Learning Skeletal Articulations with Neural Blend Shapes

PEIZHUO LI, CFCS, Peking University & AICFVE, Beijing Film Academy
KFIR ABERMAN, Google Research
RANA HANOCKA, Tel-Aviv University
LIBIN LIU, CFCS, Peking University
OLGA SORKINE-HORNUNG, ETH Zurich & AICFVE, Beijing Film Academy
BAOQUAN CHEN*, CFCS, Peking University & AICFVE, Beijing Film Academy

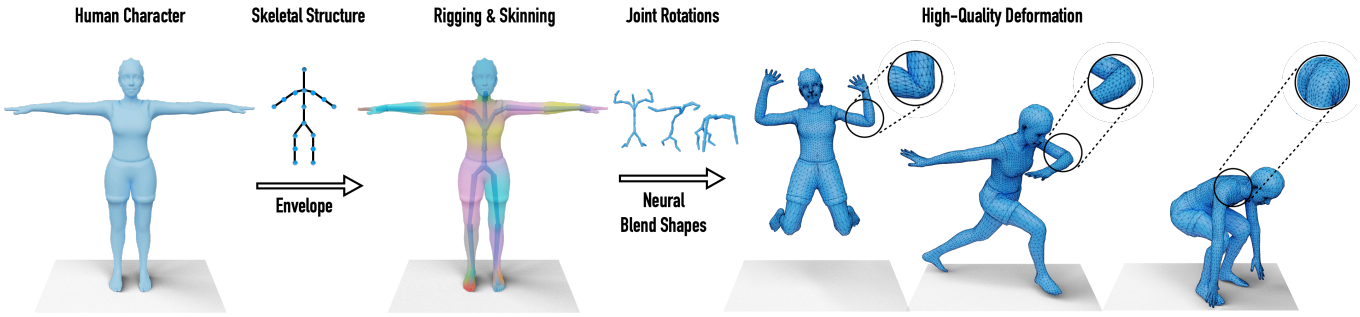


Fig. 1. Our neural skeletal articulation network learns to rig and skin an input character with arbitrary connectivity, and generates neural blend shapes. Our framework produces pose-dependent displacements that result in high quality deformations, especially in the joint regions.

Animating a newly designed character using motion capture (mocap) data is a long standing problem in computer animation. A key consideration is the skeletal structure that should correspond to the available mocap data, and the shape deformation in the joint regions, which often requires a tailored, pose-specific refinement. In this work, we develop a neural technique for articulating 3D characters using enveloping with a pre-defined skeletal structure which produces high quality pose dependent deformations. Our framework learns to rig and skin characters with the same articulation structure (*e.g.,* bipeds or quadrupeds), and builds the desired skeleton hierarchy into the network architecture. *Furthermore*, we propose *neural blend shapes* – a set of corrective pose-dependent shapes which improve the deformation quality in the joint regions in order to address the notorious artifacts resulting from standard rigging and skinning. Our system estimates neural blend shapes for input meshes with arbitrary connectivity, as well as weighting coefficients which are conditioned on the input joint rotations. Unlike recent deep learning techniques which supervise the network with ground-truth rigging and skinning parameters, our approach does not assume that the training data has a specific underlying deformation model. Instead, during training, the network observes deformed shapes and learns to infer the corresponding rig, skin and blend shapes using *indirect supervision*. During inference, we demonstrate that our network generalizes to unseen characters with arbitrary mesh connectivity, including unrigged characters built by 3D artists. Conforming to standard skeletal animation models enables direct plug-and-play in standard animation software, as well as game engines.

ACM Reference Format:
Peizhuo Li, Kfir Aberman, Rana Hanocka, Libin Liu, Olga Sorkine-Hornung, and Baoquan Chen. 2021. Learning Skeletal Articulations with Neural Blend Shapes. *ACM Trans. Graph.* 40, 4, Article 1 (August 2021), 14 pages. https://doi.org/10.1145/3450626.3459852

1 INTRODUCTION

Animating a 3D character from motion capture (mocap) data is a complex, arduous skill that animators spend years attempting to master [O'Hailey 2018]. Given a new character, a typical scenario involves manually creating a suitable character *rig* that is bound to the input geometry via *skinning* weights. Careful consideration should be given to the skeleton hierarchy of the designed character rig so as to correspond to the skeletal structure used in mocap data. Additionally, different poses, *e.g.,* bending the elbow vs. extending it, require a tailored *pose-specific* corrective deformation, especially in the joint regions.

A character can be articulated by applying joint rotations (obtained *e.g.* from motion capture data) to a skeletal deformation model, typically linear blend skinning (LBS) [Magnenat-Thalmann et al. 1988] or dual quaternion skinning (DQS) [Kavan et al. 2007]. Their simple and efficient formulation makes these methods a popular choice for animation software, games, and recently even deep

*corresponding author

Authors' addresses: Peizhuo Li, peizhuo@pku.edu.cn; Kfir Aberman, kfiraberman@gmail.com; Rana Hanocka, ranahanocka@gmail.com; Libin Liu, libin.liu@pku.edu.cn; Olga Sorkine-Hornung, sorkine@inf.ethz.ch; Baoquan Chen, baoquan@pku.edu.cn.

0730-0301/2021/8-ART1 $15.00
https://doi.org/10.1145/3450626.3459852

learning [Xu et al. 2020]. However, such skinning and rigging deformation models are an oversimplification of how humans and animals move, resulting in notorious artifacts (*e.g.,* elbow collapse). A high quality deformation can be obtained using blend shapes [Lewis et al. 2000; Weber et al. 2007] where, for example, the blending coefficients are *conditioned* on joint rotations [Loper et al. 2015] to provide fine-grained control in delicate regions. We are inspired by SMPL [Loper et al. 2015] (recently extended to STAR [Osman et al. 2020]), which achieves high quality deformations using predicted blend shapes for characters with a fixed mesh connectivity. However, a limitation of SMPL in the context of rigging and skinning is that in practice different characters almost always contain different mesh connectivities.

In this work, we present a neural technique for articulating 3D characters that learns rigging, skinning, and blend shapes for inputs with *arbitrary* mesh connectivity. We purposefully design our architecture to use a *prescribed* skeleton structure, enabling generating practical skeleton rigs that are compatible with mocap data and simplifying the *mocap-to-deformation* process. Our system animates characters using enveloping with the desired skeletal structure and pose-specific corrective deformations. It predicts skinning weights for the input mesh and computes a set of corrective, pose-dependent shapes that improve the deformation quality in joint regions, coined *neural blend shapes*.

During training, the network observes deformed shapes and learns to infer the corresponding rig, skin and blend shapes using *indirect supervision*, bypassing the need to provide supervised envelope or blend-shape deformation parameters. Unlike recent deep learning techniques that supervise the network with ground-truth rigging and skinning parameters [Liu et al. 2019; Xu et al. 2020], our approach does not assume that the training data has a specific underlying deformation model. Our indirect supervision enables learning an arbitrary number of blend shapes, which we use to generate fewer blend shape bases than the original training data. Our network also learns to *mask* the generated blend shapes, creating compact and localized bases without the need for such masks for supervision.

We learn deep features directly on the input mesh connectivity using MeshCNN [Hanocka et al. 2019], and predict a mesh attention map to modulate the deep vertex features, giving rise to neural skinning weights. In addition, we learn deep features on the target skeleton hierarchy using a skeleton-aware network [Aberman et al. 2020] to estimate the rigging parameters. We further enrich the training data by performing mesh connectivity augmentations (*e.g.,* edge collapse, flip and split), which enables us to generalize to unseen mesh connectivity during inference.

This work is the first deep learning based method for automatic enveloping combined with pose-dependent blend shapes for a skin mesh with arbitrary connectivity. Our neural blend shapes can generate high quality mesh deformations and avoid the notorious artifacts of LBS-based systems [Baran and Popović 2007; Liu et al. 2019; Xu et al. 2020]. Our framework conforms to popular skeletal animation models, enabling plug-and-play of our output in standard animation software and game engines. We demonstrate the performance of our method on a variety of examples, including unseen, unrigged characters built by 3D artists.

2 RELATED WORK

2.1 Mesh deformation of articulated shapes

Deforming a mesh based on a skeletal deformation is a fundamental problem in computer graphics. One of the earliest and most widely used skinning techniques is linear blend skinning (LBS) [Magnenat-Thalmann et al. 1988]. This method computes the deformation of the mesh as a weighted sum of the character's bone transformations, where the skinning weights can be computed manually or automatically. The simple formula of LBS allows fast evaluation and can be easily parallelized to fully utilize modern GPUs' high performance, making the method an essential technique for real-time applications, such as games. Despite this success, linear blend skinning suffers from known artifacts, such as *elbow collapse* and *candy wrapper*. Existing works develop improved techniques to overcome such shortcomings, such as dual quaternion skinning [Hejl 2004; Kavan et al. 2007; Le and Hodgins 2016], spherical based skinning [Kavan and Žára 2005], multi-linear techniques [Wang and Phillips 2002; Merry et al. 2006], and approaches with additional deformers or cages [Ju et al. 2005; Joshi et al. 2007; Jacobson et al. 2011; Kavan and Sorkine 2012; Mukai and Kuriyama 2016; Yifan et al. 2020].

While LBS and similar approaches offer efficient run-time performance, they require additional corrective deformations to express details, such as wrinkles and muscle bulges, and alleviate deformation artifacts. Example-based methods provide users with more control of the deformation behavior, see *e.g.* [Sloan et al. 2001; Lewis et al. 2000; Weber et al. 2007; Fröhlich and Botsch 2011; Loper et al. 2015]. Many such methods encode example deformations into a set of deformation bases (*i.e.* blend shapes) and compute the mesh deformation as a linear combination of these blend shapes and blending coefficients. The blend shapes can be represented as vertex displacements [Lewis et al. 2000; Weber et al. 2007], statistical models [Kry et al. 2002; Loper et al. 2015], or a compact sparse format [Seo et al. 2011]. In the standard animation pipeline, animators manually adjust blending coefficients to deform the mesh, whereas an RBF-based blend space [Lewis et al. 2000; Sloan et al. 2001] can facilitate this tedious process. Bone transformations and other high-level controls can be converted into blending coefficients by solving constrained geometric optimization problems [Sumner et al. 2005; Weber et al. 2007; Fröhlich and Botsch 2011]. Blending coefficients can also be updated in a dynamic simulation to generate secondary effects [Hahn et al. 2012; Zhang et al. 2020].

