CTSN: Predicting Cloth Deformation for Skeleton-based Characters with a Two-stream Skinning Network

Yudi Li
Zhejiang University
Hangzhou, China
drafocus@outlook.com

Min Tang
Zhejiang University
Hangzhou, China
https://min-tang.github.io/home/

Yun Yang
Zhejiang University
Hangzhou, China
syby119@126.com

Ruofeng Tong
Zhejiang University
Hangzhou, China
trf@zju.edu.cn

Shuangcai Yang
Tencent
Shenzhen, China
yscyang@tencent.com

Yao Li
Zhejiang University
Shenzhen, China
leoyaoli@tencent.com

Bailin An
Zhejiang University
Shenzhen, China
frankan@tencent.com

Qilong Kou
Zhejiang University
Shenzhen, China
kouqilong1988@gmail.com

Abstract

We present a novel learning method to predict the cloth deformation for skeleton-based characters with a two-stream network. The characters processed in our approach are not limited to humans, and can be other skeletal-based representations of non-human targets such as fish or pets. We use a novel network architecture which consists of skeleton-based and mesh-based residual networks to learn the coarse and wrinkle features as the overall residual from the template cloth mesh. Our network is used to predict the deformation for loose or tight-fitting clothing or dresses. We ensure that the memory footprint of our network is low, and thereby result in reduced storage and computational requirements. In practice, our prediction for a single cloth mesh for the skeleton-based character takes about 7 milliseconds on an NVIDIA GeForce RTX 3090 GPU. Compared with prior methods, our network can generate fine deformation results with details and wrinkles.

Keywords: Cloth deformation, learning based network, skinning

1. Introduction

Cloth animation is an important problem in computer graphics due to its wide range of applications, including video games, special effects, and virtual try-on. It is regarded as a challenging task due to the model complexity of the cloth and the ability to perform irregular cloth deformations. Furthermore, many applications require interactive performance on commodity hardware, including mobile devices. This problem has been extensively studied in the literature. In order to achieve high-quality and reliable results,

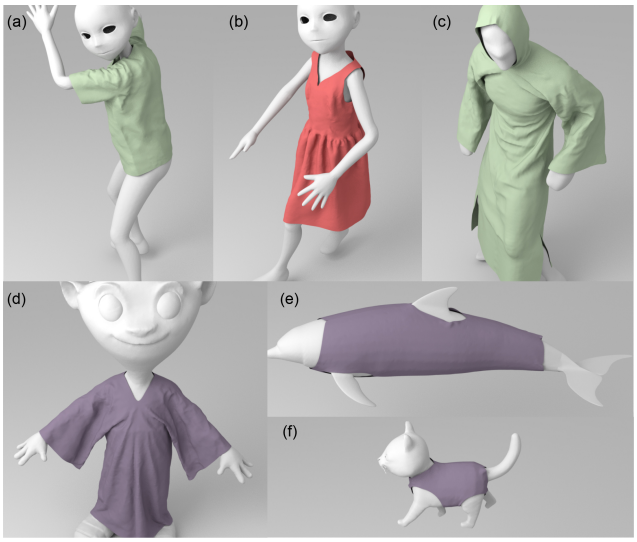

Figure 1. Given a skeleton-based representation of a character corresponding to target poses and different types of cloth (loose or tight-fitting), we use a two-stream skinning network to predict the cloth deformation for the target character. (a) and (b) correspond to the same human character with tight and loose-fitting clothing, respectively; (c) is a different human character wearing a long robe. Our network can also handle non-human characters such as a monster (d), a dolphin (e), or even a cat (f).

many efficient techniques based on physics-based simulation (PBS) have been proposed [1, 22, 4, 5, 9, 29, 31]. In these methods, the underlying cloth is modeled as a 3D surface mesh subdivided into finite contiguous triangles, and they use collision handling methods to generate accurate simulations. However, these methods cannot provide real-time frame rates for interactive applications.

There has been considerable work on using machine learning methods to significantly reduce the computational cost of predicting cloth deformation. Many learning-based networks [20, 3, 23] have been proposed for SMPL-based parametric 3D human models [16]. These SMPL-based methods are used to generate smooth deformations for humans moving with tight-fitting clothes. The prediction is generated in real-time because of the small number of parameters used in SMPL-based networks. However, the SMPL-based model is limited and cannot be used on arbitrary objects or characters used in games. In order to handle more general characters and enhance the quality of prediction, other algorithms use multi-layer perceptron (MLP) models on the vertices of the cloth mesh to learn the deformation [33]. Without using the topologies of a cloth mesh, such MLP-based method tends to train a network with a large number of parameters, which increases the memory overhead and the runtime cost. Recently, Graph Convolutional Networks (GCN) have been used to predict the cloth draping results on the human characters [8, 7, 32]. In practice, these methods need the pre-deformed cloth for the target pose [8, 7] or can only process the draping results on human characters in a T-pose [32].

In this paper, we deal with skeleton-based characters, which are widely used in computer games and other interactive applications. These include human-like characters (such as leading roles), monster characters similar to humans (such as trolls), and animal characters (such as pets). All these different characters can wear different types of clothes. We propose a learning-based cloth skinning model to capture the coarse and wrinkle features to obtain the final cloth deformation. Our approach is general and designed for all types of skeleton-based characters, including humans and animals. Furthermore, these characters can be dressed with loose or tight-fitting clothes.

Our formulation models the cloth draping deformation as the skinning of the cloth template at a canonical pose (such as a T-pose or a A-pose). Given the skeleton information and mesh information of the posed character, the deformation of the cloth is computed by skinning weights and the template cloth mesh. In order to handle different skinning characters and cloth, we design a novel two-stream network architecture to learn the residual positions of vertices of the cloth template mesh. It consists of a mesh-based residual stream and a skeleton-based residual stream. The skeleton-based residual stream is trained to obtain the coarse residual on the cloth template mesh, while the mesh-based residual stream is trained for the wrinkle features. The prediction examples of our two-stream skinning network are as show in 1.

We qualitatively and quantitatively analyze the performance of the proposed two-stream skinning network in a variety of scenarios. These include human-like characters and other characters. We validate our two-stream network thorough the ablation experiments. Compared with recent methods, our two-stream network can capture the fine details of the cloth deformation.

The novel components of our work include:

- **A learning-based cloth skinning model:** Our approach models the cloth deformation as the learning-based skinning of the template cloth mesh. Our skinning model is not limited to humans and can process many skinning characters.
- **Two-stream skinning network architecture for cloth deformation prediction:** Based on the learning-based cloth skinning model, we design a novel two-stream network architecture for cloth deformation prediction. The architecture consists of a mesh-based residual stream which is trained for wrinkle features, and a skeleton-based residual stream which is trained for coarse features.
- **Ability to process different types of clothes and characters:** Our network can process various types of characters and clothes. These characters and clothes can vary considerably.

We show the prediction results of our proposed skinning-based network on different human characters, non-human characters with different cloth types in Section 5. We compare our method qualitatively and quantitatively with other methods in Section 6. We can predict deformed clothes at averagely 7 milliseconds on an NVIDIA GeForce RTX 3090 GPU. As compared with prior approaches, our method can predict the deformation results with fine wrinkles and details.

2. Related Work

In this section, we give an overview of cloth deformation prediction using traditional PBS methods and recent learning-based methods. Many learning-based methods are limited to the SMPL model; we describe these methods in Section 2.2 and highlight other learning methods in Section 2.3.

