure 4). The other branch (red route) is to utilize the decoder $D^B_{\mathcal{M}}$ with target skeleton code $H^B_{\mathcal{S}}$ to synthesize the retargeted motion $\mathcal{M}^{A\mapsto B}_T$. The motion discriminator $C^{\mathcal{B}}$ will be used to judge whether the retargeted motion is real or not. Since the latent motion code $H^A_{\mathcal{M}}$ is constructed by common body parts, we want it could be shared between different skeletal structure. Therefore, we extract the motion code $H^B_{\mathcal{M}}$ of the retargeted motion $\mathcal{M}^{A\mapsto B}_T$ by the corresponding motion encoder $E^{\mathcal{B}}_{\mathcal{M}}$, i.e, we cyclic retarget the motion $\mathcal{M}^{A\mapsto B}_T$ to $\bar{\mathcal{M}}^A_{\mathcal{M}}$. According to the idea, the $H^A_{\mathcal{M}}$ should be similar to $H^B_{\mathcal{M}}$. In addition to the latent code consistency, we also require the consistency in the original motion space when cycle retargeting, i.e., the motion $\bar{\mathcal{M}}^A_{\mathcal{M}}$ should be similar to the source motion $\mathcal{M}^A_T$.

In the testing process (see Figure 4 (b)), the motion discriminator will be discarded, and we use source motion encoder $E^{\mathcal{A}}_{\mathcal{M}}$ and target skeleton encoder $E^{\mathcal{B}}_{\mathcal{S}}$ to synthesis the retargeted motion $\mathcal{M}^{A\mapsto B}_T$ through the motion decoder $D^{\mathcal{B}}_{\mathcal{M}}$. It is worth noting that the source and target structure can either be the same or different during training and testing.

5.6 Loss functions

According to the description of the training process, we are able to conclude the loss functions of our networks, which are similar to SAN [11]. We take the retargeting $A \rightarrow B$ as an example to show the following loss terms:

Reconstruction loss. In order to construct encoder-decoder pairs for different structure. We enforce the network to reconstruct the input samples. The loss term can be formulated as follow:

$$\mathcal{L}_{rec} = \|\mathcal{M}^A_T - \hat{\mathcal{M}}^A_T\|^2 \tag{10}$$

Where the $\hat{\mathcal{M}}^A_T$ is produced by its own motion encoder $E^{\mathcal{A}}_{\mathcal{M}}$, skeleton encoder $E^{\mathcal{A}}_{\mathcal{S}}$, and motion decoder $D^{\mathcal{A}}_{\mathcal{M}}$ (see the blue route in Figure 4(a)).

Cycle consistency loss. We regard each body part as our retargeting unit and suppose the motion encoder can produce the shared motion code $H^A_{\mathcal{M}}$. Therefore, to ensure the motion code we learned as the common motion features, we encourage the motion codes $H^A_{\mathcal{M}}$ and $H^B_{\mathcal{M}}$ to be as close as possible to each other in the latent space. At the same time, we encourage the motion $\mathcal{M}^A_T$ is similar to the cyclic retargeting motion $\bar{\mathcal{M}}^A_T$ in the original representation space by the following loss term:

$$\mathcal{L}_{cyc} = \|H^A_{\mathcal{M}} - H^B_{\mathcal{M}}\|^2 + \|\mathcal{M}^A_T - \bar{\mathcal{M}}^A_T\|^2 \tag{11}$$

Adversarial loss Since we train our networks in an unsupervised manner, we need a discriminator to ensure the retargeted motion $\mathcal{M}^{A\mapsto B}_T$ looks plausible and falls onto the motion manifold of the corresponding structure $\mathcal{B}$. Therefore, we use the adversarial loss term described as follows:

$$\mathcal{L}_{adv} = \|C^{\mathcal{B}}(\mathcal{M}^{B_{real}}_T)\|^2 + \|1 - C^{\mathcal{B}}(\mathcal{M}^{A\mapsto B}_T)\|^2 \tag{12}$$

Where the $\mathcal{M}^{B_{real}}_T$ are real examples of structure $\mathcal{B}$. When training the discriminator alone, we detach the retargeted motion generated by the motion generator from the calculation graph and maximize the loss function 12 to encourage the discriminator to distinguish between real data and synthesized motions. While training the generator, we minimize the function 12 as well as other loss terms to fool the discriminator for obtaining more realistic retargeting results.

Kinematic loss. The motion representation $\mathcal{M}^A_T$ is sufficient to determine a motion when combined with offsets S^A. But, sometimes people are more interested in the correspondence in Cartesian space, especially the position of end-effectors such as foot contacts. Therefore, we transform the local rotation of each joint into 3D coordinates in Cartesian space by forward kinematics (FK). Then we encourage the networks to minimize the joint position errors when reconstruction and cyclic retargeting.

$$\mathcal{L}_{kine} = \|FK(\mathcal{M}^A_T) - FK(\bar{\mathcal{M}}^A_T)\|^2 + \|FK(\mathcal{M}^A_T) - FK(\hat{\mathcal{M}}^A_T)\|^2 \tag{13}$$

where the $\bar{\mathcal{M}}^A_T$ and $\hat{\mathcal{M}}^A_T$ are cyclic retargeted and reconstruction motions, respectively.

The final loss function can be summarized as follow:

$$\mathcal{L}_{total} = \lambda_1\mathcal{L}_{rec} + \lambda_2\mathcal{L}_{cyc} + \lambda_3\mathcal{L}_{kine} + \lambda_4\mathcal{L}_{adv} \tag{14}$$

In our implementation each λ value is set to 1, 2.5, 10^2, 1, respectively.

5.7 Implementation Details

Our architecture is trained for 1000 epochs with a batch size of 128 under the PyTorch platform. We use Adam optimizer [50] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ to train the generator and the discriminator, whose learning rates are both set to 10^{-3}. The training time is about 12 hours with NVIDIA RTX 3090ti.

We localize motion frames by rotating the root on the y-axis so the character is always facing one direction(z-axis positive in our case) which is proven to be beneficial for the convergence of neural networks [12]. After processing, the information of global translation, as well as the y rotation of the root, is contained in the velocity vector $\bar{V}$. To enable training the network in batches, we cut the motions in the datasets into motion clips with fixed length $T = 64$. All examples of animation are downsampled at a rate of 30 fps, so the clip length T is able to make the networks distinguish most types of motion without affecting the training efficiency. We normalized each motion data $\mathcal{M}_T$ by z-score as follow:

$$\mathcal{M}_T = \frac{(\mathcal{M}_T - \mu_{\mathcal{M}})}{\sigma_{\mathcal{M}}} \tag{15}$$

Where the $\mu_{\mathcal{M}}$ and $\sigma_{\mathcal{M}}$ represent the mean and standard deviation of all motion data $\mathcal{M}_T$ in the training dataset, respectively.

6 Retargeting between Humanoid Skeletons

In this section, we mainly evaluate the effectiveness of the proposed method in motion retargeting between humanoid skeletons. For more qualitative results, please refer to the supplementary video.