Recent research explores neural network based approaches to improve traditional skinning methods. Bailey et al. [2018] approximate the deformation of a complex production rig using neural networks, which reduces the execution cost and allows film-quality deformation in real-time applications. Later research extends this idea to more complicated facial rigs [Bailey et al. 2020; Song et al. 2020]. Neural networks can also be trained to convert high-level user control into rig parameters [Bailey et al. 2020] to enable user-friendly editing of mesh deformation. Li et al. [2020] train a graph neural network (GNN) to apply corrective displacements to linear deformations and create nonlinear effects. While the model generates high quality mesh deformation, repeatedly evaluating a deep neural network at runtime can be expensive. Our method is also a neural skinning technique. We employ an envelope skin deformation

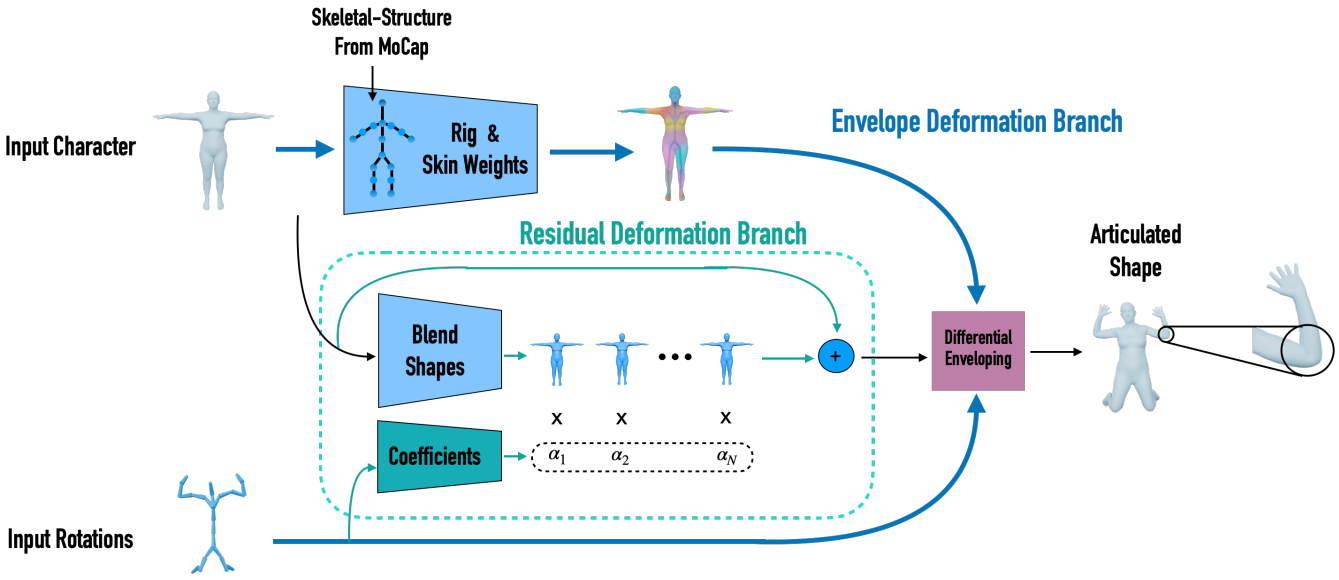


Fig. 2. Method overview. Starting with a character model in T-pose and the joint rotations on the desired skeleton hierarchy, our envelope branch predicts the corresponding skinning and rigging parameters and deforms the input character using a differential enveloping. In parallel, the residual deformation branch uses the input mesh to predict N blend shapes and uses the joint rotations to predict the corresponding blending coefficients α_i. The blend shapes add a residual deformation that is conditioned on the input joint rotations, resulting in high-quality pose-dependent deformations. Note that our system handles input characters with arbitrary mesh connectivity and does not use joint positions as input.

model and train a novel neural representation of blend shapes. Once generated for an input mesh, our lightweight neural blend shapes can be evaluated efficiently at runtime to achieve high quality mesh deformation.

2.2 Automatic skinning and rigging

Automatically rigging a skin mesh and creating ready-to-animate models has been a long-standing challenge in computer graphics. In the pioneering work *Pinocchio*, Baran and Popović [2007] propose a template-based method that automatically fits a user-provided skeleton to a target mesh and creates an animation-ready rig. However, this method does not generate blend shapes, and the resulting deformation can contain notorious LBS artifacts. Miller et al. [2010] later demonstrate a system that automatically rigs an input mesh by combining parts from a number of templates. Skeleton extraction can be also achieved by analyzing the geometric features of the input mesh [Au et al. 2008; Cao et al. 2010; Bharaj et al. 2012]. These methods are applicable to a large range of shapes, but often lack precise control of the topology of the output skeleton.

Automatic computation of skinning weights is a complementary problem to skeleton extraction. Previous research exploits methods based on projections and heat diffusion [Baran and Popović 2007; Wareham and Lasenby 2008], bounded biharmonic energy [Jacobson et al. 2011], geodesic voxel binding [Dionne and de Lasa 2013], and physics-inspired approaches [Kavan and Sorkine 2012]. Data-driven methods can utilize spatial coherence between examples and compute high quality skeleton and skinning weights by fitting the examples to skinning models like LBS [James and Twigg 2005; Schaefer and Yuksel 2007; De Aguiar et al. 2008; Hasler et al. 2010; Le and Deng 2014]. They can further extract blend shapes from the examples [Lewis et al. 2000; Kry et al. 2002; Loper et al. 2015]. However, these methods require a set of example deformations of the same mesh as input, which can be difficult to obtain in practice, and they do not necessarily produce art-directable skeleton structure.

Several recent works take advantage of the power of deep neural networks to achieve high quality rigging. *NeuroSkinning* [Liu et al. 2019] is a GNN-based network designed to predict skinning weights. It is trained with supervised learning on a skinning dataset created by professional artists. While this model can predict high quality skinning weights, it requires creating a suitable rigged skeleton with carefully placed joints. *RigNet* [Xu et al. 2020] is another GNN-based model for automatic rigging and skinning. The network is trained to apply mesh contraction to the input mesh and utilizes an attention-based clustering module to detect joints. The method allows users to guide the skeleton extraction with a tunable level-of-detail parameter, but there is no direct control over the topology of the generated skeleton. Moreover, the output of this system is an LBS-based rig without blend shapes that suffers from the standard LBS artifacts. In contrast to these techniques, our method automatically computes rigging, skinning, and blend shapes for an input mesh. Notably, our method does not assume that the training data has a specific underlying deformation model, and our indirectly supervised training does not require ground-truth rigging and skinning parameters.

3 OVERVIEW

Our goal is to animate a newly designed character using available mocap data and incorporate high quality pose dependent deformations. In this problem formulation, it is desirable to pre-define the skeletal structure of the character, *i.e.,* the bone hierarchy and joint adjacency, to be equivalent to the mocap data skeletal structure.

Given a new unseen character, we use a deep neural network to generate the parameters that enable articulation with pose-specific corrective deformations. The structure of our network is inspired by the classical animation pipeline and outputs three main components: rigging, skinning, and blend shapes. The number of degrees of freedom and the hierarchy of the underlying skeletal structure are pre-defined and *embedded* in the network, ensuring mocap compatibility for animating the character.

Our framework contains two main branches: (i) an envelope deformation branch that learns pose-invariant parameters (*i.e.,* rigging and skinning), and a (ii) residual deformation branch that learns pose-dependent residual displacements. The learned skeleton *rig* is bound to the input geometry using estimated *skinning* weights. When combined with joint rotations, this defines an envelope deformation that is capable of articulating the shape. Our neural blend shapes are inspired by SMPL [Loper et al. 2015], which proposed a comprehensive model for creating exceptional deformation quality using blend shapes represented as bases of additive displacements to the input character in rest pose.

Our network is trained on characters with the same articulation structure (*i.e.,* bipeds), but which may have different underlying deformation models. Thus, no ground truth is provided to the network-estimated rigging, skinning and blend shapes. We use indirect supervision, namely, instead of directly supervising the deformation parameters (rigging, skinning, and blend shapes) our network infers them by observing how articulated vertex positions are controlled by a set of joint rotations. As a result, the network learns to represent the articulation of every input character using the pre-defined envelope model, regardless of the underlying deformation model used during training. Our indirect supervision enables learning an arbitrary number of blend shapes, which we use to generate a smaller amount of blend shapes than the original training data, as well as a learned mask on the generated blend shapes. The network predicts compact and localized blend shapes that are pose-dependent *by construction*, without the need for such blend shapes as supervision.

Throughout the next sections we use the following notations: $\mathbf{V}$ and $\mathbf{F}$ denote the vertex positions and the connectivity of the input mesh, respectively. $\mathbf{W}$ is the output skinning weight matrix, $\mathbf{O}$ is a hierarchical set of offsets that represent the output skeleton and $\{\mathbf{B}_i\}_{i=1}^{N}$ is a set of N residual shapes that represent the blend shapes, interpolated with scalar coefficients $\{\alpha_i\}_{i=1}^{N}$.

4 METHOD

Below we describe our neural articulation framework, which consists of two main branches: an envelope deformation branch for rigging and skinning, and a residual pose-dependent deformation branch that enables predicting high-quality deformations.

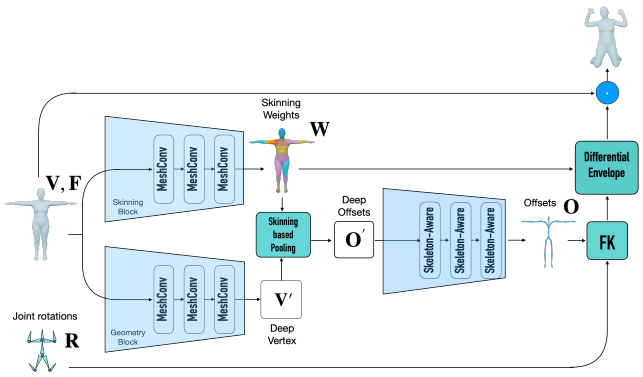


Fig. 3. The envelope deformation branch. Given a mesh in T-pose $(\mathbf{V}, \mathbf{F})$ and joint rotations $(\mathbf{R})$, our network infers the skinning $(\mathbf{W})$ and rigging $(\mathbf{O})$ parameters via indirect supervision by observing the articulated character vertex positions.

4.1 Envelope deformation branch

Our envelope deformation branch, illustrated in Figure 3, follows the typical animation workflow of rigging and skinning. The detailed architecture is provided in the Appendix.