2.1. Physics-based Simulation

PBS methods for generating deformed cloth are commonly based on the pipeline of time integration [1], collision detection [5, 29], and collision response [5, 9, 31].

While they can accurately model the deformation and result in non-penetrating simulations, the running time is not fast enough for interactive applications. To accelerate the simulation, recent research tends to use GPU-based algorithms to parallel the pipeline [30, 13]. However, current methods can simulate each frame in hundreds of milliseconds on high-end desktop GPUs. Moreover, the performance of these simulators depends on various parameters, such as material attributes, which are hard to fine tune.

2.2. SMPL-based Learning Algorithm

Many learning methods have been proposed based on SMPL-based parametric 3D human models. [16] proposed parametric skinning human models using SMPL, where the deformation of the human body mesh is driven by the skinning skeleton of the template body mesh. [20] regard the cloth mesh as the sub-mesh of the SMPL body mesh, and use an indicator matrix to select the associated vertices on the body mesh as the initial state. The proposed network, TailorNet [20], is trained as an increment from the initial state to represent the template cloth mesh. This is used to perform skinning operations to obtain the final deformation on the target pose. [14] use the skinning body mesh directly on the target pose as the initial state and learn a graph-attention-based network to predict the residual between the initial state and the final deformed cloth mesh with wrinkles. These methods use the vertices on the unposed template body mesh or posed target body mesh as the initial state of the deformed cloth mesh and train different networks to fit the residuals of the ground truth. Therefore, the predictions of these methods may not generate plausible results on some loose-fitting clothes such as dresses, because the vertices may be far away from the body mesh.

Other algorithms have been proposed that treat the cloth mesh deformation as a skinning deformation similar to the body mesh skinning [11, 16]. These methods tend to build a skinning model for cloth deformation from the canonical template cloth mesh. [23] use a garment fit regressor and a garment wrinkle regressor to learn the nonlinear residuals of the ground truth from the canonical cloth mesh. To enhance the performance on loose-fitting clothes, [25] smoothly diffuse the skinning parameters of neighbors for each vertex on the unposed cloth mesh. They propose an optimization-based strategy to project ground-truth garments to the canonical space without introducing collisions. However, the diffusion of the skinning parameters is only operated on the unposed canonical cloth, which makes the improvement of the predictions on the loose-fitting clothes limited. [3] use GCN to extract features on the unposed canonical cloth mesh to learn the blend weights. These methods ignore the impact of the poses on the skinning weight parameters. In practice, all these networks are constrained by the pose and shape parameters of SMPL.

2.3. Learning-based Cloth Deformation

Many learning-based methods have been proposed for general cloth meshes and characters that are not limited to SMPL-based representations. [8, 7] use dual quaternion skinning (DQS) [11] to generate the pre-deformation of the cloth template from the canonical pose and use GCN blocks to learn the residuals from the pre-deformation to the ground truth cloth mesh. [10] use the PCA to obtain the subspace of the cloth and the obstacle and use MLP to regress the non-linearity in subspace deformation. Unfortunately, using the previous predictions as the input of the subsequent predictions will accumulate the error and hinder the quality of the result. [33] only use the vertex coordinate of the cloth mesh to learn a cloth descriptor that can be fused with motion in latent space. Considering the difficulty of predicting the cloth deformation caused by body pose, [32] use an encoder and decoder architecture with GCN to learn the draping effect of different cloth types on the canonical pose. Other methods are designed for general triangle mesh-based obstacles [10, 15].

Many techniques have been proposed to estimate a collision-free subspace of general 3D deformable models and used to compute collision-free cloth configurations [27, 28]. For human-like characters, many learning methods [8, 2] use collision loss to penalize penetrated garment-body pairs during training. Our approach for handling arbitrary characters and clothing types is complimentary and can be combined with these methods.

3. CTSN: Our Approach

Our approach takes a skeleton-based character of the target pose and cloth template of the canonical pose as input and predicts cloth mesh deformation for the target pose character through a skinning-based network. The skeleton-based character of the target pose has the skinned mesh and the transformation information of the joints. The key concept of our approach is a novel skinning-based cloth model. We propose a network architecture composed of two residual networks based the cloth model. We present the details of our skinning-based cloth model and the network architecture in following sections.

3.1. Skinning-based Cloth Model

3.1.1 Skinning-based Character Model

Our skinning-based cloth model is inspired by the skinning-based character model, SMPL [16]. We give a brief overview of the SMPL model and the symbols used in the rest of the paper.

In the standard skeletal rigging, the posed character is calculated by the following formula:

$$M_B(\gamma) = \mathbf{W}\left(T_B, J, \gamma, W_B\right) \tag{1}$$

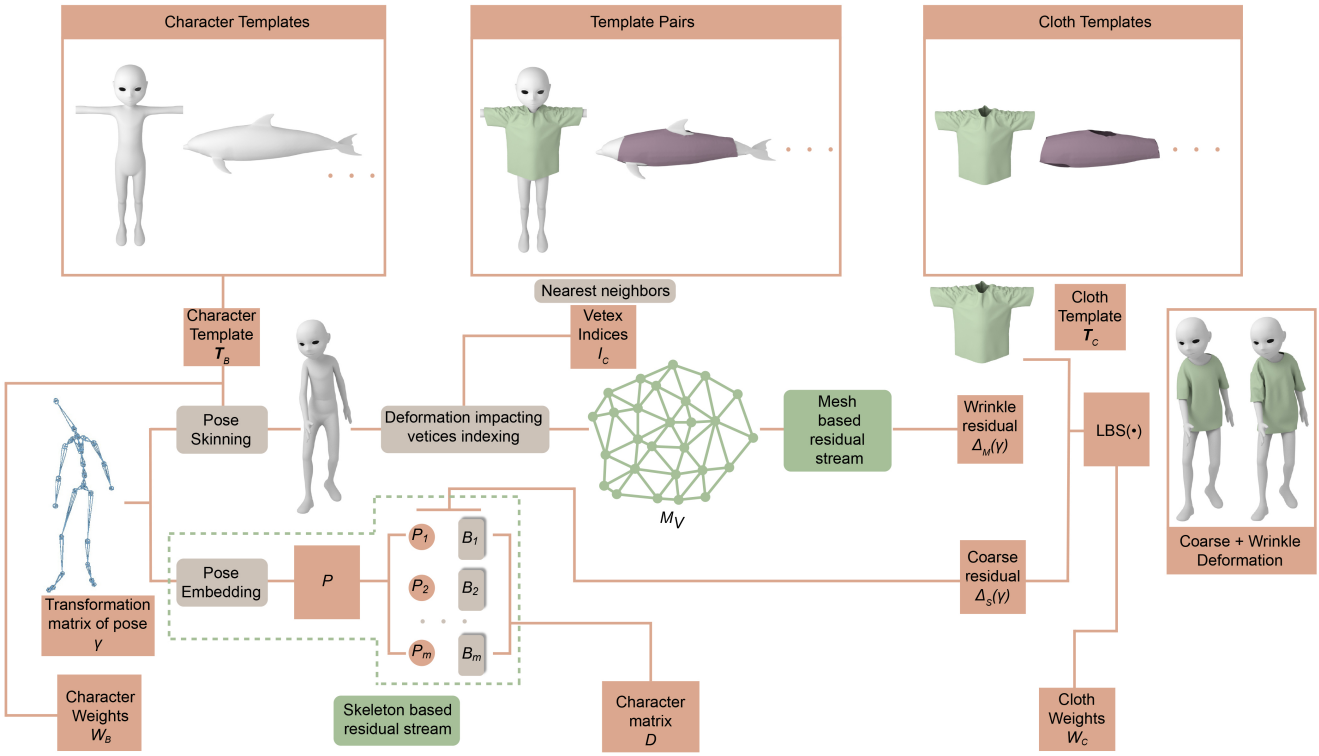

Figure 2. Our network architecture is composed of the mesh-based residual stream and the skeleton-based residual stream (shown as the green blocks) to obtain the wrinkle residual $\Delta_M(\gamma)$ and the coarse residual $\Delta_S(\gamma)$. γ is the transformation matrix of the target pose. The updated cloth template mesh $T_C(\gamma)$ is used by the skinning operation to obtain the final deformed cloth mesh $M_C(\gamma)$.