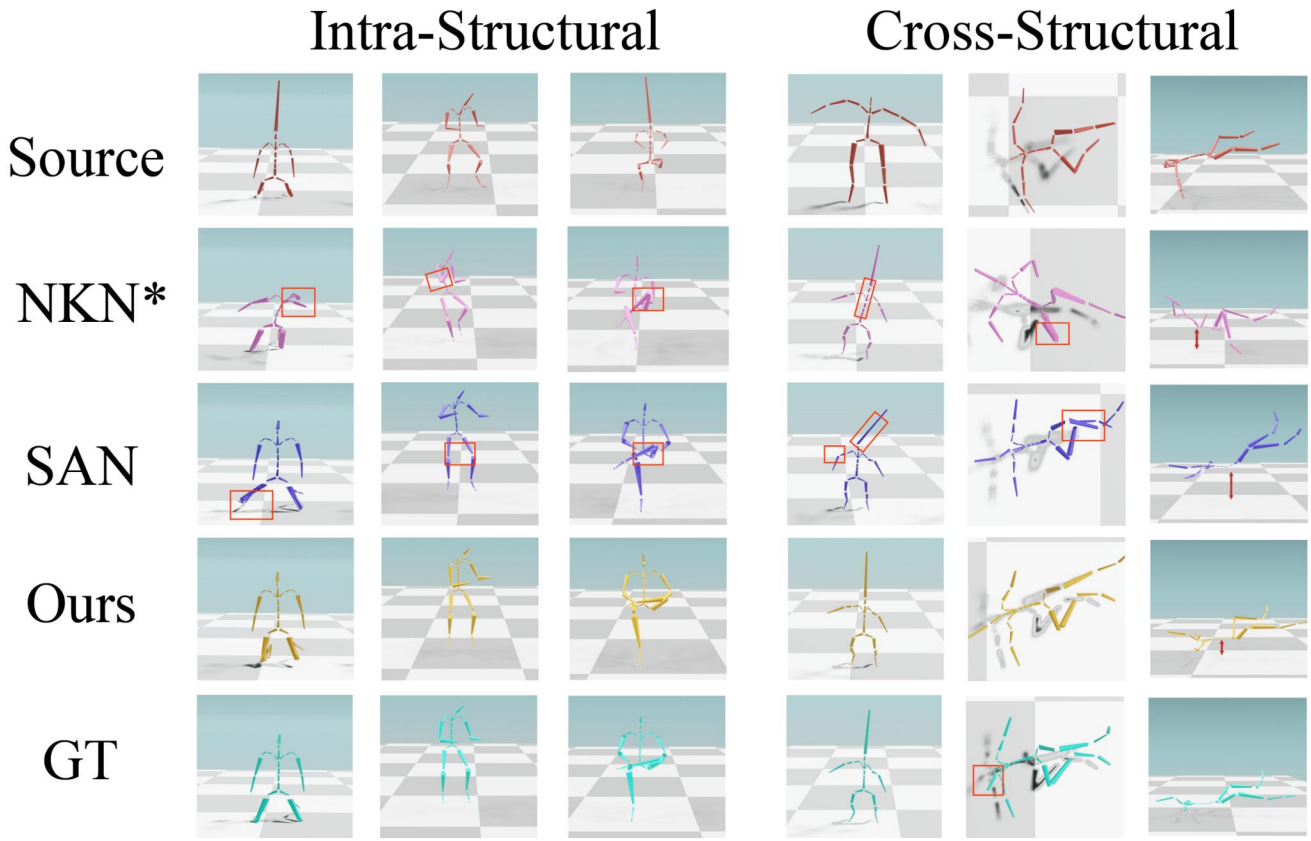

Fig. 5. Qualitative comparisons between our method, SAN, modified NKN, and the corresponding ground truth. The first row displays the source poses while the other rows show the results of the Retargeting. Flaws in the results are marked by red rectangular boxes and arrows.

6.1 Problem Setting

Humanoid retargeting is divided into two cases based on the differences between the source and target structure. One is intra-structural retargeting, i.e., the target skeleton has the same structure and joint number, but different bone proportions. The other is cross-structural retargeting, where the target skeleton has a similar topology to the source skeleton but a different number of joints. To take both cases into account, we follow the work [11] and divide the characters into two groups, $\mathcal{A}$ and $\mathcal{B}$, each containing skeletons with the same structure but different bone proportions. Compared to structure $\mathcal{B}$, the skeletons of structure $\mathcal{A}$ contain extra joints on each limb(left/right arms, left/right leg, spine, and neck).

Dataset. The Mixamo [16] is a 3D motion dataset with rich motion types and contains more than 2000 sequences performed by 29 distinct humanoid characters, of which structure $\mathcal{A}$ includes 24 characters and structure $\mathcal{B}$ contains 5 characters. In our experiments, we select 20 characters in the group $\mathcal{A}$ and 4 characters in the group $\mathcal{B}$ for training, and the remaining characters are used for testing. In addition, we follow the literature [5], [11] to clip the fingers of each character and keep the main limb joints for simplification.

Comparison Methods. The methods we compared are NKN [5], PMnet [6] and SAN [11]. NKN is a pioneering work in deep learning-based motion retargeting. It uses Recurrent Neural Network(RNN) to model the temporal relationship of motion and combine the skeleton offsets for intra-structural retargeting. PMnet [6] learns frame-by-frame poses and overall movement separately for improving the retargeting accuracy. The most similar work to ours is SAN [11], it extracts common skeletal codes by skeleton-based graph convolution which can be applied to skeletons with different structure. We will demonstrate in the following sections that our attention-based spatial modeling at the body part level is more conductive and flexible to the task of motion retargeting.

6.2 Experiments and Evaluation

We use skeletal structure $\mathcal{A}$ to evaluate the intra-structural retargeting. At this time, our modules in the architecture degenerate to the special case, that is, the structure of the target skeleton is the same as that of the source skeleton, both of which are $\mathcal{A}$. The different skeletons $[A_1, A_2, ..., A_n] \in \mathcal{A}$ are distinguished by offsets $\mathcal{S}^{A_i}$ which represents the skeleton topology and bone proportion.

To evaluate the cross-structural retargeting, we allow the motions to be adapted between structure $\mathcal{A}$ and $\mathcal{B}$. Since the target skeleton has a different number of joints than the source skeleton, both the vanilla NKN and PMnet models are no longer applicable to this case due to the requirement of the same dimension between the input and output skeletons. Therefore, we modified these models by retraining the encoder-decoder pair of each structure and forcing the latent code dimension to be the same in order to