Starting with an input character represented by a triangle mesh with vertices $\mathbf{V} \in \mathbb{R}^{V\times 3}$ and faces $\mathbf{F}$, our envelope deformation network predicts skeletal offsets (the offsets between each joint to its parent) $\mathbf{O} \in \mathbb{R}^{J\times 3}$ with a prescribed skeleton hierarchy that contains J joints, and a skinning weight matrix $\mathbf{W} \in \mathbb{R}^{V\times J}$.

Since our network is not directly supervised by skinning matrices, our envelope branch can obtain better deformation quality than standard LBS.

Our network learns to fit a rig of a pre-defined skeletal topology to any input character using indirect supervision. During training, the network is only supervised by the vertex positions of the articulated characters and the corresponding joint rotations. The network learns the relationship between joint rotations and the articulated character via the estimated rigging and skinning parameters, which are embedded in the network. Thus, with a sufficiently large number of examples, the network infers accurate rigging and skinning parameters for the input character (for more details please refer to the experiments in Section 5).

Skinning. To produce skinning weights, we incorporate a series of mesh convolution blocks using the MeshCNN operators of Hanocka et al. [2019]. Their work demonstrates that mesh convolutions are particularly useful for classification of surface parts (*i.e.,* segmentation), which is similar in spirit to skinning weights. However, our input features are different from MeshCNN. For each edge, we calculate the average positions of its two adjacent vertices. Furthermore, we max-pool one out of five of the output channels in each hidden layer then repeat and concatenate the result along the edge axis to extend the receptive field, similar to the segmentation network presented in PointNet [Qi et al. 2017]. After a forward pass, in order to predict per-vertex values, we average adjacent edge features of the corresponding vertex based on the mesh connectivity (similar to Point2Mesh [Hanocka et al. 2020]) to get the skin matrix $\mathbf{W}$.

Rigging. Given $\mathbf{V}$ and $\mathbf{F}$, our goal is to learn the rigging parameters $\mathbf{O} \in \mathbb{R}^{J\times 3}$ of a specific skeleton hierarchy that consists of J offsets. Intuitively, each offset $\mathbf{O}_j$ of the character's rig can be inferred from its surrounding mesh vertices. To learn a vertex representation that fits that task, we first pass the edge representation of $\mathbf{V}$ (similar to the skinning block) through several MeshCNN blocks to obtain a learned deep vertex representation $\mathbf{V}^{'} \in \mathbb{R}^{V\times K}$ with K channels. Then, the output skinning matrix is used to apply a *skinning based pooling* on the deep vertices, which collapses the V features into a set of J deep offsets using the relative skinning weight via

$$\mathbf{O}^{'}_j = \frac{\sum_{i=1}^{V} \mathbf{W}_{ij}\mathbf{V}^{'}_i}{\sum_{i=1}^{V} \mathbf{W}_{ij}}, \tag{1}$$

where $\mathbf{O}^{'}_j \in \mathbb{R}^K$ represents a deep feature corresponding to the jth offset, and $\mathbf{W}_{ij}$ is the skin weight that ties vertex i to offset j. This operation is similar to attention based pooling, and ensures that each offset is calculated only as a function of the vertices that are bound to it.

Given the deep offsets $\mathbf{O}^{'}$, we predict the explicit skeleton offsets to construct the rig. Since the skeletal topology is fixed in our network, we can exploit joint connectivity, such that each offset is calculated only by its corresponding deep offset and its close neighbors. Hence, we use a block of skeleton-aware operators [Aberman et al. 2020] to predict the explicit offset $\mathbf{O} \in \mathbb{R}^{J\times 3}$.

Envelope training. In order to learn skinning and rigging parameters that are not provided during training, in each iteration we inject a pose described by local joint rotations $\mathbf{R} = \{\mathbf{R}_i\}$ where $\mathbf{R}_i \in \mathbb{R}^{3\times 3}$, which guides the deformation of the input character along with the predicted rig and skin. We use two steps to convert the local joint rotations and offsets to a global affine per-joint transformation $\mathbf{T}_i \in \mathbb{R}^{4\times 4}$, which can be applied to the input vertices. First, for each joint we accumulate the local affine transformations $\{\mathbf{R}_i, \mathbf{O}_i\}$ along its kinematic chain (starting from the root) through a forward kinematics layer. Then, we apply a differential linear blend skinning (LBS) layer that calculates a per-vertex global transformation based on the skinning matrix via

$$\mathbf{T}_{\mathbf{R}_j} = \sum_i \mathbf{W}_{ji}\mathbf{T}_i. \tag{2}$$

Once the transformation is calculated, the per-vertex affine transformation $\mathbf{T}_\mathbf{R} = \{\mathbf{T}_{\mathbf{R}_j}\}$ is applied to the input character:

$$\tilde{\mathbf{V}}_\mathbf{R} = \mathbf{T}_\mathbf{R} \odot \mathbf{V}, \tag{3}$$

where $\odot$ denotes the per-vertex operation of the global affine transformations $\mathbf{T}_\mathbf{R}$ on the input vertices. An ℓ_2-loss is applied to the difference between the reconstructed vertex positions to the ground-truth articulation $\mathbf{V}_\mathbf{R}$:

$$\mathcal{L}_\mathrm{v} = \|\tilde{\mathbf{V}}_\mathbf{R} - \mathbf{V}_\mathbf{R}\|^2. \tag{4}$$

4.2 Residual deformation branch

Our residual deformation branch is inspired by the concept of blend shapes, and predicts a set of fixed residual shapes that are interpolated by pose-dependent coefficients and added to the input character to improve the deformation quality (illustration in Figure 4). In our case, both the shapes and their coefficients are learned by a neural network. During inference we feed the input character to the network once to receive the residual blend shapes, whereas the joint rotations for every frame are needed to animate the pose-dependent deformations for the character in real-time, as illustrated in Figure 5.

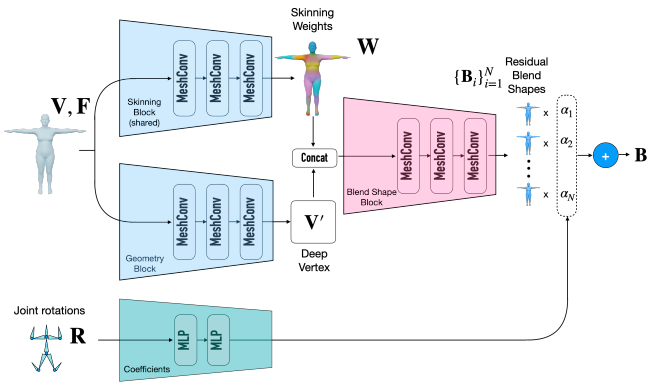


Fig. 4. The residual deformation branch predicts blend shapes and blending coefficients based on the input mesh connectivity and input joint rotations. The network learns to estimate blend shapes for arbitrary mesh connectivities and blending coefficients that are conditioned on the joint rotations.

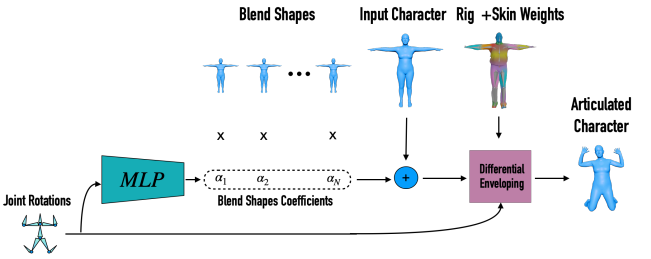


Fig. 5. Inference time. Given a new character, we can extract skin, rig, and blend shapes within a single forward pass of our network. To animate the character we need only a small network that calculates pose dependent blend shape coefficients. Conforming to standard skeletal animation models enables direct plug-and-play application in standard animation software.

Residual blend shapes. Given the input vertex positions $\mathbf{V}$ and connectivity $\mathbf{F}$, the residual branch starts with the skinning and geometry blocks with fixed weights that were pretrained in the envelope deformation branch. Then the output skinning $\mathbf{W}$ is concatenated to the deep vertices $\mathbf{V}^{'}$ along the channel dimension ($\{\mathbf{V}^{'}, \mathbf{W}\} \in \mathbb{R}^{V\times(K+J)}$) and the result is fed into the network. The combination of deep vertex and skin provides the blend shapes network with rich information about the vertices and their relationship to the skeleton, which is essential for the generation of the blend shapes. Similar to the envelope branch, we use the edge feature representations of these three components, which are passed through a block of mesh convolutions, resulting in a set of N residual shapes $\{\mathbf{B}_i\}_{i=1}^N$, $\mathbf{B}_i \in \mathbb{R}^{V\times 3}$. In parallel, we feed a small neural network that contains J MLP blocks, where each is conditioned by a single joint rotation, and output a series of pose dependent coefficients $\{\alpha_{ij}\}_{i=1}^N$ per joint j. These coefficients are used to interpolate between the

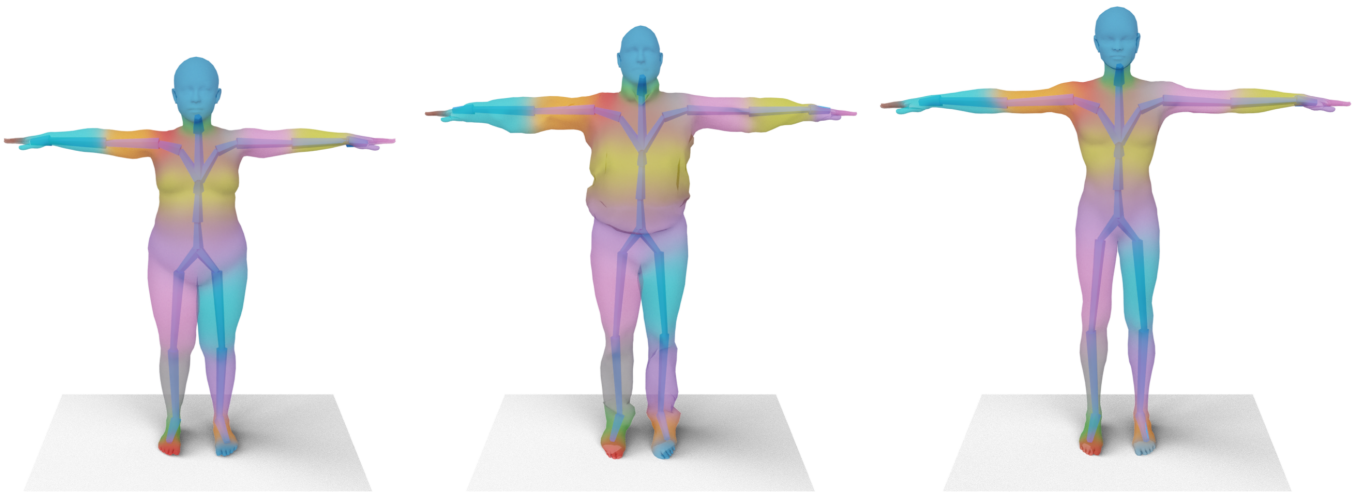


Fig. 6. Visualization of the network predicted rigging and skinning weights for various unseen characters. From left to right we see random test characters from each of the three test set groups: SMPL dataset, garment dataset, and characters handcrafted by 3D animator.

