DOI: 10.1111/cgf.14937
Pacific Graphics 2023
R. Chaine, Z. Deng, and M. H. Kim
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 7

D-Cloth: Skinning-based Cloth Dynamic Prediction with a Three-stage Network

Y. D. Li[1,3] M. Tang[1,3] X. R. Chen[1,3] Y. Yang[1,3] R. F. Tong[1] B. L. An[2] S. C. Yang[2] Y. Li[2] and Q. L. Kou[2]

[1]Zhejiang University, China
[2] Tencent, China
[3] ZJU-Tencent Game and Intelligent Graphics Innovation Technology Joint Lab, China

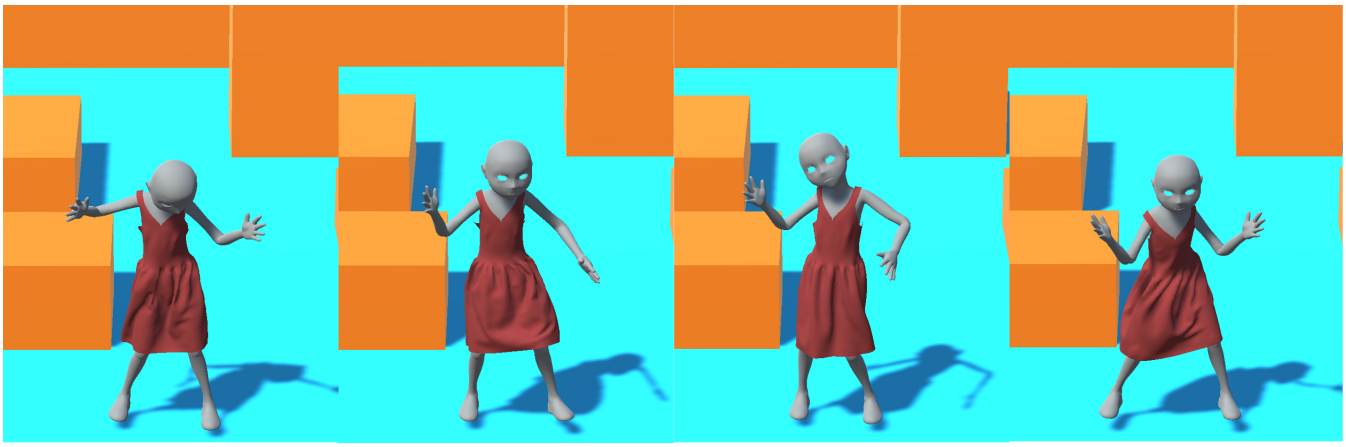

Figure 1: *The running results of our three-stage network in a Unity game scene. Taking into account the movements of characters, our network can predict cloth dynamic deformations in real-time.*

Abstract

We propose a three-stage network that utilizes a skinning-based model to accurately predict dynamic cloth deformation. Our approach decomposes cloth deformation into three distinct components: static, coarse dynamic, and wrinkle dynamic components. To capture these components, we train our three-stage network accordingly. In the first stage, the static component is predicted by constructing a static skinning model that incorporates learned joint increments and skinning weight increments. Then, in the second stage, the coarse dynamic component is added to the static skinning model by incorporating serialized skeleton information. Finally, in the third stage, the mesh sequence stage refines the prediction by incorporating the wrinkle dynamic component using serialized mesh information. We have implemented our network and used it in a Unity game scene, enabling real-time prediction of cloth dynamics. Our implementation achieves impressive prediction speeds of approximately 3.65ms using an NVIDIA GeForce RTX 3090 GPU and 9.66ms on an Intel i7-7700 CPU. Compared to SOTA methods, our network excels in accurately capturing fine dynamic cloth deformations.

CCS Concepts
• ***Computing methodologies*** → *Machine learning; Physical simulation;*

1. Introduction

Cloth simulation is a highly active and important research topic in computer graphics. The ability to accurately simulate cloth deformation has numerous applications in fashion design, movie making, video games, VR/AR, etc. Continuous advancements in cloth deformation techniques have significantly enhanced the visual quality and realism of digital content across various domains.