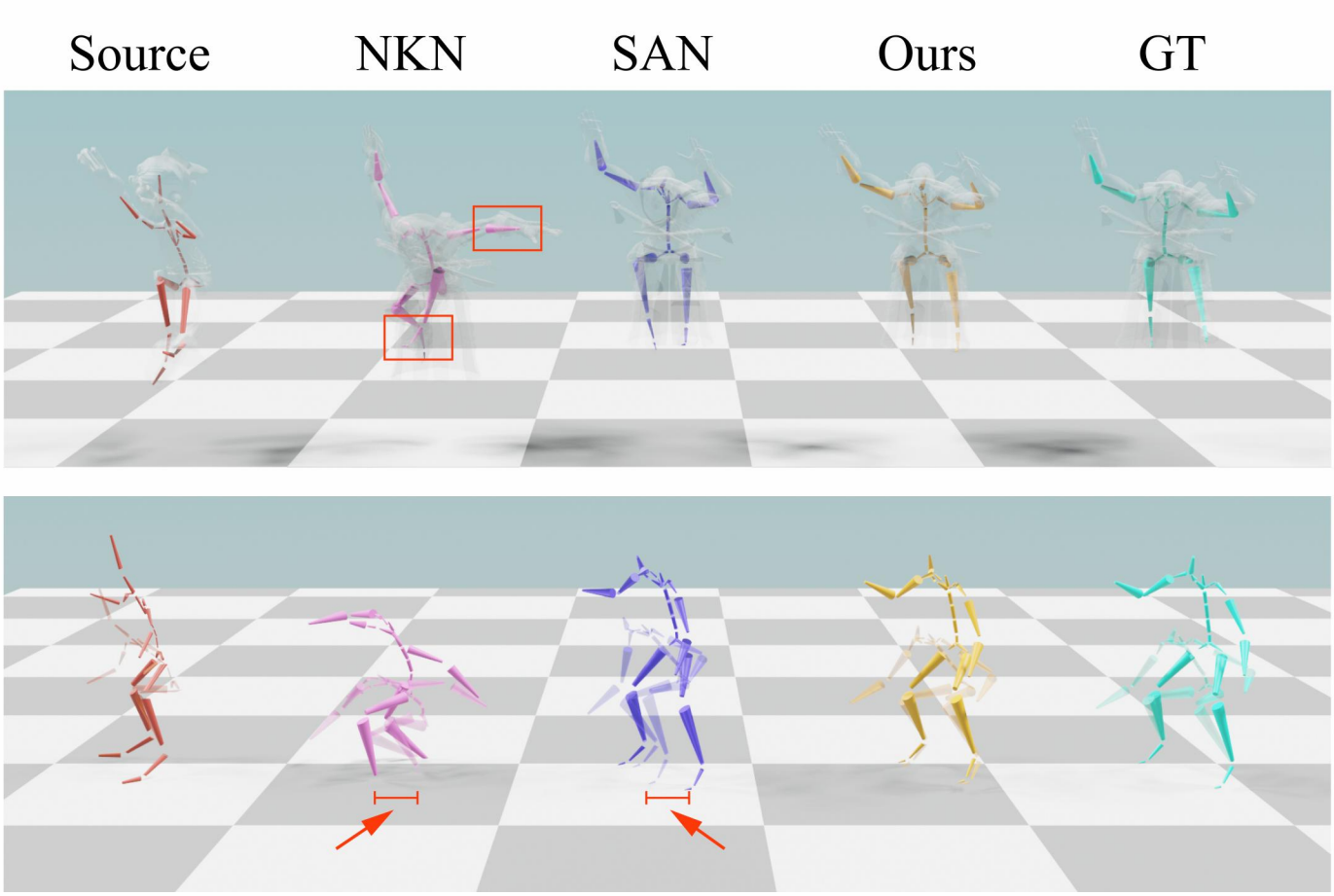

Fig. 6. A jumping-to-land motion clip. We show the retargeting results of NKN, SAN, and our method along with the corresponding ground truth. In the first row, we overlaid the transparent mesh on the skeleton. In the last row, we show the pose when standing(solid) and the pose when just touching the ground (transparent). Retargeting flaws or server foot sliding are marked with red rectangles and arrows.

share among different structure. The modified models are denoted by NKN* and PMnet*, respectively.

Columns 1-3 in Figure 5 show the comparison examples of intra-structural retargeting while the next three columns present the cross-structural retargeting results. From the comparison, we can find our method achieves more stable and accurate results. In particular, because of the unsupervised training manner, we are able to produce reasonable results when there are little flaws in the ground truth (see the middle column of the cross-structural retargeting in Figure 5)

TABLE 1
Quantitative Evaluation on Maximo Dataset. We Report the Mean Per Joint Position Error(MPJPE) over Test Clips, Normalized by the Skeleton's Height (multiplied by 10^3, similar with [11]).

	Intra-Structural	Cross-Structural
Copy rotations	8.86	N/A
NKN/NKN*	5.84	7.36
PMnet/PMnet*	4.93	6.88
SAN	2.76	2.25
Ours	**0.50**	**1.62**

6.2.1 Quantitative Evaluation

For quantitative evaluation, we use mean per joint position error(MPJPE) as a metric and compare the generated results with ground truth over I=106 test motion clips. The MPJPE formula is described as follows:

$$\mathcal{E} = \frac{1}{I|\mathcal{C}_s||\mathcal{C}_t|h_t} \sum_{i=1}^{I} \sum_{a \in \mathcal{C}_s} \sum_{b \in \mathcal{C}_t} \|FK(\mathcal{M}_{T_i}^{s \mapsto t}) - FK(\mathcal{M}_{T_i}^{t})\|^2 \tag{16}$$

Where the $\mathcal{M}_{T_i}^{s \mapsto t}$ and $\mathcal{M}_{T_i}^{t}$ are the retargeted motion from skeleton s to t and ground truth motion from the Mixamo dataset, respectively. FK is a forward kinematic function for transferring the joint rotations to global positions. i denotes the index of the test motion examples. $\mathcal{C}_s$ and $\mathcal{C}_t$ represent the source and target skeleton set. In the intra-structural retargeting case, the set $\mathcal{C}_t$ is the same as $\mathcal{C}_s$, i.e., both are the 4 testing skeletons in structure $\mathcal{A}$. In cross-structural retargeting, the source set C_s contains only 1 test skeleton in the group $\mathcal{B}$, and C_t contains the 4 skeletons in the group $\mathcal{A}$. In order to eliminate the error magnitude caused by different skeleton sizes, we divide the joint position error by target skeleton height h_t.

The quantitative results are reported in Table 1. Since the target skeleton has one-to-one joint correspondence with the source skeleton in intra-structural retargeting, we additionally compute the result of Copy rotations as a baseline, i.e., directly copying each joint rotation to the target skeleton. From the table, we can see the performance of NKN and PMnet is far behind SAN and our method since they feed

the networks with a simple concatenation of whole-body joint features, without modeling the spatial characteristics of the skeletons. Our approach outperforms all competing methods in both intra-structural and cross-structural retargeting, attributing to the expressive power of the attention mechanism and the body part strategy, which will be further discussed.

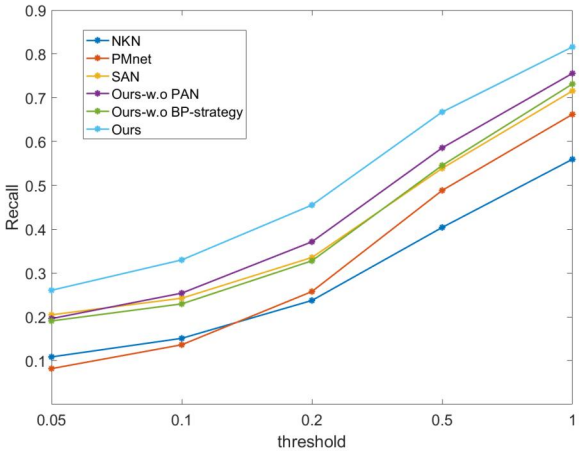

Fig. 7. The foot-contact recall of different methods. Given different foot velocity thresholds, we compute the foot-contact recall by equation 19. A higher value indicates the foot movement is closer to the ground truth.

6.2.2 Evaluation of Foot-contact Recall

In addition to the overall accuracy of motion retargeting, the movement of the end-effectors usually plays an important role, especially when considering contacts with the ground, which is crucial to our perception of human motions. However, the results produced by neural network systems often do not satisfy the contact constraints. Therefore, there is some literature like [7] utilizes numerical optimization in the hidden space to iteratively improve the solution. We believe that a good neural network prediction provides a better initial solution for the optimization process and can speed up the convergence of the algorithm, resulting in more visually plausible motions. We conduct experiments and visualizations to demonstrate that our pose-aware attention mechanism can provide more stable results, mitigating foot-sliding artifacts.

Figure 6 shows a jumping-to-land motion clip. We can see the source motion as well as the ground truth without any foot sliding after landing on the ground, but there are artifacts in each of the predicted results. The comparison in the last row of Figure 6 reveals that our method achieves more stable results with slight sliding. Although we do not specifically consider contact information during the training, the proposed method adaptively focuses attention on the foot joint within the leg body part due to the pose-aware attention mechanism, which is well visualized in Figure 8(a).