residual shapes that are summed up and added to the input vertices:

$$\tilde{\mathbf{V}} = \mathbf{V} + \sum_{j=1}^{J} \sum_{i=1}^{N} \alpha_{ij} m_j \mathbf{B}_i, \tag{5}$$

where m_j is a binary mask that specifies the vertices that are associated with joint j. This computation is done by picking all the non-zero entries in the skinning matrix that are associated with the two bones corresponding to the joint. This operation enables us to enforce localization in the structure of the blend shape and avoid undesired deformation of vertices associated with static joints (similar to [Osman et al. 2020]). Similar to the envelop branch, the loss is calculated as the difference between the articulated character and the corresponding ground truth using Eq. (4).

5 EXPERIMENTS AND EVALUATIONS

In this section we evaluate our results, compare them to other rigging, skinning, and deformation techniques, and demonstrate the effectiveness of various components in our framework through ablation study. In order to qualitatively evaluate our results to the fullest extent, please refer to the supplementary video.

5.1 Implementation details

Our neural articulation framework is implemented using the PyTorch library [Paszke et al. 2019], and the experiments were performed on NVIDIA GeForce GTX Titan Xp GPU (12 GB) and Intel Core i7-6950X/3.0GHz, CPU (16 GB RAM).

We train the network using a two stage course to fine approach. In the first phase, we train the envelope branch, and in the second phase, we fix the envelope network weights and train the residual branch. In this phase, we found that for our specific training dataset, we can boost the performance by providing supervision for the blend-shapes. However, since the blend-shape of SMPL and our model do not share the same properties (number of blend-shapes, number of vertices per blend shape) we extract our blend-shapes ground truth samples by optimizing the MLP network and $\{\tilde{\mathbf{B}}_i\}_{i=1}^{N}$ such that the output satisfies some high quality deformed ground truth. Then the extracted blend shapes can be used to supervise directly the generation of residual shapes $\{\mathbf{B}_i\}_{i=1}^{N}$. We optimize the parameters of our framework using the Adam optimizer [Kingma and Ba 2014]. We use different learning rates for the different blocks, and these are specified in Table 5. Training our network took about 3 days.

5.1.1 Data. Our network is trained on the SMPL dataset [Loper et al. 2015] which contains ten *shape* (pose-independent) blend shapes and 207 pose-dependent blend shapes. This enables generating a variety (*e.g.,* height, weight, proportions) of different shape identities and high quality deformations using the joint rotations provided in the SMPL model. Since the SMPL shapes represent a relatively *sterile* character (*i.e.,* naked and hairless humans), we incorporate an additional dataset of clothed humans proposed in Multi Garment Network [Bhatnagar et al. 2019]. The latter contains 96 characters with SMPL mesh connectivity. We used 80 out of the 96 clothed humans for training (reserving 16 models for testing). To train the network, we sample joint rotations from two different distributions (one for each branch). For the envelop deformation branch, the distribution for a single joint is $U\left(\mathbb{S}^2\right) \times \mathcal{N}\left(0, (\pi/6)^2\right)$, namely, the rotation axis is uniformly sampled from a 3D sphere and the rotation angle is sampled from a normal distribution with zero mean and variance of $(\pi/6)^2$. For the residual deformation branch, the rotation distribution is $U\left(\mathbb{S}^2\right) \times U\left[0, 2\pi\right]$ to enable capturing of even larger and exaggerated deformations. In each branch we sample rotations for each joint except for the root joint.

Our network assumes that the input character has a consistent upright and front facing orientation. Following SMPL [Loper et al. 2015], the input should also be in T-pose in order to effectively learn blend shapes, which is important for obtaining high quality deformations. In addition, the vertex positions are spatially aligned during training such that the hand tips are in a fixed height (as in

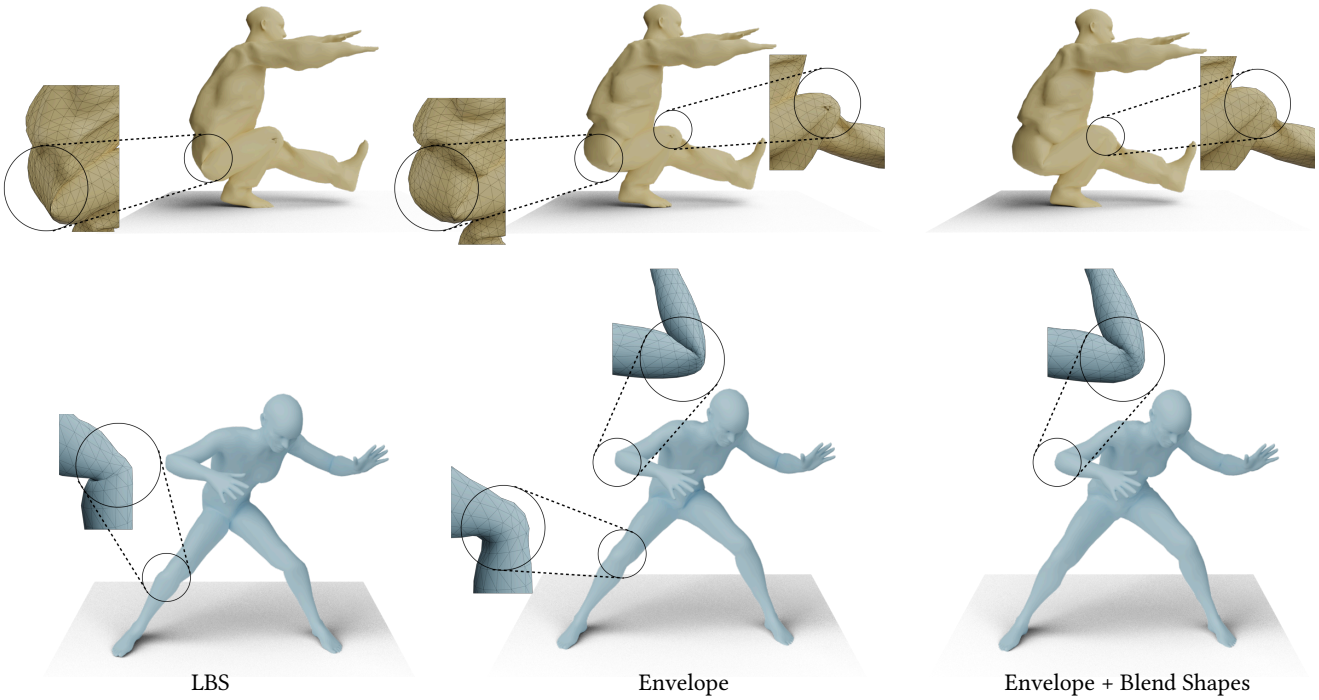


Fig. 7. Our predicted envelope deformation produces favorable results, compared to LBS. Adding neural blend shapes to the enveloping results in corrective pose-dependent displacement, which improves the deformation quality in the joint regions.

the SMPL model). In test time, if the character does not belong to the SMPL distribution, we translate it to the same height and scale the vertices such that the two extreme hand tip vertices have a certain distance, similar to the distance in the raw SMPL model, with no shape blend shapes.

As outlined in MeshCNN [Hanocka et al. 2019], our system can handle inputs which contain boundaries, self-intersections, or disconnected components. In the case of non-manifold geometries (non-manifold mesh edges or vertices), these should simply be deleted. In general, there should be a meaningful shape surface to propagate the deep mesh features; so in the case of an extreme polygon soup some form of remeshing [research 2020] or tetrahedralization [Hu et al. 2020] can be employed.

Test data. Our test set consists of three groups. The first group is five biped characters manually created by a professional animator. The second group is thirty random SMPL characters, which are sampled with different proportions and shapes than we trained on. The third group is a random subset of 16 characters from the SMPL clothes dataset that were reserved for the test set.

Garment augmentation. We also use the dataset of Bhatnagar et al. [2019] to enlarge the geometric features observed by our network. Since the shapes in this dataset have the same connectivity as SMPL, we can extract the garment displacements in two steps. First, we find the closet SMPL character that matches the surface of the clothed character (via optimization on SMPL parametric space), and then we subtract the clothed character from the fitted SMPL character to get a set of displacements, which are used to augment shapes by adding garments. This creates a variety of geometric variability in the training data, especially with regard to the location of the skin w.r.t. the character bone. We show the importance of this augmentation in the ablation study.

Mesh connectivity augmentation. In order to enrich the set of samples that the network is trained on and to enhance its robustness to different mesh connectivity, we augment the data samples by applying three topological operators on edges: collapse, split and flip [Hoppe 1996; Botsch et al. 2010].

5.2 Experiments

Our framework can generate high quality rigs with skin weights and blend shapes on various characters with arbitrary connectivity. Figure 6 shows some of our results. The test characters shown here are from each of the three test data groups: the SMPL dataset, the garment dataset, and the characters handcrafted by a 3D animator. None of these characters are used during training. In order to visualize the skinning weights we associated each bone in the skeleton with a unique color, then the color of the vertex is calculated as weighted average of the colors of the bones bound to it in the skinning weights.

Figure 7 demonstrates the quality of the deformations generated by our predicted skeleton rigs. The baseline deformations on the left are generated using LBS weights and the ground-truth skinning weights of each character, while the deformations generated by our envelope deformation branch and residual deformation branch are shown in the middle and on the right of the figure, respectively.

Notably, because our envelope branch is trained on high-quality data, we can already generate mesh deformations that are better than the baseline results using LBS and the skinning weights computed by our envelope model. This improvement can be easily spotted in Figure 7, where the volume loss around the buttocks area is significantly mitigated in the results showing in the middle. The quality of the deformation can be further improved using the blend shapes computed by our residual deformation branch. As shown in Figure 7, the volume of the meshes around the knee and the elbow are preserved even when the corresponding joints are transformed dramatically.