where $M_B(\gamma)$ is the posed character mesh; T_B is the template character mesh at the canonical pose; J is the skeleton of character; γ is the transformation matrix of the character joints; W_B is the skinning weight matrix; and $\mathbf{W}(\cdot)$ is the skinning function. The parametric skinning human model SMPL [16] uses a set of orthonormal principal components of shape and pose displacements to capture the soft-tissue dynamics. This model is represented as:

$$\begin{aligned} M_B(\beta,\theta) &= \mathbf{W}\left(T_B(\beta,\theta), J(\beta), \theta, W_B\right) \\ T_B(\beta,\theta) &= \mathbf{T}_B + B_S(\beta) + B_P(\theta) \end{aligned} \tag{2}$$

where β and θ are the shape coefficients and the pose vector, which contains the transformation information of the joints, respectively. $J(\beta)$ is the skeleton position with shape coefficients β. $T_B(\beta,\theta)$ is the template human mesh, which is the function of β and θ. To capture the soft-tissue dynamics, body shape blend offsets $B_S(\beta)$ and pose blend shapes $B_P(\theta)$ are fused to the initial template human body mesh $\mathbf{T}_B$ to generate the final template human mesh $T_B(\beta,\theta)$.

3.1.2 Our Two-stream Skinning-based Cloth Model

Cloth deformation is driven by the character motion since cloth is dressed on the surface of a character mesh. To simplify the deformation problem, we use a skinning-based model for the template cloth mesh to guide the deformation. Inspired by the SMPL model and other approaches [23], we present a new method to build a skinning based model for cloth deformation. Thus, given a template cloth mesh $\mathbf{T_C}$ at the canonical pose and the skeleton transformation matrix at the target poseγ, the deformed cloth $M_C(\gamma)$ is defined as follows:

$$\begin{aligned} M_C(\gamma) &= \mathbf{W}\left(T_C(\gamma), J, \gamma, W_C\right), \\ T_C(\gamma) &= \mathbf{T_C} + \Delta_S(\gamma) + \Delta_M(\gamma), \end{aligned} \tag{3}$$

where γ is the transformation matrix of the joints of the target character body. W_C is the skinning weight matrix for cloth template mesh $\mathbf{T_C}$. For the skinning function $\mathbf{W}(\cdot)$, LBS$(\cdot)$ represents the linear blend skinning (LBS) method [17], which is widely supported by game engines. $T_C(\gamma)$ is the optimized template cloth mesh at the canonical pose. $\Delta_S(\gamma)$ is the skeleton-based residual positions trained to obtain the coarse features. $\Delta_M(\gamma)$ is the mesh-based residual positions trained for adding wrinkle details to the coarse prediction. We highlight our two-stream network architecture in Fig. 2.

Our network architecture consists of a mesh-based residual stream and a skeleton-based residual stream. The mesh-based residual stream is designed to compute the impact of the nearest vertices of the cloth on the posed character

mesh on the cloth template mesh, i.e. $\Delta_M(\gamma)$, while the skeleton-based residual stream is used to model the influence of skeleton information of the character to the cloth template mesh, i.e. $\Delta_S(\gamma)$. Since the cloth type can be tight or loose, we train the skinning weight matrix W_C for different types of cloth. We present more details in Section 3.2, 3.3, and 3.4. In general, our network architecture can be expressed as:

$$M_C(\gamma) = \mathcal{N}_\sigma\left(\mathbf{T}_C, \mathbf{T}_B, J, \gamma, W_B, W_C\right), \tag{4}$$

where $\mathcal{N}_\sigma$ is the skinning-based network and σ represents the trainable parameters.

Similar to TailorNet [20], we decompose the deformed cloth mesh to the low-frequency and the high-frequency deformations. To obtain the low-frequency of the cloth mesh, we perform the Laplacian smoothing to the simulated cloth mesh. The high-frequency deformation is residual wrinkle details.

3.2. Skeleton-based Residual Stream

In our skeleton-based residual stream, the input is the transformation matrix γ of character joints at the target pose. We pass the transform matrix γ into the pose embedding network, which is composed of an MLP, to learn the pose embedding $\mathcal{P} = \{P_1, P_2, P_2, \cdots, P_m\}$, where m is the size of the embedding vector $\mathcal{P}$:

$$\mathcal{P} = \Phi(\gamma), \tag{5}$$

where $\Phi(\cdot)$ is the MLP-based pose embedding network.

After the pose embedding, our goal is to learn a set of character residual matrices $D = \{B_1, B_2, B_3, \cdots B_m\}$ for the character and cloth pair. As for matrix B_j, where $j \in \{1, 2, \cdots, m\}$, B_j can be expressed as:

$$B_j = \begin{bmatrix} b_{00} & \cdots & b_{02} \\ \vdots & \ddots & \vdots \\ b_{n0} & \cdots & b_{n2} \end{bmatrix}, \tag{6}$$

where $b_{00}, \cdots, b_{02}, \cdots, b_{n0}, \cdots, b_{n2}$ are trainable for the target character and cloth. n is the number of vertices of the template cloth mesh.

Finally, the pose embedding $\mathcal{P}$ is fused as the weights to the residual matrix D to obtain the skeleton-based residual component $\Delta_S(\gamma)$:

$$\Delta_S(\gamma) = \sum_{j=0}^{j=m} P_j B_j \tag{7}$$

To train the skeleton-based residual stream to obtain the coarse features, we use the obtained low-frequency deformation as the ground truth.

3.3. Mesh-based Residual Stream

The skeleton-based residual stream can only predict the position offset $\Delta_S(\gamma)$, which captures the coarse features of the target deformation. The prediction results of the skeleton-based residual stream are smooth. To improve the prediction, we use a mesh-based residual stream to learn the wrinkle residual for the final cloth deformation.

We build a KD-tree for the template cloth mesh and the body mesh at canonical pose. We use this tree data structure to find the nearest point index I_C on the body mesh for each vertex on the cloth mesh. Given the input transform matrix of the skeleton of the body, we can obtain the skinned body mesh at the target pose by using our skinning method. We obtain the positions $\mathcal{V}$ of the nearest points through the selected index I_C. In order to improve the effectiveness of our mesh-based residual stream, we also build the reference mesh graph $\mathcal{M}_\mathcal{V} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, where $\mathcal{V}$ corresponds to the nearest vertices computed previously as the nodes of the graph $\mathcal{M}_\mathcal{V}$; $\mathcal{E} \subseteq V \times V$ corresponds to the edges of the template cloth mesh, and $\mathcal{A}$ is the $(0, 1)$ adjacency matrix that highlights the connectivity of the vertices $\mathcal{V}$.

We use the Graph Transformer network [26] to extract features on the predefined constructed mesh graph $\mathcal{M}_\mathcal{V}$. The architecture of the mesh-based residual stream is illustrated in Fig. 3.