By simulating how cloth interacts with external forces, such as gravity, winds, and collisions, realistic animations and virtual try-on experiences can be created. physics-based simulation methods are commonly employed to achieve realism in cloth deformation [BW98; Pro95; BML*14; BFA02; HVTG08; TTWM14; TWL*18]. These methods involve discretizing the cloth into smaller elements, such as triangles or particles, and constructing mechanical models to simulate the stretching and bending behaviors of these elements. By applying physical laws and principles, these approaches allow for the realistic rendering of cloth motion, folding, draping, and other intricate details. However, due to the computational complexity, achieving real-time frame rates can be challenging.

For real-time cloth deformation, data-driven and machine learning-based approaches have gained popularity. These methods aim to reduce computational costs by leveraging learned information about cloth behavior. Several learning-based networks such as TailorNet [PLP20], DeepSD [BMTE21], and Virtual-TryOn [SOC19] have been proposed specifically for SMPL-based human models [LMR*15]. These methods aim to predict how a particular garment would fit and drape on a specific human body pose. While these predictions are static, dynamic cloth deformation is also employed to capture realistic motion and interaction of clothes in scenarios. By incorporating the dynamic loss function into an unsupervised learning framework, [BME22] aims to capture the temporal aspects and physical behavior of cloth in motion. The proposed dynamic loss function appears to evaluate the stretching and bending of cloth materials, enabling the prediction of dynamic cloth deformation. [PMJ*22] extracts virtual skeletons using SSDR [LD12] from a dataset of dynamic cloth deformation sequences and utilizes them for predicting cloth deformations. By employing the GRU module [CVG*14], the method aims to predict the sequence of virtual skeleton information, which is then used for the skinning process to obtain the final cloth deformation. However, this method relies heavily on the extraction results for virtual skeletons of SSDR. A small number of skeletons may result in lower-quality skinning results.

In this paper, we present a three-stage network designed specifically for skeleton-based characters to predict sequential cloth dynamic deformations. Our method focuses on constructing a learning-based skinning model that accurately captures the deformation of clothes worn by these characters. We decompose the cloth deformation into three distinct components: static, coarse dynamic, and wrinkle dynamic components. Each stage of our network is dedicated to predicting one of these components, ultimately leading to the final dynamic deformation results.

Specifically, the three-stage network comprises a static stage, a skeleton sequence stage, and a mesh sequence stage, each playing a crucial role in predicting the different components of cloth deformations. The static stage is responsible for predicting the static deformation of clothes for the posed character by optimizing the nonlinear skinning weights of the clothes. The skeleton sequence stage leverages skeleton sequence information to predict the coarse dynamic component of the cloth template. The mesh sequence stage focuses on predicting the wrinkle dynamic component of the cloth template. By utilizing mesh sequence information, this stage introduces fine-scale details and intricate dynamic wrinkles to the cloth mesh. The combination of the skeleton sequence stage and the mesh sequence stage adds dynamic features to the static deformation obtained in the earlier stage. By decomposing cloth deformation into static, coarse dynamic, and wrinkle dynamic components and utilizing the three-stage network, we can achieve more realistic and visually appealing cloth simulations for skeleton-based characters.

Through our qualitative and quantitative analyses, as well as the comparative experiments, we demonstrate that our three-stage network has the capability to reasonably predict dynamic cloth deformation. We provide a comprehensive evaluation by presenting several examples of predicted dynamic deformations, showcasing the capabilities of our approach. To further validate the efficacy of our three-stage network, we performed ablation experiments to demonstrate the effectiveness and necessity of each stage in our proposed approach. We also conducted comparative experiments with SOTA methods, including those focusing on static prediction and dynamic prediction. The comparative experimental results provide insights into the strengths and advantages of our approach over these methods.