A challenging task for neural networks is the ability to generalize and extrapolate beyond the training data, which will always contain only a *subset* of the real world data we expect to encounter in test time. We validate how robust our system is to characters found *in the wild*, by animating characters from the Mixamo dataset [Adobe Systems Inc. 2018] using our model which has not seen Mixamo characters during training. Mixamo is a particularly challenging set since the characters contain vastly different mesh connectivity, body proportions, and decorative geometries. While Mixamo already contains rigging and skinning, it does not contain the required skeletal structure for animating with a given mocap data, and there are no blend shapes or pose-dependent corrective deformations. To this end, we use our system to predict a rig with a desired skeletal-structure, enabling animating Mixamo characters using mocap motions with neural blend shapes. This result is especially remarkable, since the mesh contains three times the amount of vertices which we trained our network on. In Figure 8(a), we see the result of our skinning and rigging on the Mixamo character, where our predicted skeleton rig is ready to animate with the given mocap data. Our neural blend shapes creates a corrective deformation in the muscle area resulting in accurate preservation of the muscle bulging, whereas as the original Mixamo dataset does not incorporate high quality deformations as shown in Figure 8(c) and the supplementary video. In addition, see Figure 8(b), where the original Mixamo character rig is not compatible with mocap skeleton structure, whereas our network predicts a mocap-ready skeleton on the same character.

Connectivity robustness. In this experiment, we demonstrate that training with mesh connectivity augmentations (edge flip, split, collapse) results in a system that is robust to changes in the input mesh connectivity. Given the same input character, we perform a significant amount of connectivity augmentations to achieve variations of the mesh input. We observe that the network estimated skinning and rigging parameters remain stable, which can be seen in Figure 9. We also show that the corresponding enveloping produces stable deformations for such input connectivity variations, which can be seen in the supplementary video.

Neural Blend Shapes. In this experiment we visualized the corrective effect of the learned pose dependent blend shapes. For a given character, we fed our network with 3 sets of joint rotations (equivalent to 3 poses) and visualized the output in Figure 10. The top row exhibits the deformed character for each of the poses, while the bottom raw visualizes the magnitude of the resulting displacements (via color map), on top of the same character in rest pose.

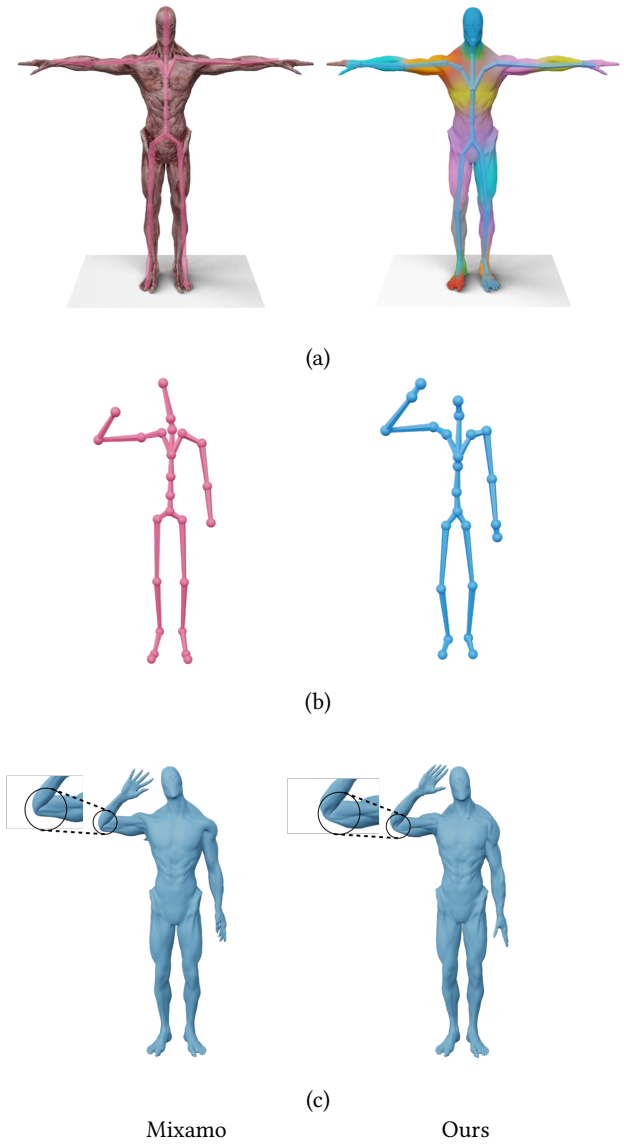


Fig. 8. Our system generalizes to characters from other datasets (Mixamo). (a) Original Mixamo character and its skeleton (left) and our mocap-ready output skeleton with the corresponding skinning weights (right). (b) Original Mixamo skeleton posed (left) and our mocap-ready skeleton posed (right). (c) Original Mixamo deformation (left) and our high quality deformation which is achieved by the corrective blend shape that predicts muscle bulge (right).

It can be seen that our neural blend shapes displace the vertices in regions that correspond to the bent joint (elbow, knee, and hip joints) resulting in a corrective deformation in these regions.

In addition to the corrective displacements, we also visualize the learned blending coefficients on top of a particular motion in order to examine the active joints which can be seen in Figure 11. For each pose, each joint is colorized according to the average values of

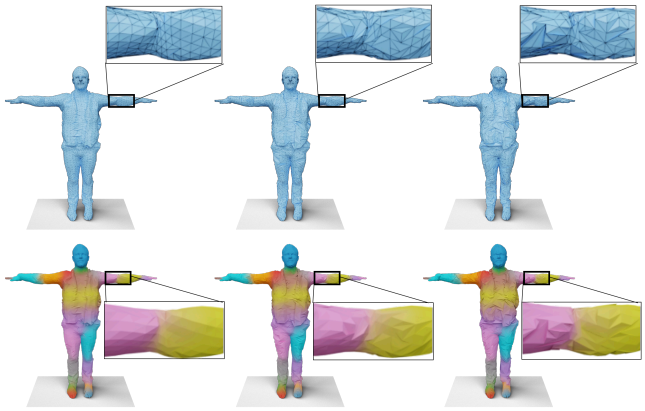


Fig. 9. Robustness to variations in connectivity. Given the same input character, we perform connectivity augmentations and observe that the corresponding skinning weights are stable.

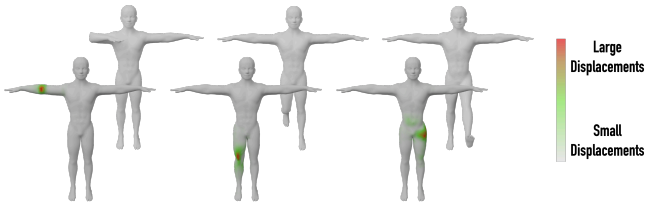


Fig. 10. Neural blend shape corrective displacement visualization. Top: the same shape is shown in three different poses, bottom: corresponding blend shape displacement (visualized with a heat map). Observe that a rotation in the elbow, knee, and hip joints resulted in corrective deformations in the corresponding regions.

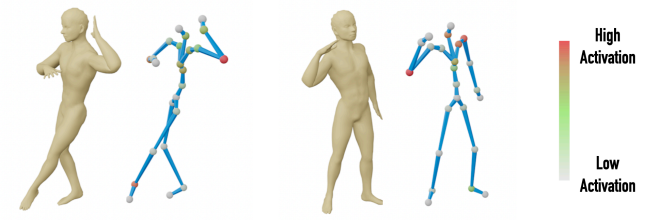


Fig. 11. Neural blend shape coefficients visualization. In each pair the left shape is our output deformed character, and on the right is the corresponding posed rig. The joints are colored by the blend shape coefficient activation corresponding to the joint.

the displacement that are associated with it based on the skinning matrix. The results demonstrate that when the joint is active (bent) the activation of the corresponding coefficients is higher. Please refer to the supplementary video to see the full animation.

In addition, we ran an experiment to evaluate the number of blend shapes needed to obtain high quality pose dependent deformations. We trained the network with a varying number of blend shapes N and found that 9 blend shapes was enough to obtain high quality deformations, both quantitatively and qualitatively. See the quantitative results in Figure 12.

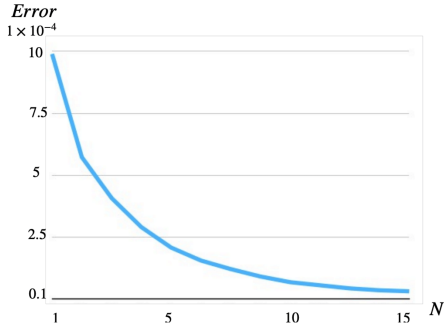


Fig. 12. Number of blend shapes as a function of high quality deformation error.

5.3 Evaluation

In this section we compare our method to state-of-the-art skinning and rigging methods as well as off-the-shelf tools.

Rigging. We first compare to the recent work of Xu et al. [2020] which proposed a deep neural network that learns to generate arbitrary skeletal rigs and the corresponding skinning weights using supervised learning, and demonstrated impressive performance on shapes with varying skeletal hierarchies. However, their system does not enable the user to control the output skeletal hierarchy but only allows to modify the density of the predicted joints through a single scalar. Yet, animating an arbitrary skeleton rig using mocap data is not directly possible due to incompatibility in the skeletal structure. The latter requires motion retargeting between two skeletons with different, unseen, structures, which is still an open problem [Gleicher 1998; Aberman et al. 2020]. Although the recent work of Aberman et al. [2020] proposed a method to retarget motion of skeletons with different hierarchical structures, it requires having datasets contain different characters with the exact same hierarchical structures for both source and target skeletons, which is incompatible with our setting. Figure 13 shows 3 different outputs of RigNet for an input character that was designed by a 3D artist and for different input scalar values (0.015, 0.028, 0.09). It can be seen that every output rig contains different number of joints (12, 25, 50) and different skeletal hierarchies which can not be controlled directly by the user. In practice, we tried to find the scalar value that leads to 24 joints – the number of joints in our target skeleton, and couldn't (the closet we found is 25). In contrast, our network predicts skeleton with a fixed hierarchy which is embedded in the network, thus, can be animated with corresponding mocap data.