The visualization shows the contribution of each joint to the corresponding body part in different postures. We take the output of $softmax$ in equation 5 at the first attention layer and draw the attention weights of the joint embeddings $X_t \in \mathbb{R}^{(J+1)\times d}$ to body part tokens $Y \in \mathbb{R}^{N\times d}$ by heatmap. The color near the top of the color bar indicates that the joint contributes more to its corresponding body part and vice versa. It is noticed that the feature of root velocity will participate in the calculation of attention in each body part since the root velocity is important for distinguishing the motion types. We will take the maximum weight value when visualizing if a joint belongs to more than one body part and the color of the root is overlaid with the root velocity "joint".

For different motion poses, the attention weights computed by the network vary considerably. For example, in sequence (b) of Figure 8, we can observe that in the case of stationary pose (the first frame), the PAN focuses more on the root/root velocity and the weight contributed by each joint to the corresponding body part do not differ much since the pose remains stationary. This meets our intuition that the root velocity is important since it can distinguish multiple motion categories and each joint contributes similar weight at rest pose. However, it is not a static rule, attention maps for other poses in Figure 8 (b) show that our pose-aware attention mechanism has the ability to dynamically extract the spatial features between joints.

To further demonstrate our attention mechanism can mitigate foot-sliding artifacts, we use the foot-contact recall as a metric and compare our approach with other methods. Firstly, we collect all of the contact frames for each foot in ground truth motion by velocity threshold, described as follows:

$$c_{gt}(t;\epsilon) = \begin{cases} 1 & \text{if } \|P^{gt}_{t,j} - P^{gt}_{t-1,j}\|^2 < \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where the $P^{gt}_{t,j}$ represents the ground-truth position of foot joint j at time t. We collect all the contact frames to form a set $\mathcal{C}_{gt} = [t_1, t_2, ..., t_n]$ and then check the retargeted motion for foot contacts on these frames by the following equation:

$$c_{tar}(t_i;\epsilon) = \begin{cases} 1 & \text{if } \|P^{tar}_{t_i,j} - P^{tar}_{t_i-1,j}\|^2 < \epsilon \\ 0 & \text{otherwise} \end{cases} \quad t_i \in \mathcal{C}_{gt} \tag{18}$$

Finally, we are able to calculate the foot-contact recall by:

$$\mathcal{R}_\epsilon = \frac{\sum_{t_i \in \mathcal{C}_{gt}} c_{tar}(t_i;\epsilon)}{\sum_{t \in T} c_{gt}(t;\epsilon)} \tag{19}$$

We plot the line graph (see Figure 7) given different threshold ϵ and can find that our full approach outperforms all competing methods at each threshold, further demonstrating the proposed method can produce more stable results. We also conduct an ablation study on this metric, which will be discussed in section 8.

6.2.3 Motion Retargeting from Human3.6M

To demonstrate the generalization of our method, we present motion retargeting from the Human3.6 [17] characters to Mixamo's skeletons using the model trained from Mixamo data only, which means that our architecture can retarget unseen motions performed by unseen skeletons to unseen/seen skeletons. We use the ground truth 3D motions (joint rotations for our method and Copy rotation, joint positions for NKN) from the human3.6M dataset, and downsample them to 30fps.

In order to enable the encoder trained on the Mixamo structural $\mathcal{B}$ (22 joints) to receive the articulated motion from

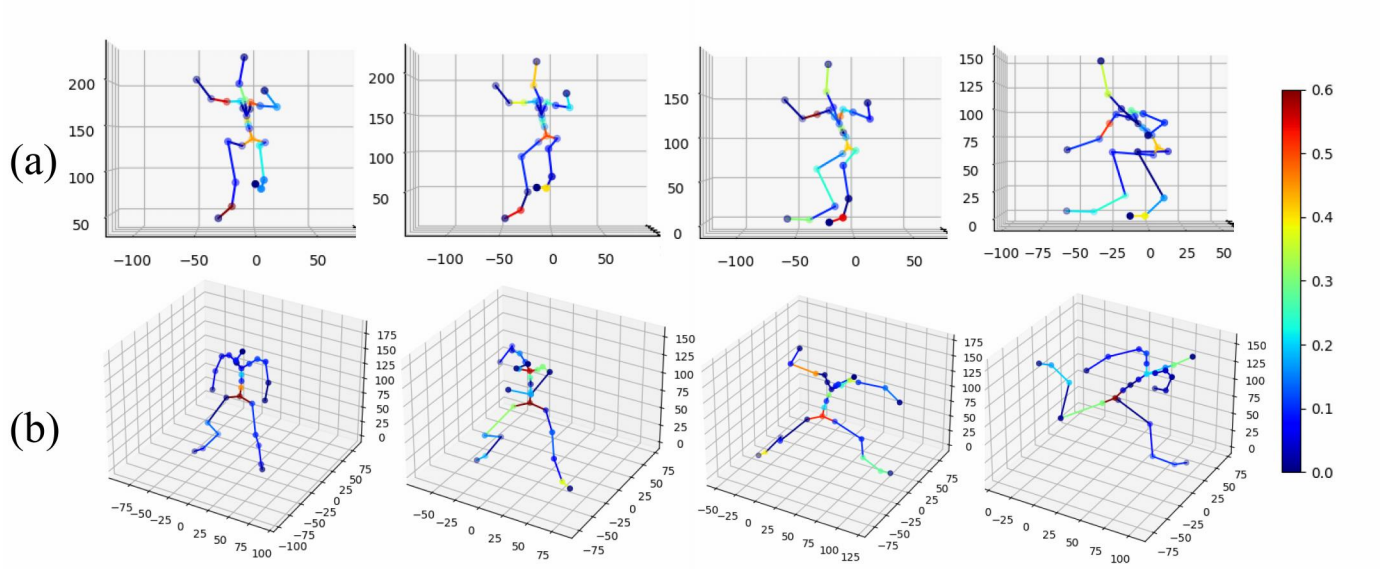

Fig. 8. Visualization of the pose-aware attention. We visualize two motion sequences (a) and (b), temporally ordered from left to right. The contribution of each joint to the corresponding body part is described by a heat map. Notably, we overlaid the attention weights of root rotation and root velocity.

Human3.6M (17 joints), we did joint mapping similar to NKN, which duplicates the joint positions in Human3.6M to corresponding Mixamo joints. Since our method receives joint rotations as input, we map the Spine rotation (H36M) into the Spine(Mixamo) and set the rotation of Spine1 (Mixamo) to zero. At the same time, the offset of Spine1 will also be set to zero, thus creating a zero-length bone in the test time. A similar operation will be applied to follow mapping pairs: LeftShoulder into LeftShoulder and LeftArm, RightShoulder into RightShoulder and RightArm, LeftFoot into LeftFoot and LeftToeBase, RightFoot into RightFoot and RightToeBase. Thus, our motion encoder trained for structural $\mathcal{B}$ can receive the motions of human3.6M dataset.

As shown in Figure 9, our method can generalize to unseen articulated motions and outperform NKN and Copy Rotation in terms of realism and plausibility. For example, we can see that directly copying the joint rotations of the human3.6m' motion to a target skeleton may produce unreasonable poses due to the structural differences between the source and target skeletons.

For NKN, since the method retargets motions in an autoregressive manner, the errors will accumulate as the motion sequence becomes longer, resulting in the generated character motion floating in the air (Row 2-4 in Figure 9). In contrast, our temporal modeling is based on 1D convolution, which makes the model more stable in long-term retargeting. Please refer to the supplementary video for more qualitative results.