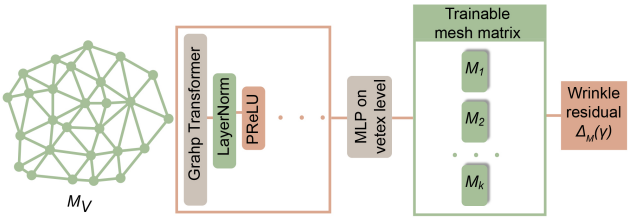

Figure 3. The architecture of our mesh-based residual stream. We use Transformer Graph Convolutional Network to extract features of the reference mesh graph $\mathcal{M}_\mathcal{V} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$. The extracted features are transmitted to vertex level MLP layers and trainable mesh matrices to obtain the wrinkle residual.

In the Graph Transformer layers of our mesh-based residual network, we define $H^{(l)} = \left\{h_1^{(l)}, h_2^{(l)}, \ldots, h_n^{(l)}\right\}$ as the node features of previous layer l, where n is the number of nodes. $h_i^l \in \mathbb{R}^F$ represents the features of node i in layer l whose dimension is F. h_j^l represents the features of node j in layer l, where node j is the neighbor of node i. The multi-head attention features $f_{c,ij}^{(l)}$ of head c from node j to node i are computed as follows:

$$\begin{aligned} q_{c,i}^{(l)} &= W_{c,q}^{(l)} h_i^{(l)} + b_{c,q}^{(l)} \\ k_{c,j}^{(l)} &= W_{c,k}^{(l)} h_j^{(l)} + b_{c,k}^{(l)} \\ e_{c,ij} &= W_{c,e} e_{ij} + b_{c,e} \\ f_{c,ij}^{(l)} &= (q_{c,i}^{(l)})^\top (k_{c,j}^{(l)} + e_{c,ij}) \end{aligned} \tag{8}$$

where $W_{c,q}^{(l)}$, $W_{c,k}^{(l)}$, $W_{c,e}$, $b_{c,q}^{(l)}$, $b_{c,k}^{(l)}$, and $b_{c,e}$ are trainable parameters. e_{ij} represents the edge features.

After normalization, the multi-head attention coefficients $\alpha_{c,ij}^{(l)}$ of head c from node j to node i are computed as:

$$\alpha_{c,ij}^{(l)} = \text{softmax}\left(\frac{f_{c,ij}^{(l)}}{\sqrt{d}}\right) \tag{9}$$

where d is the hidden size of each head. The output features $\hat{h}^{(l+1)}$ of the node i in layer $l+1$ are calculated by the following formula:

$$\begin{aligned} v_{c,j}^{(l)} &= W_{c,v}^{(l)} h_j^{(l)} + b_{c,v}^{(l)} \\ \hat{h}^{(l+1)} &= \|_{c=1}^{C} \left[\sum_{j \in \mathcal{N}(i)} a_{c,ij}^{(l)} \left(v_{c,v}^{(l)} + e_{c,ij} \right) \right] \end{aligned} \tag{10}$$

where C is the number of the head. $W_{c,v}^{(l)}$ and $b_{c,v}^{(l)}$ are trainable parameters. $\mathcal{N}(i)$ is the neighbors of the node i. $\|$ is the concatenation operation for C head attention.

In order to improve the ability of the feature extraction, $\beta_i^{(l)}$ is calculated as follows:

$$\begin{aligned} r_i^{(l)} &= W_r^{(l)} h_i^{(l)} + b_r^{(l)} \\ g_i^{(l)} &= W_g^{(l)} \left[\hat{h}_i^{(l+1)}; r_i^{(l)}; \hat{h}^{(l+1)} - r_i^{(l)} \right] \\ \beta_i^{(l)} &= \text{sigmoid}\left(g_i^{(l)} \right) \end{aligned} \tag{11}$$

Thus, the final output features of the node i in layer $l+1$ are updated as:

$$\begin{aligned} r_i^{(l+1)} &= \left(1 - \beta_i^{(l)}\right) \hat{h}_i^{(l+1)} + \beta_i^{(l)} \left(W_r^{(l)} h_i^{(l)} + b_r^{(l)} \right) \\ h_i^{(l+1)} &= \text{ReLU}\left(\text{LayerNorm}\left(r_i^{(l+1)} \right) \right). \end{aligned} \tag{12}$$

As shown in Fig. 3, we use the Graph Transformer network to extract features of the mesh graph. After the feature extraction on the mesh graph, we use a vertex level MLP and a set of trainable mesh matrices to obtain the wrinkle residual positions. The trainable mesh matrices are represented as $\{M_1, M_2, M_3, \cdots M_k\}$. $\text{ReLU}(\cdot)$ is used to match the nonlinearity of the high-frequency deformation. $\Delta_M(\gamma)$ is computed from the mesh graph $\mathcal{M}_\mathcal{V}$ as:

$$\Delta_M(\gamma) = \Psi(\mathcal{M}_\mathcal{V}), \tag{13}$$

where $\Psi(\cdot)$ represents the mesh-based residual stream. Similar to the skeleton-based residual stream, we use the high-frequency deformation as the ground truth to train the mesh-based residual stream.

3.4. Skinning Operation

After obtaining the skeleton-based residual component Δ_S and the mesh-based residual component Δ_M, we compose a new optimized template cloth mesh $T_C(\gamma)$.

To solve the impact of cloth types (tight-fitting or loose-fitting) on the final prediction results, we learn a weight residual ΔW_C for different cloth types. ΔW_C is represented as:

$$\Delta W_C = \begin{bmatrix} w_{00} & \cdots & w_{0k} \\ \vdots & \ddots & \vdots \\ w_{n_0} & \cdots & w_{nk} \end{bmatrix} \tag{14}$$

where $w_{00}, \cdots, w_{nk}$ are trainable parameters and k is the maximum number of joints.

The fusion skinning weight matrix is generated as:

$$W_C = W_C^I + \Delta W_C, \tag{15}$$

where W_C^I represents the initial cloth weight obtained from the template body skinning weight W_B through KD-tree.

In general, pose embedding function $\Psi(\cdot)$ and D are trained by skeleton-based residual stream for the coarse deformation, while $\Psi(\mathcal{M}_\mathcal{V})$ is trained by mesh-based residual stream for the wrinkle deformation. W_C is trained for processing different types of cloth.

3.5. Loss Function

To optimize the parameters of our network architecture, we use the following loss function to minimize the difference between the predicted deformed cloth mesh and the ground truth:

$$\mathcal{L} = \frac{1}{\sum_{i=1}^{b} N} \sum_{i=1}^{b} \sum_{j=1}^{N} \left\| x_p^j - x_g^j \right\|_2, \tag{16}$$

where x_p^j is the predicted position of vertex j on the deformed cloth mesh M_{CP}. x_g^j is the position of vertex j on the ground truth cloth mesh M_{CG}. N is the number of vertices of cloth mesh M_{CG}. $\|\cdots\|_2$ is the L_2 distance. b is the batch size.

4. Dataset and Implementation

In this section, we describe the generation of our dataset and some implementation details.