We next compare our results to the method of Baran and Popović [2007] (a.k.a Pinocchio) which fits a template skeleton for each input character. The target template is selected from a set of predefined skeletal hierarchies based on a cost function that evaluates its geometric fitting to the input shape. Figure 14 shows the comparison to the automatic rigging results of Pinocchio and RigNet for two different humanoids. It can be seen that the selected output skeletal

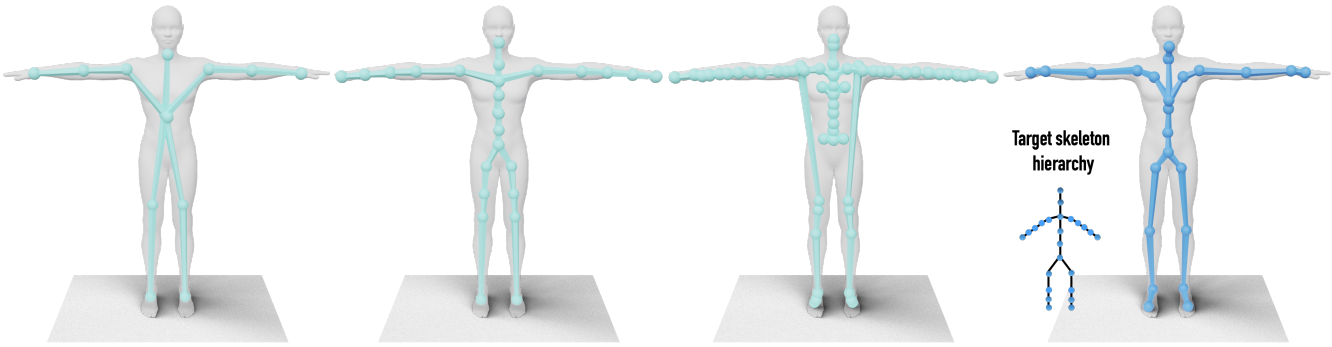


Fig. 13. Predicted skeleton rig on a character designed by a 3D artist. RigNet [Xu et al. 2020] only provides scalar control over the granularity of the skeleton (shown: predicted rig with 12, 25, or 50 bones corresponding to the scalar 0.015, 0.028, 0.09), but cannot control the skeleton hierarchy. We were not able to obtain the desired number 24, which is achieved by the output of our method, complying with the target skeletal hierarchy (rightmost example). Note the undesirable placement of the root node, and extraneous joints in the knee region.

Table 1. Quantitative comparison between our rigging results to the Pinocchio [Baran and Popović 2007] and RigNet [Xu et al. 2020].

	CD-J2J	CD-J2B	CD-B2B
Pinocchio [Baran and Popović 2007]	0.474	0.164	0.025
RigNet [Xu et al. 2020]	0.194	0.084	0.009
Ours	**0.012**	**0.007**	**0.004**

Table 2. Quantitative comparison between our skinning results to Pinocchio [Baran and Popović 2007].

	Skinning Weight(L1)	Avg Dist.	Max Dist.
Pinocchio [2007]	23.1	0.27	20.3
Ours	**2.56**	**0.011**	**1.68**

Table 3. Ablation Study

Envelope	No Mesh Aug	No Garm Aug	All
0.024	0.66	0.23	**0.011**

hierarchy of Pinocchio is sparse (18 joints), which limits the granularity of deformation that can be achieved and requires a manual specification of joint correspondence in order to animate the output with motion from mocap data.

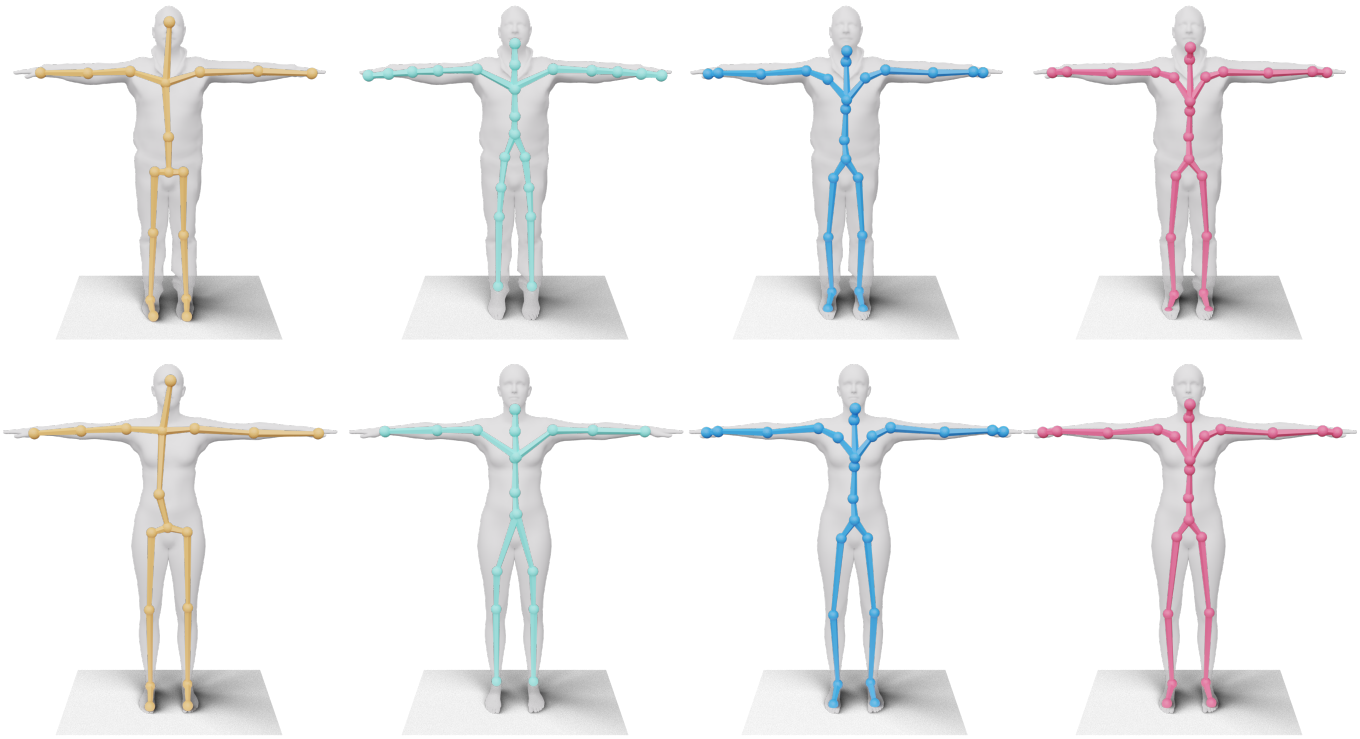
We employ the metrics CD-J2J, CD-J2B, and CD-B2B proposed by [Xu et al. 2020] to quantitatively evaluate the automatic rigging results. Briefly speaking, these metrics evaluates the quality of a rigging by measuring the spacial distances between the joints and bones. Ideally, all CD-J2J, CD-J2B, and CD-B2B measures should be low for a high quality rig. We refer interested reader to the original paper for detailed definitions of these metrics. The results are reported in Table 1. It can be seen that our method outperforms the other methods quantitatively. Note that the metrics enables to calculate distances between skeleton with different hierarchies.

Skinning and Deformation. We compared our skinning and deformation results to the output of Blender software which uses an updated version of the skinning algorithm from Pinocchio [Baran and Popović 2007]. To measure the skinning error we use a simple L1 metric between the estimated skinning matrix and the ground-truth one. In order to perform this comparison we had to ensure that the number of joints is similar in all of the outputs, thus, we provided Blender with the ground-truth skeleton. In this way, we received skinning matrix with similar dimensions to the ground truth. For the deformation evaluation, we have chosen a fixed test sequence for each comparison and calculated the average error of vertex displacements and max error. The results are reported in Table 2 and the output skinning matrix weights of each methods is visualized in Figure 15. It can be seen that our method out performs the Blender software quantitatively and qualitative.

5.4 Ablation study

Residual Branch. In this section, we perform an ablation study to evaluate the importance of each of the components in our system. In particular, we retrain our framework (a) without neural blend shapes (only envelope), (b) without connectivity augmentation, and (c) without garment augmentation. We calculate the L2 distance between the ground-truth displaced vertices and the predicted vertex displacements on the test dataset. The results are shown in Table 3. We can see that each of these components is important to our system, and removing any one of them will result in a drop in accuracy. Moreover, the most critical component which has the most impact on quantitative deformation accuracy is the mesh augmentation component. Although only using envelope (without using neural blend shapes) has the least numeric influence of the three, it is extremely important for a high visual quality deformation as we have showed throughout the paper.

Connectivity Augmentation. The network trained without connectivity augmentations produces undesirable deformation artifacts in the joints, and struggles to generate meaningful corrective blend shape displacements, which can be seen in the supplementary video



Pinocchio [2007] RigNet [2020] Ours Ground truth

Fig. 14. Automatic rigging results on two different humanoids. Left to right: Pinocchio [2007], RigNet [2020], ours and ground truth.

and in Figure 16. This shows that our connectivity augmentations are key to the generalization capabilities of the neural blend shapes on different mesh connectivity.

Garment Augmentation. The network trained without garment augmentations struggles to generalize to unseen characters. This operation augments the input characters with piecewise smooth displacements extracted from the dataset of clothed characters and enables the network to not only generalize to clothed characters but also to achieve better results for naked, unseen characters. We believe this is because the garment augmentation creates a variety of geometric variability in the training data, especially with regard to the location of the skin w.r.t. the character bone. In particular, we observe that the garment augmentation is critical factor in enabling our system to generalize the character created by a 3D artist, which can be seen in Figure 17 and in the supplementary video.

Envelope. We trained our network only on the envelope deformation (*i.e.,* without neural blend shapes). While our envelope obtains better deformations than LBS, it cannot produce pose-dependent corrective displacements in the joint regions. We can see that the quality of the deformation is improved when using the blend shapes residual corrective displacement. A result of this is shown in Figure 7, where the volume of the mesh around the elbow are perfectly preserved even when the corresponding joints are transformed dramatically. Please refer to the supplementary video for more results.