The contributions of this work include:

- **Cloth dynamic skinning model:** By utilizing sequence-based information, such as skeleton sequence and mesh sequence, our network goes beyond traditional static skinning methods and takes into account the temporal aspect of cloth deformation.
- **Three-stage network for cloth dynamic deformation prediction:** We adopt a decomposition approach to divide the deformation process into three distinct components: static, coarse dynamic, and wrinkle dynamic components. By learning and modeling each component through a static stage, a skeleton sequence stage, and a mesh sequence stage, respectively, we can effectively combine their results to predict the final dynamic deformation of the cloth.
- **Graph sequence GRU model (GSGRU) for mesh sequential information:** We have introduced a graph sequence GRU model that enables the extraction of features, obtaining outputs, and next hidden information from serialized graphs.

Overall, our approach offers a comprehensive solution to predicting cloth deformation by breaking it down into distinct components and leveraging the static and temporal information to achieve a realistic representation of cloth dynamics on skeleton-based characters.

2. Related Work

In this section, we provide an overview of cloth deformation techniques, focusing on both traditional physically-based simulation (PBS) methods and recent learning-based approaches. For learning-based approaches, there are two main categories: static prediction and dynamic prediction methods. We will discuss both approaches and highlight their characteristics.

2.1. Physics-based Simulation

Traditional PBS methods iteratively perform time integration [BW98], collision detection [BFA02; TTWM14], and collision

response [BFA02; HVTG08; TWL*18] over a sequence of time steps. The cloth simulation progresses, and the deformation of the cloth is updated accordingly. These methods, based on principles of physics, can accurately model cloth deformation and ensure non-penetrating simulations. However, their computational cost can be a limitation for interactive applications. To address this limitation, recent research has focused on leveraging GPUs to parallelize the computations [TWT*16; TWL*18; LTT*20; WWW22]. These methods take advantage of the massive parallelism offered by GPUs to accelerate the simulation pipeline and are typically difficult to use on devices that are not equipped with GPUs.

2.2. Learning-based Static Prediction

Traditional PBS methods can be computationally expensive and are hard to achieve real-time performance. In recent years, there has been a growing interest in using learning-based approaches for real-time cloth deformation prediction. One popular model used in learning-based approaches is the SMPL model [LMR*15]. By leveraging the SMPL model, researchers have developed methods that can predict cloth deformation in real-time. Patel et al. proposed TailorNet [PLP20]. The cloth mesh is considered a sub-mesh of the SMPL body mesh. The network aims to learn the incremental changes from the template cloth mesh to obtain the subsequent skinning posed cloth. Unfortunately, the performance of this method on loose-fitting garments may be limited. [BME21] uses an unsupervised learning method and cloth consistency loss for mass-spring material to satisfy the prediction of the proposed network. However, the lack of ground truth data with well-deformed clothes will pose a challenge in achieving high physical realism in the prediction results. This model may struggle to accurately capture the intricate details and physical behaviors of loose-fitting clothes or produce unrealistic deformations.

There are alternative methods that deviate from SMPL-based approaches and employ different techniques for cloth deformation prediction. By combining DQS [KCŽO07] for initial pre-deformation and using GCN [KW16]blocks to learn the residuals, [GCS*19; GCP*20] methods aim to improve the accuracy and realism of cloth deformation predictions. By leveraging the power of deep learning and incorporating specific techniques for handling mesh data, they provide an alternative approach to cloth deformation prediction outside the SMPL-based paradigm. But these methods are also limited to the tight cloth type. In [VSGC20], an encoder-decoder architecture is employed, along with GCNs, to learn the draping effect of different cloth types on the canonical pose. But it can not effectively handle cloth deformation in poses other than the canonical T-pose.