To quantitatively evaluate each retargeting method, we randomly select the source motion clips (300 frames per clip) from the Human3.6M datasets and retarget them to the unseen characters of Mixamo, then allow users to score them for evaluation. Specifically, We run our user study on a total of 20 users and choose Mousey, Mremireh, and Vampire as test characters (neither our model nor NKN saw them during training). We assign 5 retargeting clips to each character, making a total of 15 questions. For each question, we randomly place the retargeting results produced by each method on the page, labeled A, B, and C. The participants are asked to grade the similarity between the retargeted and source motions on a scale of integers from 1 to 5, with "1" denoting completely dissimilar and "5" is completely similar. Table 2 shows the results of the user study. We can find that our method outperforms the NKN and Copy Rotation in terms of the scores and has a smaller standard deviation, which means that our method can achieve more accurate and robust retargeting.

TABLE 2
User Study of Retargeting from Human3.6M to Characters of Mixamo.

Copy Rotation	NKN	Ours
3.06± 1.07	3.69± 1.04	**3.78± 0.98**

7 Retargeting between Biped and Quadruped

Our retargeting framework is able to define more flexible correspondence due to the body part retargeting strategy and pose-aware attention mechanism. To demonstrate this property, we design motion retargeting between quadrupeds and bipeds (see Figure 10 left and right sides) on the more challenging datasets lafan1 [18] and quadrupeds [13].

7.1 Problem Setting

Datasets. The lafan1 [18] is a human motion dataset that includes walking, running, sitting, sprinting, fighting, etc. There are 77 unique motion sequences, and we choose the locomotion clips (i.e., BVH files starting with the keywords aiming, run, walk, and sprint) because the motion mode of quadrupeds is relatively simple (mainly locomotion). We use subject 1 for testing and the rest for training. The

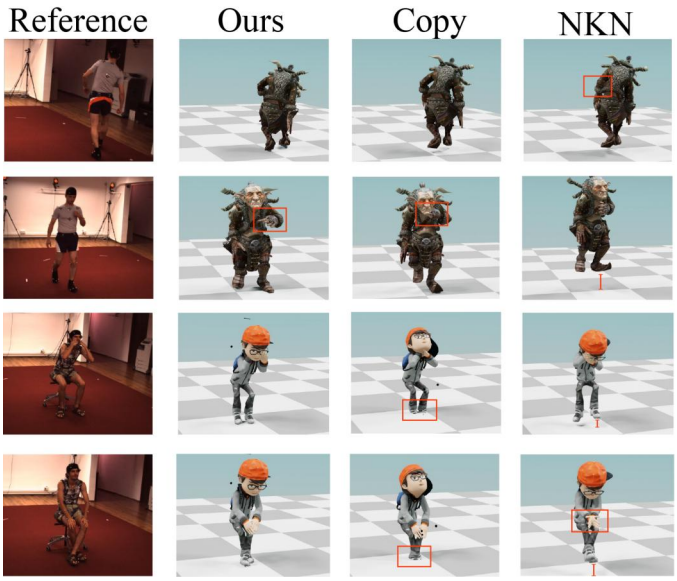

Fig. 9. Qualitative results of retargeting from Human3.6M motions to Mixamo skeletons (the Mixamo skeletons in rows 1-2 are unseen during training, while the ones in rows 3-4 are seen). We use ground truth 3D motions of the Human3.6M dataset as the source motions.

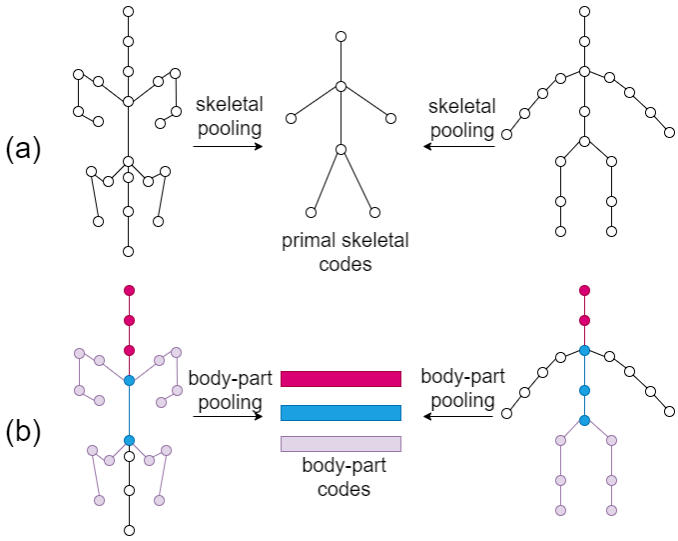

Fig. 10. Different corresponding strategies between our method and SAN. (a) SAN extracts the primal skeleton features based on the neighborhood relationship of the joints. (b) we construct the shared latent space of the source and target skeletons in terms of body parts.

quadruped dataset [13] consists of 52 unique dog motion sequences including idle, walk, run, sit, stand, and a few jumps. We downsample the frame rate to 30 fps, consistent with the lafan1 dataset. For train-test split, we split the whole 52 motion files randomly and make sure the train-test ratio is close to the bipedal dataset split. Please refer to Appendix B for detailed information.

Figure 10 shows the differences between our method and SAN [11] in terms of the corresponding strategies for bipedal and quadrupedal skeletons. The two skeletons have different structure, but the main limbs are topologically similar. The core idea of SAN is to extract primal skeletal codes through multiple pooling operations (see Figure 10 (a)). However, this strategy leads to a correspondence between the arm motions of bipeds and the foreleg motions of quadrupeds, which is semantically implausible. Instead, we define N=3 common body parts of bipeds and quadrupeds (see Figure 10 (b)) rather than a primal skeleton. Since it is difficult to find a spatial correspondence between the leg part of this two structure, we directly pack the biped's two legs to construct a correspondence with the four legs of the quadruped to ensure correctness on the semantic level. It is worth noting that we do not encode the motion of the arms of biped or the tail of quadruped, because these body parts are difficult to construct a correspondence between the structure. It means that the decoders corresponding to each structure receive only the codes of common body parts and decode the whole-body motions with these partial part features.

During the training process, our PAN calculates the attention weights for each body part, which function similarly to the gating network in MANN [13] to learn to distinguish between different actions and motion states by clustering. Meanwhile, since we use temporal convolutional layers to compress the motion, our architecture has a wider perceptual field than frame-by-frame methods and thus does not take action labels as input. The naturalness of the generated whole-body motion will be judged by the motion discriminator, and the $\mathcal{L}_{adv}$ is used to force the motion decoder to generate natural and reasonable motions. Meanwhile, we will constrain only the common body parts in $\mathcal{L}_{rec}$, $\mathcal{L}_{cyc}$, $\mathcal{L}_{kine}$. We additionally add velocity constraints to the biped-quadruped retargeting setting as follow:

$$\mathcal{L}_{vel} = \| \frac{V_s}{\|V_s\|} (\frac{\|V_s\| - v_{smin}}{v_{smax} - v_{smin}}) - \frac{V_t}{\|V_t\|} (\frac{\|V_t\| - v_{tmin}}{v_{tmax} - v_{tmin}}) \|^2 \tag{20}$$

Where the V_s and V_t represent the root joint velocity of source and target skeletons. v_*min and v_*max indicate the minimum and maximum velocity scale in the training datasets, respectively. This loss term forces the velocity to be mapped in proportion which can help us align the motion manifolds of the two morphologies and circumvent unreasonable retargeting. The coefficient of the velocity loss term is $\lambda_{vel} = 10^3$ and other coefficients are the same as in equation 14.

7.2 Experiments and Evaluation

When retargeting the quadruped motion to the bipedal character, the results produced by SAN all present a "bent" posture (see the last two rows of Figure 11) because of the unreasonable correspondence. Similarly, SAN fails on retargeting from biped to quadruped (see the first two rows of Figure 11) where only two legs are in contact with the ground and the poses are very unnatural. The qualitative results in Figure 11 show our method can produce more realistic-looking results, demonstrating our body-part corresponding strategy is more reasonable. We also show the results of PMnet*, which performs between our method and SAN since it does not consider any spatial relationship between these two skeletons but only models the motion along the temporal dimension.