Table 4. Indirect supervision - Ablation study

	Indirect Sup.	Direct Sup.
$\mathcal{L}_w$ converges to	0.14	0.10
$\mathcal{L}_s$ converges to	0.012	0.009
# of iters to reach $\mathcal{L}_v = 0.024$	80,000	45,000

Indirect Supervision. Our network is trained with *indirect supervision*, namely, ground-truth skinning and rigging samples are not provided during training, and the network is supervised only by the ground-truth deformation. This relaxes the assumption that the data samples should have a specific underlying deformation model. However, our training framework can easily employ direct supervision when ground-truth samples of skinning and rigging are available. In such a case, we can apply ℓ_2-loss to the difference between the generated skeleton and skinning weight to the ground-truth via

$$\mathcal{L}_{\text{supervised}} = \mathcal{L}_s + \mathcal{L}_w = \|\tilde{O} - O\|^2 + \|\tilde{W} - W\|^2, \quad (6)$$

where $\mathcal{L}_s$ and $\mathcal{L}_w$ are the skeleton and skinning loss terms.

As demonstrated throughout the paper, indirect supervision approach is capable to learn high deformation quality. However, as shown in Table 4, direct supervision configuration converges faster comparing to the indirect supervision, although the deformation quality of both training configurations is comparable.

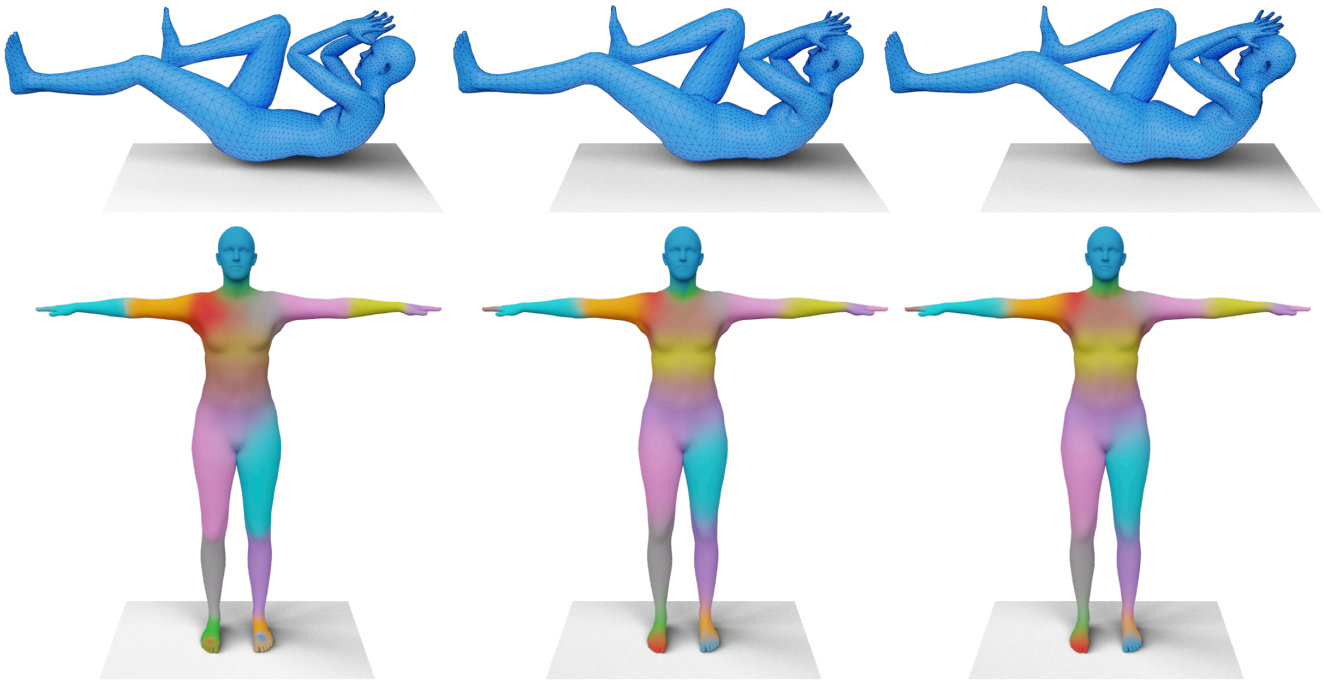


Fig. 15. Comparison with the automatic skinning and deformation method in Blender (Based on Pinocchio [2007]).

6 DISCUSSION AND CONCLUSION

We presented an approach to train a neural network to rig and skin an input character mesh with a specific, prescribed skeleton structure and automatically generate neural blend shapes to enhance the articulated deformation quality in a pose-dependent manner. The fact that our framework incorporates the desired skeleton structure makes it practical for animation with existing motion data, such as available mocap libraries or legacy animation data of previously designed characters; the overall process is compatible with typical workflows in animation software. At the same time, the learned blend shapes ensure high quality of the output deformation, avoiding the usual LBS pitfalls, and are in general a powerful means to learn various deformation effects and apply them to unseen meshes with arbitrary connectivity. Unlike many existing example-based approaches, our system only requires a single mesh as input to compute the skeleton rig and the neural blend shapes, which makes it applicable to a large range of scenarios.

As is typical in the deep learning paradigm, the output of our method is bounded by the quality of the deformations that exist in the training dataset. Most of the results shown in this paper are trained using examples generated by SMPL-like models [Loper et al. 2015; Bhatnagar et al. 2019]. The resulting rigs give rise to high quality mesh deformation in most of our experiments, but can generate artifacts in certain cases where the selected model was not trained sufficiently. Fortunately, our indirectly supervised learning does not assume that the training data has a specific underlying model and can be easily extended to leverage other types of examples to learn versatile deformations. A direct enhancement to the results shown here is to train the system based on production rigs. We are also interested in exploring the possibility of learning secondary dynamics from simulation in the future.

Our system is trained to generate rigs with a prescribed skeleton structure. Changing this skeleton structure, *e.g.*, adding extra joints or altering the connectivity, requires retraining the entire network. In future work, it would be interesting to explore skeleton-aware models, such as those proposed in [Aberman et al. 2020], to embed the skeleton structure in the rigging network to support arbitrary rigs using the same model, as well as relaxing some of the assumption we make on the input character (manifold mesh, T-pose shape, etc.). Automatically adapting and combining multiple template skeleton structures to multi-component characters [Miller et al. 2010; Bharaj et al. 2012] would be another interesting problem to explore.

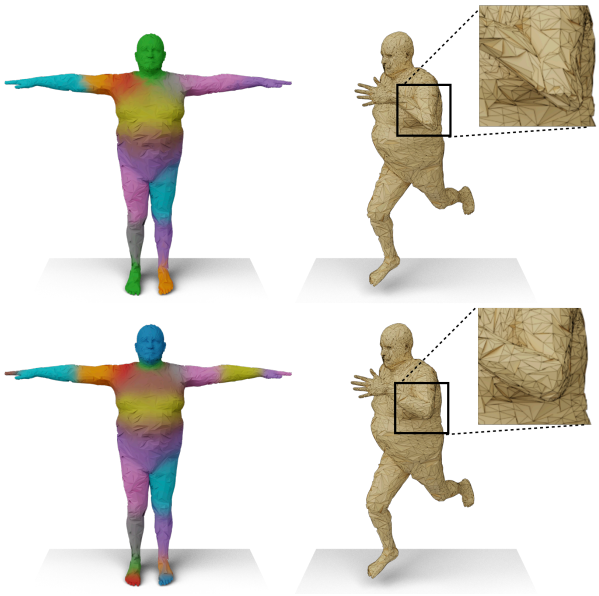


Fig. 16. Connectivity augmentation ablation. Top: network trained without connectivity augmentation, bottom: our network trained with connectivity augmentations. On the left are the network predicted skinning weights and on the right the deformed character. Observe that training with connectivity augmentations are key to generalization of the neural blend shapes on different mesh connectivity.

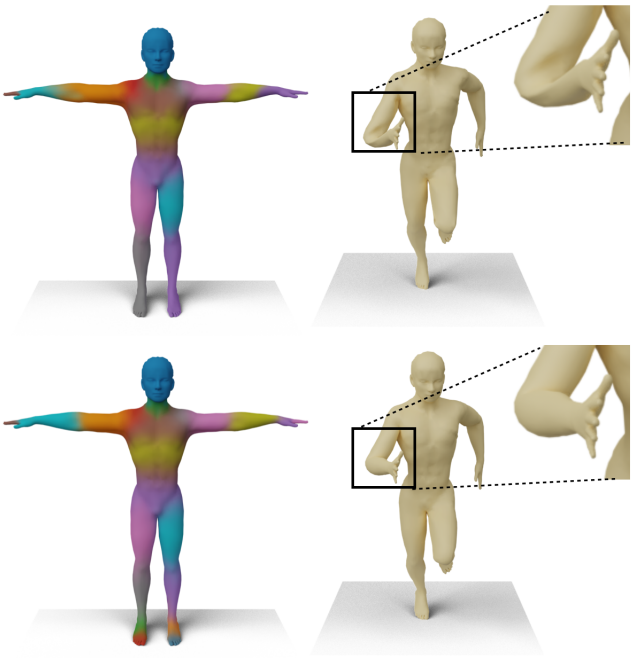


Fig. 17. Garment augmentation ablation. Top: network trained without garment augmentation, bottom: our network trained with garment augmentations. On the left are the network predicted skinning weights and on the right the deformed character. Training with garment augmentations are key to our systems ability to generalize to the character created by a 3D artist.

ACKNOWLEDGMENTS

We thank Sigal Raab and Andreas Aristidou for their valuable inputs that helped to improve this work. We also thank Cong Li and Yulong Zhang for their kind help with the experiments and ablation study. This work was supported in part by National Key R&D Program of China (2019YFF0302902, 2020AAA0105200), Beijing Academy of Artificial Intelligence (BAAI), and Ant Group.

REFERENCES

Kfir Aberman, Peizhuo Li, Olga Sorkine-Hornung, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. 2020. Skeleton-Aware Networks for Deep Motion Retargeting. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 62.

Adobe Systems Inc. 2018. Mixamo. https://www.mixamo.com. https://www.mixamo.com Accessed: 2018-12-27.

Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. 2008. Skeleton Extraction by Mesh Contraction. *ACM Transactions on Graphics* 27, 3 (Aug. 2008), 1–10. https://doi.org/10.1145/1360612.1360643

Stephen W. Bailey, Dalton Omens, Paul Dilorenzo, and James F. O'Brien. 2020. Fast and Deep Facial Deformations. *ACM Trans. Graph.* 39, 4, Article 94 (July 2020), 15 pages. https://doi.org/10.1145/3386569.3392397

Stephen W Bailey, Dave Otte, Paul Dilorenzo, and James F O'Brien. 2018. Fast and deep deformation approximations. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

Ilya Baran and Jovan Popović. 2007. Automatic rigging and animation of 3d characters. *ACM Transactions on graphics (TOG)* 26, 3 (2007), 72–es.