2.3. Learning-based Dynamic Prediction

Static prediction methods, which focus on predicting the deformation of clothes in a single frame or pose, often lack the dynamic features that contribute to realistic cloth simulation. As a result, the predicted results may appear stiff or lacking in natural movement. The use of sequential information in cloth deformation prediction is an active area of research, and various techniques and architectures have been proposed to leverage this information effectively. [SOC19] utilizes a garment fit regressor and a garment wrinkle regressor to capture the deviations from the canonical shape. And this method uses the GRU module to capture the sequence information to predict the dynamic behavior of cloth deformations. [STOC21] uses the GRU module and proposes an optimization-based method to obtain the collision-free prediction of cloth deformation. This method aims to produce collision-free results based on sequence prediction. However, these models are specifically designed for the SMPL model and often rely on the parameters of SMPL.

Besides the SMPL-based models, [HDDN19] uses Principal Component Analysis (PCA) to obtain the subspace representation and then employs Multi-Layer Perceptron (MLP) networks to regress this subspace. [WSFM19], focuses on learning a cloth descriptor using only the vertex coordinates of the cloth mesh. This cloth descriptor is then fused with motion information in latent space to capture the dynamic behavior of the cloth. [BME22] employs an unsupervised learning approach to enhance the generation of dynamic cloth deformations. By leveraging the sequence-based dynamic loss, the model can capture temporal dependencies and generate more accurate and visually appealing cloth deformations. However, these approaches lack of using the topology information of the mesh. [PMJ*22] uses the SSDR method to extract virtual bones from a dataset of dynamic cloth deformation sequences. And then use a GRU module to predict the virtual bone for the subsequent skinning process to obtain the final cloth deformation. But this approach relies on the effectiveness of the SSDR method. [HLB*23] employs a recurrent graph neural network to enhance the coarse-resolution mesh with physics-based intricacies. This is achieved by distinctively separating the force modeling and integration processes.

3. Method

In this paper, we decompose the cloth deformation into static, coarse dynamic, and wrinkle dynamic components to enhance the predictive performance of the network. Figure 2 visually demonstrates the distinctions between static and dynamic deformations, highlighting the increased sense of realism in dynamic deformations.

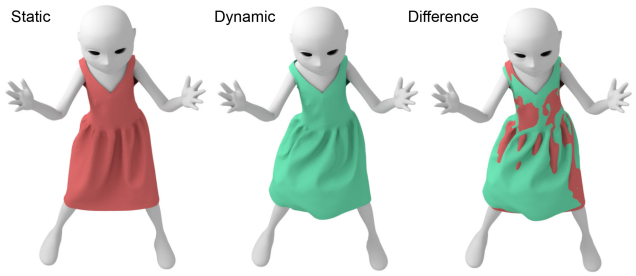

Figure 2: *The static and dynamic cloth deformation: The first column is the static cloth deformation. The second column is the dynamic cloth deformation. The third column is the difference between static and dynamic.*

Inspired by [SOC19; STOC21; SOC22], we proposed a three-stage network architecture that aims to construct a dynamic skin-