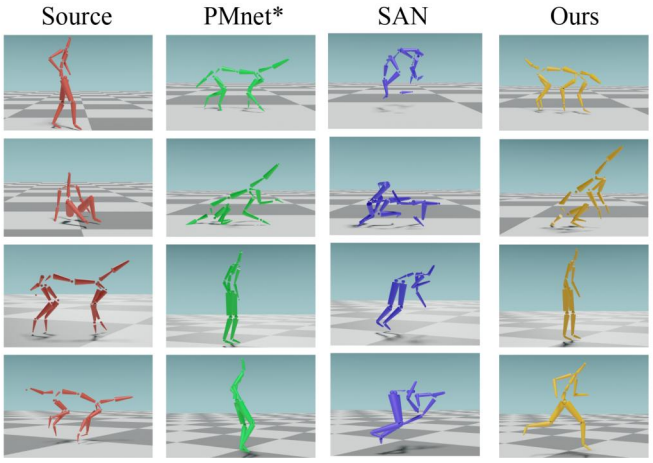

Fig. 11. Qualitative results of retargeting between bipedal and quadrupedal. The leftmost column shows the source poses and the columns on the right represent the retargeting results of modified PMnet, SAN, and our method. The results are all evaluated on the test set.

7.2.1 Quantitative Evaluation

For quantitative evaluation, since the lafan1 and quadruped datasets have no motion pairs, we evaluate the retargeting by two metrics: Fréchet Inception Distance(FID) and user study.

TABLE 3
We Evaluate the Fréchet Inception Distance(FID) on Lafan1 and Quadruped Dataset.

	Quadruped→Biped	Biped→Quadruped
NKN*	93.81	169.47
PMnet*	62.55	217.11
SAN	376.36	763.89
Ours	**51.68**	**132.76**

FID is widely used to evaluate the overall quality of generated motion [51], which depicts how similar the generated motions are to the real motions by comparison of the deep feature distributions. As we know, similar motions should have similar hidden features in the hidden layer of the network, so we pre-train an auxiliary autoencoder [48] to help us compute the FID scores (i.e. all the retargeted motions and real motions will be fed into this network for obtaining the hidden feature distribution), where the formula is shown below:

$$FID = \|m - m_w\|^2 + Trace(c + c_w - 2(cc_w)^{\frac{1}{2}}) \quad (21)$$

where the m and m_w represent the mean values of the latent features from the generated motions and real motions, respectively. while c and c_w denote the corresponding covariance matrices. The latent features we use are from the L3 hidden layer of architecture [48]. For more detailed information about the architecture, please refer to Appendix A.

The comparison results are shown in Table 3 and a smaller score indicates better performance. We compare our method with vanilla SAN [11] as well as NKN* and PMnet* (see Sec 6.2 for their definitions). We denote the evaluation item as $Biped \rightarrow Quadruped$ since we retarget the bipedal motions to the quadruped and compute the distribution with the real quadrupedal motions. $Quadruped \rightarrow Biped$ can be calculated in the opposite direction. To make the calculation of the metric insensitive to the root velocity distribution, we uniformly sample 1200 motion clips (64 frames each) in each test set based on the root velocity distribution to represent the real motion distribution. The results shown in Table 3 illustrate that we outperform other competing methods thanks to our advanced spatial modeling and body-part corresponding strategy. SAN fails on this task because the convolution operation based on the skeleton neighborhood leads to an unreasonable semantic correspondence between these two skeletons. NKN* and PMnet* do not consider any spatial relationship, resulting in inferior performance to our method.

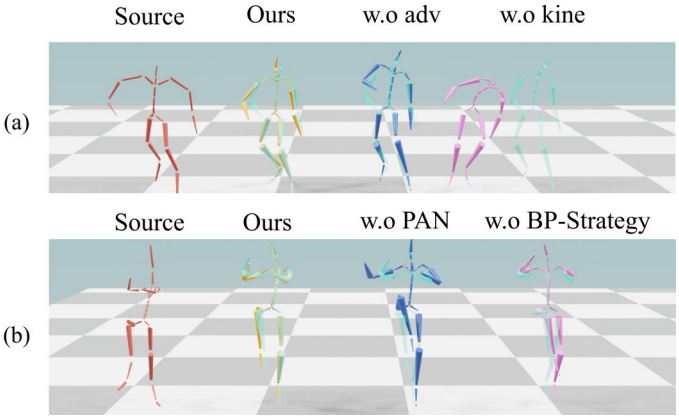

Fig. 12. Qualitative results of ablation study. We remove some modules from the architecture to compare with our full method. The outputs are overlaid with the ground truth(transparent cyan skeleton).

7.2.2 User Study

FID is capable of depicting the naturalness of the retargeted motions, but the similarity of retargeted motion to the source is difficult to quantify in the absence of ground truth. Therefore, we choose to judge the similarity by a user study. For each direction of retargeting($Quadruped \rightarrow Biped, Biped \rightarrow Quadruped$) we choose three types of action: $idle, move$, and sit. The participants are asked to grade the similarity between the retargeted and source motionslike Sec 6.2.3 We run this user study on a total of 20 users and 12 questions, each is a random sample from the datasets. The action types are labeled automatically, similar to [34], i.e. we determine whether the action type is $move$ or $idle$ by the root velocity magnitude. As for the sit label, we detect whether the hip/tail joint touches the ground during the motion. Table 4 show the mean and standard deviation of the performance scores of each method on different actions. The scores show that we outperform other competing methods on most action types, which demonstrates our method also achieves more visually plausible retargeting from a subjective perspective.

8 Ablation Study

We conduct an ablation study to further demonstrate the contribution of each component in our architecture and help us understand the design of our framework. We provide

TABLE 4
Mean Subjective Ratings with Standard Deviation. We Ask the Users to Grade the Similarity Between the Retargeted and Source Motions.

	Biped→ Quadruped			Quadruped→ Biped		
	idle	move	sit	idle	move	sit
NKN*	1.94±0.75	2.94 ±0.56	1.53±0.51	2.88±0.99	2.18±0.64	2.06±1.25
PMnet*	2.35±0.70	3.06 ± 0.75	1.82±0.73	**3.82 ± 0.88**	3.89±0.86	2.65±1.37
SAN	1.59±1.18	1.41±1.00	2.12±1.11	1.29±0.47	1.35±0.61	2.24±1.20
Ours	**4.29±0.69**	**4.53 ± 0.62**	**4.53±0.51**	3.76±1.03	**4.06±0.75**	**3.76±1.03**

several examples under different self-comparison settings in Figure 12 and the quantitative results are detailed in Table 5. The evaluation also can be found in the supplementary video.

TABLE 5
Quantitative Results of Ablation Study. We Remove Some Components of Our Architecture or Loss Terms from Our Full Method to Evaluate the Retargeting Performance on the Mixamo Dataset.

	Intra-Structural	Cross-Structural
ours	**0.50**	**1.62**
w.o BP-strategy	0.58	2.29
w.o PAN	1.19	2.28
w.o $\mathcal{L}_{kine}$	1.11	48.08
w.o $\mathcal{L}_{adv}$	1.16	3.23

8.1 Effect of Pose-Aware Attention

To demonstrate the effect of the pose-aware attention mechanism in our architecture, we compare our full approach with the model without the attention mechanism. Specifically, we remove the attention operations described by equation 4 and 5, replace them with a simple MLP layer, and keep the rest of our architecture fixed. When the pose-aware attention is removed, the model can not dynamically process the spatial features. The results shown in Figure 12 (b) and Table 5 both demonstrate that the pose-aware attention mechanism can improve the accuracy of the retargeting. In addition, the recall curves of foot contact in Figure 7 also show that the proposed attention mechanism is beneficial to improve the stability of the retargeting.