4.1. Dataset

We have generated many different characters and clothing types to validate our network architecture (as shown in Fig. 4). We upload the character meshes of Andy and Qman in canonical poses, such as a T-pose, to the motion capture website Mixamo[1]. We download many character poses computed from that website as FBX files. To eliminate the absoluteness of the vertex position and make it easy to train our network, we move the hip joint of the character mesh to

[1] https://www.mixamo.com/

No.	Character	Cloth	Cloth faces	Character faces	Joints	Sample number
1	Andy	Dress	15176	12999	25	15040
2	Andy	Tshirt	12392	12999	25	15040
3	QMan	Tshirt	16148	14664	25	12480
4	QMan	Robe	19168	14664	25	12480
5	Monster	Robe	13637	20112	25	485
6	Dolphin	Cloth	7878	3212	11	93
7	Cat	Cloth	3107	2636	25	103

Figure 4. The attributes of different characters and clothing types used for our evaluation. We obtain different poses of characters from the Mixamo website. We extract the transformation matrix and skinning weight from the motion files. We use the cloth simulator ARCSim to precompute the deformed cloth mesh for training.

the origin of the coordinates. Next, we extract the transformation matrix γ of the character at different poses and the skinning weights W_B from the FBX files.

After extracting of the motion files, we use the skinning operation to obtain the character meshes at different poses with transformation matrix of joint γ. We use different clothing types such as a T-shirt, dress, and robe. The T-shirt is tight-fitting, and the dress and robe are loose and can result in complex deformations. In order to compute the ground truth of the deformed cloth, we use the physics-based simulator ArcSim [19, 18, 21] to simulate the cloth. During the simulation, we perform linear interpolation between the adjacent poses and relax the cloth mesh to compute the quasi-static deformation.

To evaluate that our network can process more complex and different characters, we applied our network on non-human characters such as a monster, a dolphin, and a cat. The monster character has a skeleton similar to the human character, while the dolphin and the cat have different skeletons. The dolphin character has no leg joints, while the cat model has four legs without hands. We can also simulate the cloth deformation on these characters. The monster character wears a loose robe, and the dolphin and the cat wear tight-fitting clothes designed for these characters.

The attributes of the skinned character and cloth meshes are shown in Fig. 4. We have highlighted the number of triangles of each character mesh and cloth mesh, the number of joints of the character, and the number of samples used by our algorithm.

4.2. Network Implementation and Training

We train our network on a standard PC (Ubuntu 20.04 LTS/Intel I7 CPU@4.2G Hz/8G RAM, NVIDIA GeForce RTX 3090 GPU). Our network is implemented using PyTorch 1.7.0 and Python 3.8.8.

Following [20] and [33], we also split our dataset for training and testing. For the motion clips obtained from Mixamo, we split 90% motion clips as training data and the last 10% motion clips as the test data, which are unseen during training.

We train our network on the dataset containing different characters and cloth types. As shown in Fig. 4, our dataset has 5 skeleton-based characters (2 human characters and 3 non-human characters) with 7 different types of cloth. During training, we set the learning rate at $1e-3$ and use an Adam optimizer [12] to train the parameters of the neural network.

4.3. Penetration Handling

It is hard to obtain collision-free predictions or configurations with learning-based methods on the test data, which is unseen during training. We use a method similar to [33] to reduce the penetrations between the cloth and the character. After the prediction, the predicted deformed cloth mesh is optimized by minimizing the following function to avoid penetrations between the cloth and the character:

$$E_B = \sum_{i \in V_{pene}} \left\| v_i - \left(v_i^B + \epsilon n_i^B \right) \right\|, \tag{17}$$

where V_{pene} is the set of penetrated vertices of predicted cloth. For each penetrated vertex v_i, the closest point vertex v_i^B and normal n_i^B are computed over the character mesh. E_B is the error between penetration vertices on the cloth and the character mesh. and ϵ is a small step to pull out the penetrated vertices from the character mesh. During the optimization process, the positions of V_{pene} are updated, which reduces the number of localized penetrations or collisions.

5. Results

In this section, we highlight the deformation prediction results of our network on the unseen test data. We compare our predictions on the unseen test data with the ground truth results obtained using a physics-based simulator (ArcSim).

5.1. Predicted Deformation using Our Network

Fig. 5 shows the predicted T-shirt deformation at different poses for the character Andy. Our predictions show the fine details with wrinkles, similar to those in the ground truth deformation. We also show the prediction results of other types of cloth and another character, Qman, in Fig. 6. Fig. 6 (a) shows the predicted deformation of the dress on the character Andy, while Fig. 6 (b) and (c) show the cloth deformation on the other character, Qman. The dress on the character Andy in Fig. 6 (a) and the robe on the character Qman in Fig. 6 (c) are both loose-fitting types of clothing. These predictions validate the effectiveness of our network. Since we train the mesh-based residual stream and skinning weight for each clothing type, the deformation details can be easily captured, enhancing the predictions.

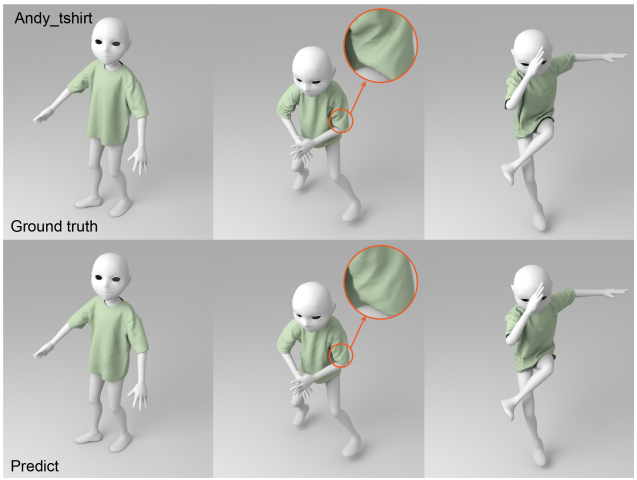

Figure 5. The predicted deformed T-shirt dressed on the character Andy in different poses. All the input poses are unseen during the network training. The top row shows the ground truth of the deformation, while the bottom row highlights the predictions of our network. We also highlight the fine details and folds in the zoomed images.

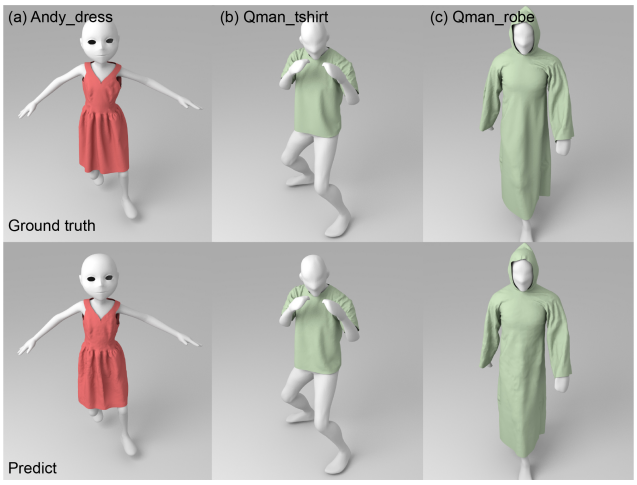

Figure 6. The predicted deformed cloth on other human characters. The first column shows the prediction on the character Andy. The middle and last columns show the deformation predictions on the character Qman.