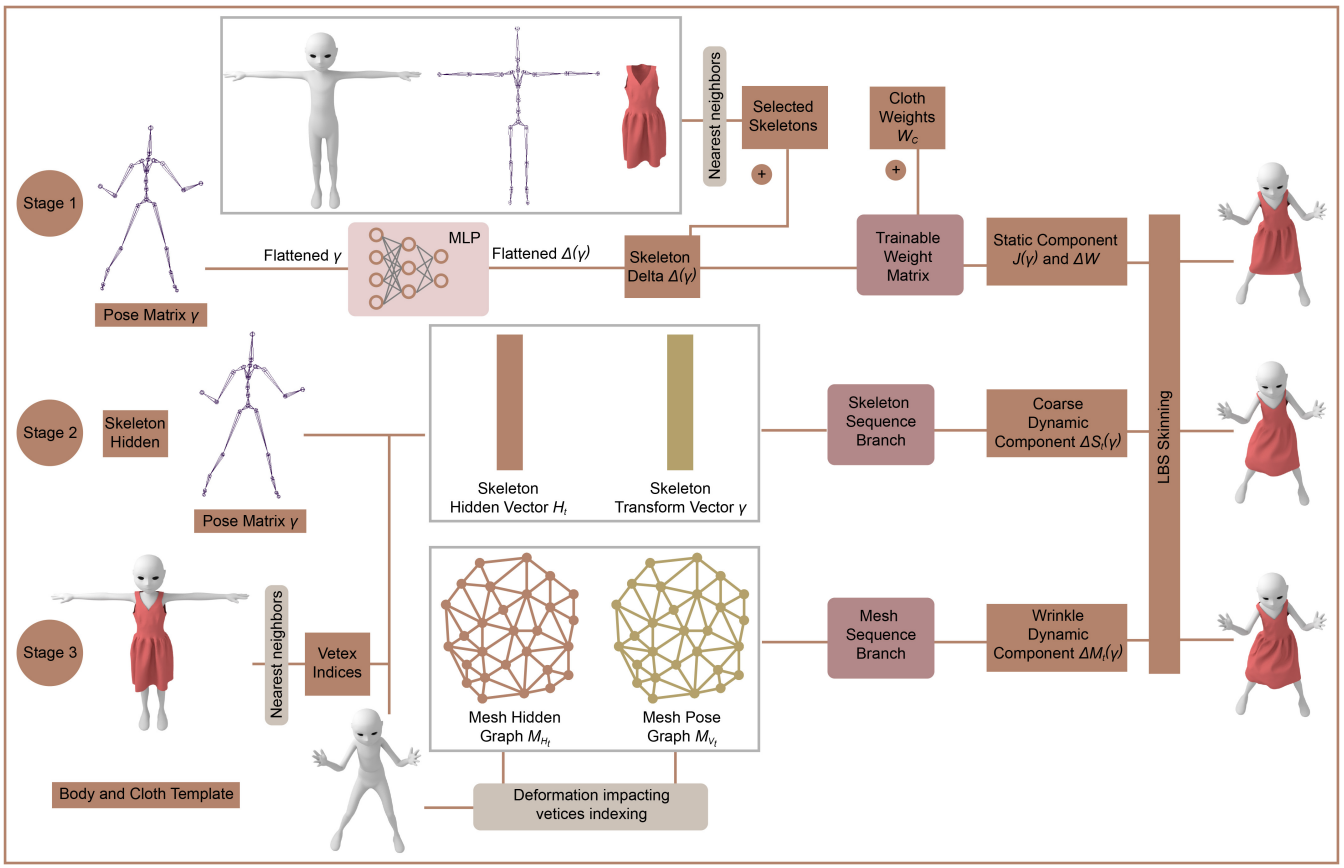

Figure 3: *Our three-stage network architecture: The static stage aims to learn a static cloth skinning model. The skeleton sequence stage and the mesh sequence stage are trained to add coarse dynamic component $\Delta_{S_t}(\gamma)$ and wrinkle dynamic component $\Delta_{M_t}(\gamma)$. The serialized skeleton information and the serialized mesh information are used to add dynamics.*

ning model for clothes worn on skeleton-based characters using a learning-based approach. Thus, the deformed cloth is obtained by applying the skinning operation to the template cloth mesh $\mathbf{T_C}$ and then transforming it according to the target pose using the skeleton transformation matrix γ. The specific skinning formulas are as follows:

$$M_C(\gamma) = \mathcal{W}(T_C(\gamma), \mathcal{J}(\gamma), \gamma, W_C), \tag{1}$$

where γ is the transformation matrix of the target pose joints. $T_C(\gamma)$ is the cloth template mesh relied on γ. $\mathcal{J}(\gamma)$ is the skeleton of cloth relied on the joints transformation γ. W_C is the skinning weight matrix for cloth template mesh $\mathbf{T_C}$. $W(\cdot)$ represents the skinning function. In this paper, the linear blend skinning (LBS) method [MLTM88] is utilized.