8.2 Effect of Body Part Strategy

To illustrate the effectiveness of the body part strategy (BP-strategy), we replace the "body part tokens" with "whole body token", i.e., we remove the mask matrix U from equation 5 so that all joints are associated with the only "whole body token". In addition, the convolution kernels in equation 6 are not body-part related and will degrade to the vanilla 1D convolution kernels. The results in Table 5 and Figure 6 both show some decrease in performance for the model without the BP-strategy. We believe that the body parts provide a geometric prior for the neural networks and using body parts as the retargeting units is beneficial for network convergence when an unsupervised training manner is used.

8.3 Effect of Kinematic Loss

We investigate the effectiveness of kinematic loss by removing the loss term $\mathcal{L}_{kine}$ during training. The kinematic loss can supervise the training in Cartesian space which is more crucial to our perception. From Table 5, we observe that our full model performs better than the model without this loss, especially in the Cross-Structural retargeting since the Cross-Structural retargeting is harder than the Intra-Structural retargeting. We believe that the rotation is more sensitive than the joint position since the rotation error of the father joint will be amplified in the coordinate positions of the child's joint by matrix chain multiplication. Therefore, we must introduce the kinematic loss when training the networks.

8.4 Effect of Adversarial Loss

To evaluate the contribution of adversarial loss, we discard the loss term $\mathcal{L}_{adv}$ and retrain our networks. The results in Table 5 show that our full method outperforms the model without adversarial loss term in both retargeting scenarios. We believe that $\mathcal{L}_{adv}$ is very important in unsupervised learning since it ensures that the motion generated by the networks falls in the motion manifold of the corresponding structure. The example shown in Figure 12 (a) also illustrates the importance of the adversarial loss term in our retargeting framework.

9 Discussion and Future Work

We propose a novel motion retargeting framework that uses body parts as the basic retargeting units, which together with our pose-aware attention mechanism can dynamically extract the spatial features of the motion. Our architecture can learn the shared motion space of body parts from unstructured motion capture data, which can easily allow retargeting among skeletons with different structure. Due to our dynamical spatial modeling, our method allows for more accurate and flexible retargeting compared to other approaches. In particular, we significantly improve the performance of motion retargeting between quadrupeds and bipeds.

The main limitation of our work is the dependence on motion statistics, i.e., we still require an amount of balanced motion capture data for each structure. Both datasets need to contain motions with various facing directions, velocity directions, velocity intervals, and the same action types as much as possible. We show in the supplemental video a failure case when retargeting the backward walking of a biped to a quadruped, where we find that the quadruped moves unnaturally and without gaits. This is because, although the biped training set includes backward walking, the quadruped training set lacks such movement, making it difficult to correspond when learning. When we test the retargeting of bipedal artistic movements, such as hopping

(unseen action) to the quadruped. We find that our method can only use some corresponding laws learned from locomotion to generate the retargeted motion of the quadruped, which cannot guarantee naturalness and rhythm. This motivates us to introduce more control signals to achieve anthropomorphic retargeting in the future.

In some scenarios, we do not have access to the capture data of the target skeleton. For this problem, we can consider introducing one-shot or zero-shot learning into motion retargeting in the future, which can further expand the application of automatic motion retargeting. Another drawback of our work is that we currently rely on self-supervised and adversarial learning to align the motion manifolds of different structure. This correspondence may result in some output being semantically inconsistent with the source motion and lacking physical realism, e.g., distinguishing between sitting on the floor and sitting on a chair when retargeting the biped motion to the quadruped. One potential direction is to combine our spatial modeling with the Dynamic Motion Reassembly from Virtual Chimeras [29] to achieve physically realistic motion retargeting.

From another perspective, our approach is in fact to learn the motion manifolds of various body parts. Therefore, we are also able to implement motion editing or user interaction in the latent space like [7], [48] to achieve body part level control in the future. Recently, DeepPhase [52] starts to learn motion manifold from a temporal alignment perspective, which can extract periodic features of the whole body. This alignment is very helpful for retargeting between different organisms because it is difficult for us to obtain pairwise motion data. We are able to introduce periodicity constraints into the loss function to make the generated motions more reasonable.

ACKNOWLEDGMENTS

This work was supported by National Key Research and Development Program of China (NO. 2022YFB3303202).

REFERENCES

[1] M. Loper, N. Mahmood, and M. J. Black, "Mosh: Motion and shape capture from sparse markers," ACM Transactions on Graphics (ToG), vol. 33, no. 6, pp. 1–13, 2014.

[2] S. Kim, M. Sorokin, J. Lee, and S. Ha, "Human motion control of quadrupedal robots using deep reinforcement learning," arXiv preprint arXiv:2204.13336, 2022.

[3] D. Rempe, L. J. Guibas, A. Hertzmann, B. Russell, R. Villegas, and J. Yang, "Contact and human dynamics from monocular video," in European conference on computer vision. Springer, 2020, pp. 71–87.

[4] Y. Ye, L. Liu, L. Hu, and S. Xia, "Neural3points: Learning to generate physically realistic full-body motion for virtual reality users," arXiv preprint arXiv:2209.05753, 2022.

[5] R. Villegas, J. Yang, D. Ceylan, and H. Lee, "Neural kinematic networks for unsupervised motion retargetting," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8639–8648.

[6] J. Lim, H. J. Chang, and J. Y. Choi, "Pmnet: Learning of disentangled pose and movement for unsupervised motion retargeting." in BMVC, 2019, p. 136.

[7] R. Villegas, D. Ceylan, A. Hertzmann, J. Yang, and J. Saito, "Contact-aware retargeting of skinned motion," arXiv preprint arXiv:2109.07431, 2021.

[8] M. Gleicher, "Retargetting motion to new characters," in Proceedings of the 25th annual conference on Computer graphics and interactive techniques, 1998, pp. 33–42.

[9] C. Kwang-Jin and K. Hyeong-Seok, "On-line motion retargetting," The Journal of Visualization and Computer Animation, vol. 11, pp. 223–235, 2000.

[10] J. Lee and S. Y. Shin, "A hierarchical approach to interactive motion editing for human-like figures," in Proceedings of the 26th annual conference on Computer graphics and interactive techniques, 1999, pp. 39–48.

[11] K. Aberman, P. Li, D. Lischinski, O. Sorkine-Hornung, D. Cohen-Or, and B. Chen, "Skeleton-aware networks for deep motion retargeting," ACM Transactions on Graphics (TOG), vol. 39, no. 4, pp. 62–1, 2020.

[12] D. Holden, T. Komura, and J. Saito, "Phase-functioned neural networks for character control," ACM Transactions on Graphics (TOG), vol. 36, no. 4, pp. 1–13, 2017.

[13] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control," ACM Transactions on Graphics (TOG), vol. 37, no. 4, pp. 1–11, 2018.

[14] S. Starke, H. Zhang, T. Komura, and J. Saito, "Neural state machine for character-scene interactions." ACM Trans. Graph., vol. 38, no. 6, pp. 209–1, 2019.

[15] C. Zhong, L. Hu, Z. Zhang, Y. Ye, and S. Xia, "Spatio-temporal gating-adjacency gcn for human motion prediction," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 6447–6456.

[16] A. S. Inc, "Adobe's mixamo." https://www.mixamo.com, 2021, accessed: 2021-04-02.

[17] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," IEEE transactions on pattern analysis and machine intelligence, vol. 36, no. 7, pp. 1325–1339, 2013.