Our network can also process other non-human characters with skeletons. The predicted results of our network and the ground truth on the non-human characters are shown in Fig. 7. Fig. 7 (a) shows the result of our network on a non-human character, Monster. The skeleton hierarchy of Monster is similar to the human characters in Fig. 6. To show the complex characters that our network can process, we highlight the results of our network on the Dolphin character in Fig. 7 (b) and the Cat character in Fig. 7 (c). The Dolphin has no arm joints or legs joints, while the Cat has four legs without arms. The cloth on the Dolphin

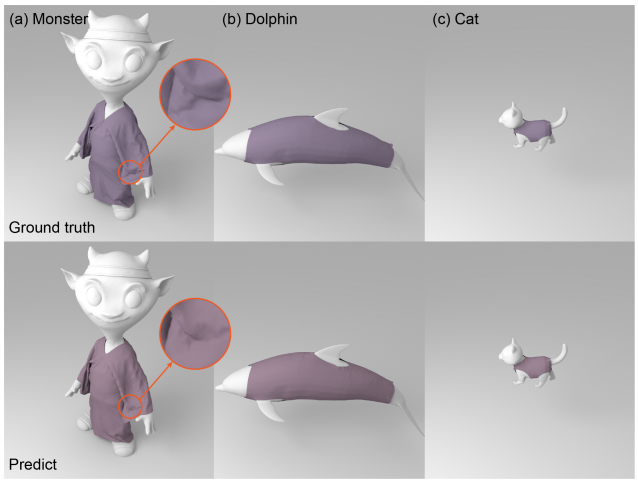

Figure 7. The results of our network on non-human characters.The first column shows the deformed robe on the Monster, whose skeleton is similar to that of human characters. The middle column shows the deformed cloth on the Dolphin, which has no legs. The last column shows the cloth on the Cat, which has no arms.

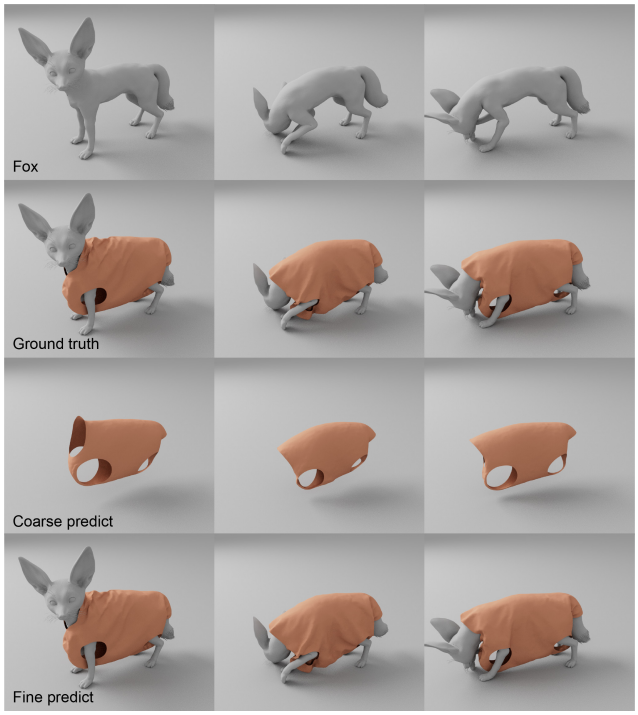

Figure 8. The results of our network on non-human character fox. There is a loose-fitting cloth dressed on the character fox. The first row shows the target pose of the character fox. The second row shows the ground truth. The third row shows the coarse prediction. The last row shows the fine detailed prediction.

and the Cat are designed specifically for these characters. The results of our network show the fine predictions of the cloth deformations on these non-human characters. As for the non-human characters, the deformation of loose-fitting

cloth is also well predicted. Fig. 8 shows the loose-fitting cloth dressed on the character Fox. Our network can predict the coarse deformed cloth and the fine detailed one. The results of cloth deformation on the Dolphin, the Cat and the Fox show the capability of our network processing non-human characters. The prediction of deformation can catch the fine wrinkle details.

Fig. 9 shows the result of penetration handling described in Sec.4.3. After post-processing, the penetration between the back of the character fox and the dressed cloth is eliminated and the penetration-free result is obtained.

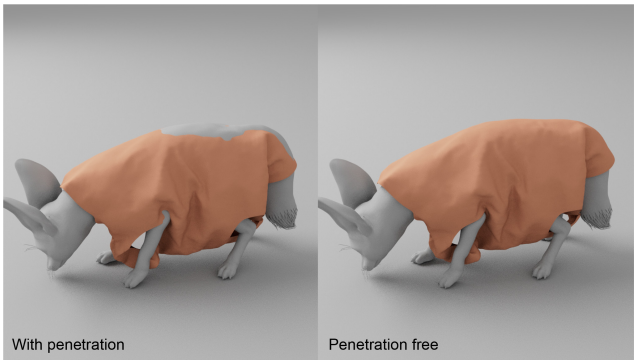

Figure 9. The results of penetration Handling. The left is the situation of penetration between the character fox and the loose-fitting cloth. The right is the penetration-free result.

5.2. Prediction Runtime

We can perform cloth deformation prediction with our network both on GPUs and CPUs. We have highlighted the runtime of predicting a single cloth mesh in Table. 1. The runtime for a GPU is collected on an NVIDIA GeForce RTX 3090 GPU. The runtime for a CPU is collected on an Intel I7 CPU. As shown by the table, we can perform a single prediction within 7ms on a GPU, which is much faster than prior learning-based [15] or physically-based algorithms [13]. The running time of our deformation prediction algorithm on CPU in less than 0.2s.

Methods	CPU run time (s)	GPU run time (s)
ARCSim	3.45	/
I-Cloth	/	5.12E-2
PBNS	9.55E-2	7.12E-2
DeePSD	3.12E-1	1.25E-1
Our Method	1.72E-1	7.032E-3

Table 1. The average CPU and GPU runtime for a single cloth mesh prediction.

6. Comparisons

In this section, we qualitatively and quantitatively compare the results of our network with prior learning-based methods. We also perform some ablation experiments to validate the effectiveness of our network.

6.1. Comparisons with Prior Learning Methods

Many approaches have been proposed to predict cloth deformations using learning-based networks. We have highlighted many recent methods and their attributes in terms of handling different kinds of characters and clothing types in Table 2. Some methods [20, 3, 2, 6] are based on the SMPL model, which limits them to only processing SMPL human bodies. [23, 25] are also based on the SMPL model. However, it is possible to extend them to remove the dependence on SMPL-based representation. Therefore, we modify these two methods and compare their results with our method in the following sections. [10] uses PCA to extract the principal components of the character vertices and cloth vertices to learn the relationship with the next deformation in the subspace. However, this method uses the previous prediction as the input for subsequent predictions and may result in accumulated errors. [8, 7] use DQS [11] to pre-deform the cloth mesh from the canonical pose to the target pose and then use a learning-based network to predict the residual of the pre-deformed cloth mesh and ground truth. This method only works well on tight-fitting cloth, and its predictions tend to be smooth and may lose wrinkle details. [33] use MLP to learn the intrinsic features for cloth vertices and character vertices, which results in a model with many redundant parameters. Furthermore, these methods are mostly limited to one or many specific characters or clothing types. In contrast, our network can overcome these limitations and is more general.

6.2. Qualitative Comparisons

We have implemented the modified versions of [3] and [2] to process the non-SMPL characters. We replace the SMPL skinning method with a character skinning method, which is based on using skeletons. The modified version of [2] is an unsupervised method. [3] contains the supervised part and the unsupervised part in its network. We have compared our network with the supervised part of [3].