Specifically, our three-stage network consists of three parts: the static stage, skeleton sequence stage, and mesh sequence stage. These stages work together to predict the dynamic deformation of the cloth. In the static stage, we focus on predicting the static deformation of the cloth for specific character poses. After this stage, we obtain the optimized cloth skinning weight W_C and pose-based skeleton $\mathcal{J}(\gamma)$. The skeleton sequence stage and mesh sequence stage are the dynamic stages. The skeleton sequence stage aims to add coarse dynamic deformation from the static prediction of the static stage. The mesh sequence stage focused on capturing the dynamic fine details for the coarse dynamic prediction of the coarse sequence stage. The skeleton sequence stage and mesh sequence stage are formulated as follows:

$$T_C(\gamma) = \mathbf{T_C} + \Delta_{S_t}(\gamma) + \Delta_{M_t}(\gamma), \tag{2}$$

where $\mathbf{T_C}$ is the template cloth mesh at the canonical pose. $\Delta_S(\gamma)$ is the coarse dynamic component obtained from the skeleton sequence stage. $\Delta_M(\gamma)$ is the wrinkle dynamic component obtained from the mesh sequence stage. We highlight our three-stage network architecture in Fig. 3. Illustrated in Fig. 3, both the static stage and the skeleton sequence stage employ solely skeleton information to construct the cloth skinning model. In contrast to the static stage, which exclusively relies on individual static skeleton information, the skeleton sequence stage utilizes skeleton sequence data to transition the static skinning model into a dynamic representation. The mesh sequence stage leverages mesh sequence information to incorporate intricacies into the dynamic cloth skinning model.

By combining the results of the static stage, skeleton sequence

stage, and mesh sequence stage, we obtain the final dynamic deformation of the cloth. The three-stage network allows us to learn and incorporate different components of the cloth deformation process, resulting in realistic and dynamic cloth deformation.

3.1. Static Stage

The static stage in our three-stage network is dedicated to constructing a learning-based skinning model for deforming static cloth meshes. This stage obtains two components: cloth skeleton $\mathcal{J}(\gamma)$ and cloth skinning weights W_C.

To achieve this, the static stage network is designed with two parts. The first part uses MLP to learn the increment $\Delta(\gamma)$ of the cloth skeleton. As depicted in Figure 3, during the static stage, the transformation matrix γ is flattened and input into the MLP to derive the flattened increment $\Delta(\gamma)$ for the cloth skeleton. For the initial cloth skeleton $\mathcal{J}_\mathcal{C}$, we choose the relevant character bones through the nearest point on the character mesh for each cloth vertex. To obtain this, we build a KD-tree for the template cloth mesh and the template character mesh. The chosen bone is the character bone of the nearest point. Thus the newly updated cloth skeleton is obtained by:

$$\mathcal{J}(\gamma) = \mathcal{J}_\mathcal{C} + \Delta(\gamma), \tag{3}$$

The second part of the static stage aims to learn the increment of the cloth weights ΔW. We use a set of trainable weight residual matrices $D = \{W_1, W_2, W_3, \cdots W_m\}$ to generate the increment ΔW. m is the number of the cloth bones. For matrix W_j, where $j \in \{1, 2, \cdots, m\}$, W_j is expressed as:

$$W_j = \begin{bmatrix} w_{00} & \cdots & w_{0m} \\ \vdots & \ddots & \vdots \\ w_{n0} & \cdots & w_{nm} \end{bmatrix}, \tag{4}$$

where $w_{00}, \cdots, w_{0m}, \cdots, w_{n0}, \cdots, w_{nm}$ are trainable parameters. n is the number of vertices of the template cloth mesh.

Thus, the updated cloth weight is calculated as follows:

$$\begin{aligned} \Delta W &= W_1 + W_2 + W_3 + \cdots + W_m, \\ W_C &= W_I + \Delta W, \end{aligned} \tag{5}$$

where W_I is the chosen character weight through the nearest point operation as above.

By combining the results from both parts of the static stage network, we obtain a comprehensive learning-based skinning model for static cloth deformation. This model allows us to predict the incremental changes in the cloth skeleton and skinning weights.