[18] F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal, "Robust motion in-betweening," vol. 39, no. 4, 2020.

[19] C. Hecker, B. Raabe, R. W. Enslow, J. DeWeese, J. Maynard, and K. van Prooijen, "Real-time motion retargeting to highly varied user-created morphologies," ACM Transactions on Graphics (TOG), vol. 27, no. 3, pp. 1–11, 2008.

[20] Z. Popović and A. Witkin, "Physically based motion transformation," in Proceedings of the 26th annual conference on Computer graphics and interactive techniques, 1999, pp. 11–20.

[21] S. Tak and H.-S. Ko, "A physically-based motion retargeting filter," ACM Transactions on Graphics (TOG), vol. 24, no. 1, pp. 98–117, 2005.

[22] K. Yamane, Y. Ariki, and J. Hodgins, "Animating non-humanoid characters with human motion data," in Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 2010, pp. 169–178.

[23] Y. Seol, C. O'Sullivan, and J. Lee, "Creature features: online motion puppetry for non-human characters," in Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 2013, pp. 213–221.

[24] S. Uk Kim, H. Jang, and J. Kim, "A variational u-net for motion retargeting," Computer Animation and Virtual Worlds, vol. 31, no. 4-5, p. e1947, 2020.

[25] W.-S. Jang, W.-K. Lee, I.-K. Lee, and J. Lee, "Enriching a motion database by analogous combination of partial human motions," The Visual Computer, vol. 24, no. 4, pp. 271–280, 2008.

[26] R. Heck, L. Kovar, and M. Gleicher, "Splicing upper-body actions with locomotion," in Computer Graphics Forum, vol. 25, no. 3. Wiley Online Library, 2006, pp. 459–466.

[27] W. Ma, S. Xia, J. K. Hodgins, X. Yang, C. Li, and Z. Wang, "Modeling style and variation in human motion," in Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 2010, pp. 21–30.

[28] D.-K. Jang, S. Park, and S.-H. Lee, "Motion puzzle: Arbitrary motion style transfer by body part," arXiv preprint arXiv:2202.05274, 2022.

[29] S. Lee, J. Lee, and J. Lee, "Learning virtual chimeras by dynamic motion reassembly," ACM Transactions on Graphics (TOG), vol. 41, no. 6, pp. 1–13, 2022.

[30] M. Abdul-Massih, I. Yoo, and B. Benes, "Motion style retargeting to characters with different morphologies," in Computer Graphics Forum, vol. 36, no. 6. Wiley Online Library, 2017, pp. 86–99.

[31] Z. Liao, J. Yang, J. Saito, G. Pons-Moll, and Y. Zhou, "Skeleton-free pose transfer for stylized 3d characters," arXiv preprint arXiv:2208.00790, 2022.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in neural information processing systems, 2017, pp. 5998–6008.

[33] S. Xia, C. Wang, J. Chai, and J. Hodgins, "Realtime style transfer for unlabeled heterogeneous human motion," ACM Transactions on Graphics (TOG), vol. 34, no. 4, pp. 1–10, 2015.

[34] S. Starke, Y. Zhao, T. Komura, and K. Zaman, "Local motion phases for learning multi-contact character movements," ACM Transactions on Graphics, vol. 39, no. 4, 2020.

[35] S. Starke, Y. Zhao, F. Zinno, and T. Komura, "Neural animation layering for synthesizing martial arts movements," ACM Transactions on Graphics (TOG), vol. 40, no. 4, pp. 1–16, 2021.

[36] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Two-stream adaptive graph convolutional networks for skeleton-based action recognition," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 12 026–12 035.

[37] K. Lin, L. Wang, and Z. Liu, "End-to-end human pose and mesh reconstruction with transformers," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 1954–1963.

[38] C. Zheng, S. Zhu, M. Mendieta, T. Yang, C. Chen, and Z. Ding, "3d human pose estimation with spatial and temporal transformers," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 11 656–11 665.

[39] J. Zhang, Z. Tu, J. Yang, Y. Chen, and J. Yuan, "Mixste: Seq2seq mixed spatio-temporal encoder for 3d human pose estimation in video," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 13 232–13 242.

[40] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Decoupled spatial-temporal attention network for skeleton-based action-gesture recognition," in Proceedings of the Asian Conference on Computer Vision, 2020.

[41] M. Petrovich, M. J. Black, and G. Varol, "Action-conditioned 3d human motion synthesis with transformer vae," arXiv preprint arXiv:2104.05670, 2021.

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

[43] I. Baran, D. Vlasic, E. Grinspun, and J. Popović, "Semantic deformation transfer," in ACM SIGGRAPH 2009 papers, 2009, pp. 1–6.

[44] U. Celikcan, I. O. Yaz, and T. Capin, "Example-based retargeting of human motion to arbitrary mesh models," in Computer Graphics Forum, vol. 34, no. 1. Wiley Online Library, 2015, pp. 216–227.

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

[46] J. Ren, M. Chai, O. J. Woodford, K. Olszewski, and S. Tulyakov, "Flow guided transformable bottleneck networks for motion retargeting," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 10 795–10 805.

[47] K. Aberman, R. Wu, D. Lischinski, B. Chen, and D. Cohen-Or, "Learning character-agnostic motion for motion retargeting in 2d," arXiv preprint arXiv:1905.01680, 2019.

[48] D. Holden, J. Saito, T. Komura, and T. Joyce, "Learning motion manifolds with convolutional autoencoders," in SIGGRAPH Asia 2015 technical briefs, 2015, pp. 1–4.

[49] D. Holden, J. Saito, and T. Komura, "A deep learning framework for character motion synthesis and editing," ACM Transactions on Graphics (TOG), vol. 35, no. 4, pp. 1–11, 2016.

[50] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," Computer Science, 2014.

[51] C. Guo, X. Zuo, S. Wang, S. Zou, and L. Cheng, "Action2motion: Conditioned generation of 3d human motions," 2020.

[52] S. Starke, I. Mason, and T. Komura, "Deepphase: Periodic autoencoders for learning motion phase manifolds," ACM Transactions on Graphics (TOG), vol. 41, no. 4, pp. 1–13, 2022.

Lei Hu Lei Hu received a B.Sc. degree in mathematics and applied mathematics from Southwest Jiaotong University (SWJTU), China, in 2019. He is currently pursuing a Ph.D. degree in computer science at the Institute of Computing Technology, Chinese Academy of Sciences, supervised by Prof. Shihong Xia.

Zihao Zhang Zihao Zhang (Member, IEEE) received a B.Sc. degree in mathematics from Sichuan University, Sichuan, China, in 2016, and a Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2022. He is currently a special research assistant with the Intelligent Processor Research Center at the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include human motion modeling and image restoration.

Chongyang Zhong Chongyang Zhong received a B.Sc. degree in automation from Tsinghua University (THU), China, in 2017. He is currently pursuing a Ph.D. degree in computer science at the Institute of Computing Technology, Chinese Academy of Sciences, supervised by Prof. Shihong Xia.

Boyuan Jiang Boyuan Jiang received a B.Sc. degree in mathematics and applied mathematics from China University of Geosciences, Beijing(CUGB), China, in 2020. He is currently pursuing an M.E. degree in computer science at the Institute of Computing Technology, Chinese Academy of Sciences, supervised by Prof. Shihong Xia.

Shihong Xia Shihong Xia received a Ph.D. degree in computer science from the University of Chinese Academy of Sciences. He is currently a professor at the Institute of Computing Technology, Chinese Academy of Sciences (ICT, CAS), and the director of the human motion modeling laboratory. His primary research is in the area of computer graphics, virtual reality, and artificial intelligence.