Fig. 10 shows the comparison between the prediction of PBNS [2], DeePSD [3], and our method. As shown in Fig. 10, the PBNS method [2] tends to predict the deformed cloth mesh, which is tightly wrapped on the character and can introduce artifacts in the deformation. The DeePSD method [3] tends to predict smooth deformations, resulting in penetrations with the character even after post-processing. This implies that the prediction of DeePSD [3] is driven less by the transformation matrix of the character. In contrast, the results of our method tend to generate deformations with fine wrinkles. We have also implemented the learning algorithm [15] and obtained similar results with our method. The prediction results of [15] can also gener-

Network for coth deformation	SMPL human	Non-SMPL human	Non-human skinned animal	Rigid obstacle	Static prediction	Single network
TailorNet[20]	✓	✗	✗	✗	✓	✗
DeePSD [3]	✓	✓	✓	✗	✓	✗
[23]	✓	✗	✗	✗	✗	✗
[25]	✓	✗	✗	✗	✗	✗
[10]	✓	✓	✓	✓	✗	✗
GarNet [8]	✓	✓	✗	✗	✓	✗
[33]	✓	✓	✗	✗	✓	✗
[2]	✓	✓	✓	✗	✓	✗
DRAPE [6]	✓	✗	✗	✗	✓	✗
N-Cloth [15]	✓	✓	✓	✓	✓	✗
Our method	✓	✓	✓	✗	✓	✓

Table 2. We compare the characteristics and features of our approach with prior methods. We highlight the unique capabilities of our approach.

ate fine wrinkles. However, [15] uses significantly higher memory footprint (about 928.8MB).

6.3. Quantitative Comparisons

We also perform quantitative comparisons between our method and previous methods. We use the following error metrics to evaluate the prediction results of our network and others.

$$\begin{aligned} \mathcal{E}_{dist} &= \frac{1}{N}\sum_{i=1}^{N}\left\|x_p^i - x_g^i\right\|, \\ \mathcal{E}_{norm} &= \frac{1}{N}\sum_{i=1}^{N}\arccos\left(\frac{(n_p^i)^T n_g^i}{\left\|n_p^i\right\|\left\|n_g^i\right\|}\right), \end{aligned} \tag{18}$$

where x_p^i is the position of vertex i of the predicted mesh P. x_g^i is the ground truth of vertex i. N is the number of vertices of the cloth mesh. n_p^i and n_g^i are the normal vectors of vertex i on the predicted mesh and the ground truth, respectively.

Evaluation	PBNS	DeePSD	Our Method
mean $\mathcal{E}_{dist}$(m)	7.350E-2	3.10E-2	1.08E-2
std $\mathcal{E}_{dist}$(m)	1.05E-2	6.05E-3	2.11E-3
mean $\mathcal{E}_{norm}(^\circ)$	42.44	31.72	9.12
std $\mathcal{E}_{norm}(^\circ)$	3.26	3.28	1.57

Table 3. We compare the mean and standard deviations of mesh errors on test samples based on the ground truth computed from physics-based simualtors.

The calculated error metrics are shown in Table 3. The results generated from our network are more accurate than PBNS [2] and DeePSD [3].

We also compare the memory footprint (i.e., number of parameters used) of different networks in Fig. 11 by measuring the model size. Compared with [15], whose memory footprint is 928.8MB, the memory footprint of our method is much less (36.5MB). The memory footprint of DeePSD is 3.22MB, and PBNS is 30.4 MB.

6.4. Ablation Experiments

To validate the effectiveness of our network architecture, we implement a series of ablation experiments. Fig.12 shows the results of the modified network without some parts of the overall architecture. Fig. 12 (a) is the ground truth of the deformed cloth. Fig. 12 (b) is the cloth skinning deformation only with the fixed initial skinning weight. With the fixed skinning weight, there are artifacts on the skinning deformation, such as legs and belly. Fig. 12 (c) is the result with the skeleton-based residual stream and trainable cloth skinning weight. The deformation in Fig. 12 (c) tends to obtain the coarse residual. Fig. 12 (d) is the result of our full network architecture with the skeleton-based residual stream, the mesh-based residual stream and trainable cloth skinning weight. Compared with the result of Fig. 12 (c), Fig. 12 (d) shows that our mesh-based residual stream can capture the fine details of the final deformation. Fig. 12 (e) is the result of our network without the trainable cloth skinning weight. Without the trainable skinning weight, the skinning result tends to predict more artifacts. There are folds on the legs similar with the result of Fig. 12 (b).

The parameter m for the skeleton-based residual stream and k for mesh-based residual stream also impacts the performance. We have used different values of m and k to train our network. With the increase in the value of m, the prediction of our network becomes more accurate. How-

Figure 10. Comparison of results between our network and previous methods. The first column is the ground truth of the deformed cloth. The second and third columns are the results of [2] and [3]. The last column is the result of our method. The top and bottom rows are the front and back views of the deformed predictions.

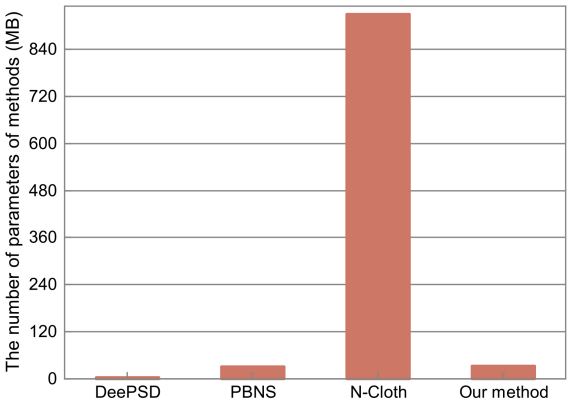

Figure 11. Our approach can is general in terms of handling all skeleton-based models and meshes, but has low memory overhead.

ever, the memory footprint also increases, which increases the model size of our network. We show the relevant memory footprint of our network on the scene of Qman dressing robe with $m = 5, 32, 100$ and $k = 64, 128, 256$, respectively in Table. 4. We choose $m = 32$ and $k = 128$ by experiments and find that increasing m and k does not obviously improve the results.

Modified network	Memory footprint (MB)
$m = 5, k = 64$	25.8
$m = 32, k = 128$	36.5
$m = 100, k = 256$	59.5

Table 4. Memory footprint with different m and k.

7. Conclusions, Limitations and Future Work

We present a two-stream skinning-based network to predict cloth deformation from a template cloth in a canonical pose. Our method can process different characters and cloth types retaining the fine details. Since our network is based on the skinning operation, the memory footprint of our method is low. The runtime performance of our network is fast, and we can predict a single cloth deformation in 7ms on a desktop GPU.

Our approach does have some limitations. Like prior learning-based methods, collision-free predictions are not guaranteed by our network. As part of future work, we would like to overcome the above limitations and extend our work to unsupervised networks [3] or self-supervised networks [24]. In addition, our method tend to train a specific model for each character due to the difference between human and non-human characters.