Gaurav Bharaj, Thorsten Thormählen, Hans-Peter Seidel, and Christian Theobalt. 2012. Automatically Rigging Multi-component Characters. *Computer Graphics Forum* 31, 2pt4 (2012), 755–764. https://doi.org/10.1111/j.1467-8659.2012.03034.x _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2012.03034.x.

Bharat Lal Bhatnagar, Garvita Tiwari, Christian Theobalt, and Gerard Pons-Moll. 2019. Multi-Garment Net: Learning to Dress 3D People from Images. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE.

Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. 2010. *Polygon mesh processing*. CRC press.

Junjie Cao, Andrea Tagliasacchi, Matt Olson, Hao Zhang, and Zhinxun Su. 2010. Point Cloud Skeletons via Laplacian Based Contraction. In *2010 Shape Modeling International Conference*. IEEE, Aix-en-Provence, France, 187–197. https://doi.org/10.1109/SMI.2010.25

Edilson De Aguiar, Christian Theobalt, Sebastian Thrun, and Hans-Peter Seidel. 2008. Automatic conversion of mesh animations into skeleton-based animations. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 389–397.

Olivier Dionne and Martin de Lasa. 2013. Geodesic voxel binding for production character meshes. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '13)*. Association for Computing Machinery, New York, NY, USA, 173–180. https://doi.org/10.1145/2485895.2485919

Stefan Fröhlich and Mario Botsch. 2011. Example-driven deformations based on discrete shells. *Computer graphics forum* 30, 8 (2011), 2246–2257.

Michael Gleicher. 1998. Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 33–42.

Fabian Hahn, Sebastian Martin, Bernhard Thomaszewski, Robert Sumner, Stelian Coros, and Markus Gross. 2012. Rig-Space Physics. *ACM Transactions on Graphics* 31, 4 (July 2012), 72:1–72:8. https://doi.org/10.1145/2185520.2185568

Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. 2019. MeshCNN: A Network with an Edge. *ACM Trans. Graph.* 38, 4, Article 90 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3322959

Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. 2020. Point2Mesh: A Self-Prior for Deformable Meshes. *ACM Trans. Graph.* 39, 4, Article 126 (July 2020), 12 pages. https://doi.org/10.1145/3386569.3392415

Nils Hasler, Thorsten Thormählen, Bodo Rosenhahn, and Hans-Peter Seidel. 2010. Learning skeletons for shape and pose. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. 23–30.

Jim Hejl. 2004. Hardware skinning with quaternions. *Game Programming Gems* 4 (2004), 487–495.

Hugues Hoppe. 1996. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 99–108.

Yixin Hu, Teseo Schneider, Bolun Wang, Denis Zorin, and Daniele Panozzo. 2020. Fast Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 39, 4, Article 117 (July 2020), 18 pages. https://doi.org/10.1145/3386569.3392385

Alec Jacobson, Ilya Baran, Jovan Popovic, and Olga Sorkine. 2011. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.* 30, 4 (2011), 78.

Doug L. James and Christopher D. Twigg. 2005. Skinning mesh animations. In *ACM SIGGRAPH 2005 Papers (SIGGRAPH '05)*. Association for Computing Machinery, New York, NY, USA, 399–407. https://doi.org/10.1145/1186822.1073206

Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. 2007. Harmonic coordinates for character articulation. *ACM Transactions on Graphics* 26, 3 (July 2007), 71–es. https://doi.org/10.1145/1276377.1276466

Tao Ju, Scott Schaefer, and Joe Warren. 2005. Mean value coordinates for closed triangular meshes. *ACM Transactions on Graphics* 24, 3 (Jul 2005), 561–566. https://doi.org/10.1145/1073204.1073229

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2007. Skinning with dual quaternions. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*. 39–46.

Ladislav Kavan and Olga Sorkine. 2012. Elasticity-Inspired Deformers for Character Articulation. *ACM Trans. Graph.* 31, 6 (2012), 196:1–196:8.

Ladislav Kavan and Jiří Žára. 2005. Spherical blend skinning: a real-time deformation of articulated models. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*. 9–16.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Paul G. Kry, Doug L. James, and Dinesh K. Pai. 2002. EigenSkin: real time large deformation character skinning in hardware. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '02)*. Association for Computing Machinery, New York, NY, USA, 153–159. https://doi.org/10.1145/545261.545286

Binh Huy Le and Zhigang Deng. 2014. Robust and accurate skeletal rigging from mesh sequences. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–10.

Binh Huy Le and Jessica K Hodgins. 2016. Real-time skeletal skinning with optimized centers of rotation. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–10.

John P Lewis, Matt Cordner, and Nickson Fong. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 165–172.

Tianxing Li, Rui Shi, and Takashi Kanai. 2020. DenseGATs: A Graph-Attention-Based Network for Nonlinear Character Deformation. In *Symposium on Interactive 3D Graphics and Games*. 1–9.

Lijuan Liu, Youyi Zheng, Di Tang, Yi Yuan, Changjie Fan, and Kun Zhou. 2019. Neuroskinning: Automatic skin binding for production characters with deep graph networks. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. 2015. SMPL: A skinned multi-person linear model. *ACM transactions on graphics (TOG)* 34, 6 (2015), 1–16.

Nadia Magnenat-Thalmann, Richard Laperrire, and Daniel Thalmann. 1988. Joint-dependent local deformations for hand animation and object grasping. In *In Proceedings on Graphics interface'88*. Citeseer.

Bruce Merry, Patrick Marais, and James Gain. 2006. Animation space: A truly linear framework for character animation. *ACM Transactions on Graphics (TOG)* 25, 4 (2006), 1400–1423.

Christian Miller, Okan Arikan, and Don Fussell. 2010. Frankenrigs: building character rigs from multiple sources. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. 31–38.

Tomohiko Mukai and Shigeru Kuriyama. 2016. Efficient dynamic skinning with low-rank helper bone controllers. *ACM Transactions on Graphics* 35, 4 (July 2016), 36:1–36:11. https://doi.org/10.1145/2897824.2925905

Tina O'Hailey. 2018. *Rig it right! Maya animation rigging concepts*. Routledge.

Ahmed A A Osman, Timo Bolkart, and Michael J. Black. 2020. STAR: A Spare Trained Articulated Human Body Regressor. In *European Conference on Computer Vision (ECCV)*. https://star.is.tue.mpg.de

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 652–660.

Autodesk research. 2020. Meshmixer. https://www.meshmixer.com/

Scott Schaefer and Can Yuksel. 2007. Example-based skeleton extraction. In *Symposium on Geometry Processing*. 153–162.

Jaewoo Seo, Geoffrey Irving, J. P. Lewis, and Junyong Noh. 2011. Compression and direct manipulation of complex blendshape models. In *Proceedings of the 2011 SIGGRAPH Asia Conference (SA '11)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/2024156.2024198

Peter-Pike J. Sloan, Charles F. Rose, and Michael F. Cohen. 2001. Shape by example. In *Proceedings of the 2001 symposium on Interactive 3D graphics (I3D '01)*. Association for Computing Machinery, New York, NY, USA, 135–143. https://doi.org/10.1145/364338.364382

Name	Layers	i/o_c	LR
Skinning Block	`MConv + LReLU + Pool`	3/64	2×10^{-4}
	`MConv + LReLU + Pool`	64/128	
	`MConv + LReLU + Pool`	128/256	
	`MConv + Softmax`	256/24	
Geometry Block	`MConv + LReLU`	3/64	1×10^{-5}
	`MConv + LReLU`	64/128	(stage 1)
	`MConv + LReLU`	128/256	1×10^{-4}
	`MConv + LReLU`	256/512	(stage 2)
Offset Block	`SConv + LReLU`	512/256	1×10^{-4}
	`SConv + LReLU`	256/128	
	`SConv + LReLU`	128/64	
	`SConv`	64/3	
Blend Shape Block	`MConv + LReLU`	536/256	1×10^{-4}
	`MConv`	256/27	
Blend Shape Coefficient	`FC + LReLU`	9/18	1×10^{-3}
	`FC + LReLU`	18/32	
	`FC`	32/9	

Table 5. Network Architectures

Steven L Song, Weiqi Shi, and Michael Reed. 2020. Accurate Face Rig Approximation with Deep Differential Subspace Reconstruction. *arXiv preprint arXiv:2006.01746* (2020).

Robert W Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popović. 2005. Mesh-based inverse kinematics. *ACM transactions on graphics (TOG)* 24, 3 (2005), 488–495.

Xiaohuan Corina Wang and Cary Phillips. 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 129–138.

Rich Wareham and Joan Lasenby. 2008. Bone Glow: An Improved Method for the Assignment of Weights for Mesh Deformation. In *Articulated Motion and Deformable Objects (Lecture Notes in Computer Science)*, Francisco J. Perales and Robert B. Fisher (Eds.). Springer, Berlin, Heidelberg, 63–71. https://doi.org/10.1007/978-3-540-70517-8_7

Ofir Weber, Olga Sorkine, Yaron Lipman, and Craig Gotsman. 2007. Context-aware skeletal shape deformation. In *Computer Graphics Forum*, Vol. 26. Wiley Online Library, 265–274.

Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. 2020. RigNet: Neural Rigging for Articulated Characters. *ACM Transactions on Graphics* 39, 4 (July 2020). https://doi.org/10.1145/3386569.3392379

Wang Yifan, Noam Aigerman, Vladimir G. Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. 2020. Neural Cages for Detail-Preserving 3D Deformations. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Seattle, WA, USA, 72–80. https://doi.org/10.1109/CVPR42600.2020.00015

Jiayi Eris Zhang, Seungbae Bang, David I. W. Levin, and Alec Jacobson. 2020. Complementary Dynamics. *ACM Transactions on Graphics* 39, 6 (Nov. 2020), 179:1–179:11. https://doi.org/10.1145/3414685.3417819

A NETWORK ARCHITECTURES

In this section we describe the details for the network architectures.

Table 5 describes the architecture for our envelope deformation branch and residual deformation branch, where `MConv`, `SConv`, `FC`, `LReLU`, `Pool` and `Softmax` denote mesh convolution [Hanocka et al. 2019], skeleton-aware convolution [Aberman et al. 2020], fully connected layer, leaky ReLU, max-pool 1/5 channels and softmax activation.