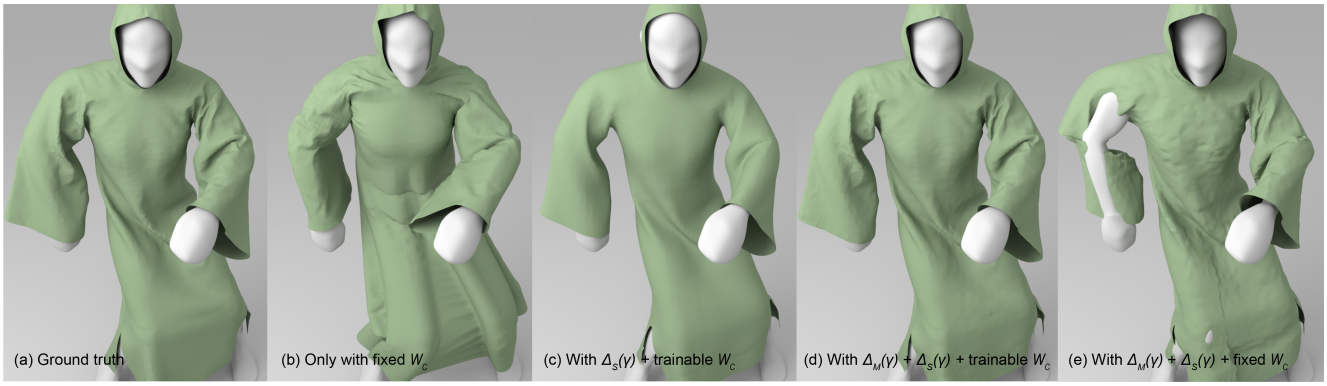

Figure 12. The ablation experiments of our network. We have disabled the mesh-based residual stream, the skeleton-based residual stream, and the trainable cloth weights in our method to show the benefits of each component of our architecture.

Acknowledgement

This work is supported in part by the National Natural Science Foundation of China under Grant No.: 61972341, Grant No.: 61972342, Grant No.: 61732015, and the Tencent-Zhejiang University joint laboratory.

References

[1] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, 1998. 2

[2] H. Bertiche, M. Madadi, and S. Escalera. Pbns: physically based neural simulation for unsupervised garment pose space deformation. *ACM Transactions on Graphics (TOG)*, 40(6):1–14, 2021. 3, 9, 10, 11

[3] H. Bertiche, M. Madadi, E. Tylson, and S. Escalera. Deepsd: Automatic deep skinning and pose space deformation for 3d garment animation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5471–5480, 2021. 2, 3, 9, 10, 11

[4] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly. Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph. (SIGGRAPH)*, 33(4):154:1–154:11, July 2014. 2

[5] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph.*, 21(3):594–603, 2002. 2

[6] P. Guan, L. Reiss, D. A. Hirshberg, A. Weiss, and M. J. Black. Drape: Dressing any person. *ACM Transactions on Graphics (TOG)*, 31(4):1–10, 2012. 9, 10

[7] E. Gundogdu, V. Constantin, S. Parashar, A. S. Banadkooki, M. Dang, M. Salzmann, and P. Fua. GarNet++: Improving fast and accurate static 3d cloth draping by curvature loss. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 2, 3, 9

[8] E. Gundogdu, V. Constantin, A. Seifoddini, M. Dang, M. Salzmann, and P. Fua. GarNet: A two-stream network for fast and accurate 3d cloth draping. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8739–8748, 2019. 2, 3, 9, 10

[9] D. Harmon, E. Vouga, R. Tamstorf, and E. Grinspun. Robust treatment of simultaneous collisions. *ACM Trans. Graph.*, 27(3):23:1–23:4, 2008. 2

[10] D. Holden, B. C. Duong, S. Datta, and D. Nowrouzezahrai. Subspace neural physics: Fast data-driven interactive simulation. In *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–12, 2019. 3, 9, 10

[11] L. Kavan, S. Collins, J. Zara, and C. O'Sullivan. Skinning with dual quaternions. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, I3D '07, page 39–46, New York, NY, USA, 2007. 3, 9

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[13] C. Li, M. Tang, R. Tong, M. Cai, J. Zhao, and D. Manocha. P-cloth: interactive complex cloth simulation on multi-gpu systems using dynamic matrix assembly and pipelined implicit integrators. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020. 3, 9

[14] T. Li, R. Shi, and T. Kanai. Detail-aware deep clothing animations infused with multi-source attributes. *arXiv preprint arXiv:2112.07974*, 2021. 3

[15] Y. Li, M. Tang, Y. Yang, Z. Huang, R. Tong, S. Yang, Y. Li, and D. Manocha. N-cloth: Predicting 3d cloth deformation with mesh-based networks. *arXiv preprint arXiv:2112.06397*, 2021. 3, 9, 10

[16] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015. 2, 3, 4

[17] N. Magnenat-thalmann, R. Laperrire, D. Thalmann, and U. D. Montréal. Joint-dependent local deformations for hand animation and object grasping. In *In Proceedings on Graphics interface '88*, pages 26–33, 1988. 4

[18] R. Narain, T. Pfaff, and J. F. O'Brien. Folding and crumpling adaptive sheets. *ACM Transactions on Graphics (TOG)*, 32(4):1–8, 2013. 7

[19] R. Narain, A. Samii, and J. F. O'brien. Adaptive anisotropic remeshing for cloth simulation. *ACM transactions on graphics (TOG)*, 31(6):1–10, 2012. 7

[20] C. Patel, Z. Liao, and G. Pons-Moll. Tailornet: Predicting clothing in 3d as a function of human pose, shape and gar-

ment style. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7365–7375, 2020. 2, 3, 5, 7, 9, 10

[21] T. Pfaff, R. Narain, J. M. De Joya, and J. F. O'Brien. Adaptive tearing and cracking of thin sheets. *ACM Transactions on Graphics (TOG)*, 33(4):1–9, 2014. 7

[22] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Proc. of Graphics Interface*, pages 147–154, 1995. 2

[23] I. Santesteban, M. A. Otaduy, and D. Casas. Learning-based animation of clothing for virtual try-on. In *Computer Graphics Forum*, volume 38, pages 355–366. Wiley Online Library, 2019. 2, 3, 4, 9, 10

[24] I. Santesteban, M. A. Otaduy, and D. Casas. Snug: Self-supervised neural dynamic garments, 2022. 11

[25] I. Santesteban, N. Thuerey, M. A. Otaduy, and D. Casas. Self-supervised collision handling via generative 3d garment models for virtual try-on. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11763–11773, 2021. 3, 9, 10

[26] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020. 5

[27] Q. Tan, Z. Pan, and D. Manocha. Lcollision: Fast generation of collision-free human poses using learned non-penetration constraints. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI*, 2021. 3

[28] Q. Tan, Z. Pan, B. Smith, T. Shiratori, and D. Manocha. Active learning of neural collision handler for complex 3d mesh deformations, 2021. 3

[29] M. Tang, R. Tong, Z. Wang, and D. Manocha. Fast and exact continuous collision detection with Bernstein sign classification. *ACM Trans. Graph. (SIGGRAPH Asia)*, 33:186:1–186:8, November 2014. 2

[30] M. Tang, H. Wang, L. Tang, R. Tong, and D. Manocha. Cama: Contact-aware matrix assembly with unified collision handling for gpu-based cloth simulation. In *Computer Graphics Forum*, volume 35, pages 511–521. Wiley Online Library, 2016. 3

[31] M. Tang, T. Wang, Z. Liu, R. Tong, and D. Manocha. I-cloth: Incremental collision handling for gpu-based interactive cloth simulation. *ACM Transactions on Graphics (TOG)*, 37(6):1–10, 2018. 2

[32] R. Vidaurre, I. Santesteban, E. Garces, and D. Casas. Fully convolutional graph neural networks for parametric virtual try-on. In *Computer Graphics Forum*, volume 39, pages 145–156. Wiley Online Library, 2020. 2, 3

[33] T. Y. Wang, T. Shao, K. Fu, and N. J. Mitra. Learning an intrinsic garment space for interactive authoring of garment animation. *ACM Transactions on Graphics (TOG)*, 38(6):1–12, 2019. 2, 3, 7, 9, 10