3.2. Skeleton Sequence Stage

The static stage of our three-stage network establishes a skinning model that introduces non-linearity into the cloth deformation process. However, the skinning model is static. To enhance the realism of prediction, it is important to transition from a static skinning model to a dynamic skinning model. To capture coarse dynamic effects in cloth deformation, we introduce the skeleton sequence stage. The network architecture of the skeleton sequence stage is illustrated in Fig. 4. As illustrated in Fig. 4, the GRU module [CVG*14] serves as a pose embedding, processing the skeleton sequence information.

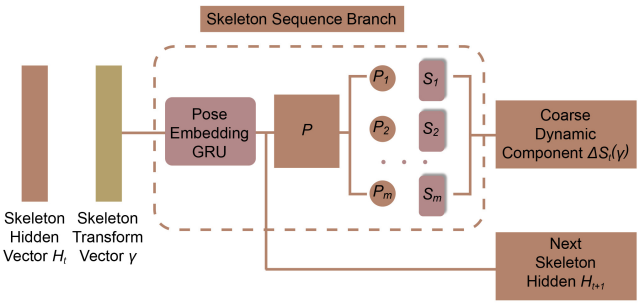

Figure 4: *The skeleton sequence stage network: The skeleton sequence stage takes two inputs: the skeleton transformation of the target pose and the hidden information. And it generates the coarse dynamic component and the next hidden information.*

The skeleton sequence stage utilizes serialized skeleton information to incorporate dynamic coarse information into the static skinning model. There are the current skeleton transformation γ of target pose and the hidden information $\mathcal{H}_t$ at current state t as two inputs. The initial hidden information, denoted as $\mathcal{H}_0$ at initial state $t = 0$, is initialized as a zero vector. We pass these two inputs into the pose embedding GRU network, which is composed of GRU module [CVG*14], to learn the dynamic pose embedding $\mathcal{P}_t = \{P_1, P_2, P_2, \cdots, P_k\}$, where k is the size of the embedding vector $\mathcal{P}$:

$$\mathcal{P}_t, \mathcal{H}_{t+1} = \mathbf{GRU}(\gamma, \mathcal{H}_t), \tag{6}$$

where $\mathcal{H}_{t+1}$ is the next hidden information at next state $t+1$.

After the pose embedding, we use a set of trainable skeleton matrices $S_t = \{S_1, S_2, S_3, \cdots S_k\}$. As for matrix S_j, where $j \in \{1, 2, \cdots, k\}$, S_j is expressed as:

$$S_j = \begin{bmatrix} s_{00} & \cdots & s_{02} \\ \vdots & \ddots & \vdots \\ s_{n0} & \cdots & s_{n2} \end{bmatrix}, \tag{7}$$

where $s_{00}, \cdots, s_{02}, \cdots, s_{n0}, \cdots, s_{n2}$ are trainable parameters. n is the number of vertices of the template cloth mesh.

Then, the pose embedding $\mathcal{P}_t$ is fused with the skeleton matrices S_t to obtain the coarse dynamic component $\Delta_{S_t}(\gamma)$:

$$\Delta_{S_t}(\gamma) = \sum_{j=0}^{j=k} P_j S_j \tag{8}$$

By taking the skeleton transformation of the current pose, and the hidden information into account, the skeleton sequence stage effectively predicts the coarse dynamic component of the cloth template and updates the next hidden information.

The skeleton sequence stage captures the sequence dependencies and movement patterns of the character to add dynamic features to the cloth deformation. This stage acts as an intermediary stage,

bridging the gap between static skinning and full-fledged dynamic deformation.

3.3. Mesh Sequence Stage

The mesh sequence stage is an essential component of our three-stage network, designed to enhance the dynamic deformation of the cloth by adding wrinkle detail features. It builds upon the coarse dynamic skinning model generated in the skeleton sequence stage.

In the mesh sequence stage, we introduce serialized mesh data as input to capture the detailed features of the cloth deformation. This serialized mesh data consists of mesh $\mathcal{M}_{\mathcal{V}_t}$ with node features at the current state t and mesh $\mathcal{M}_{\mathcal{H}_t}$ with hidden information.

Specifically, we use the positions $\mathcal{V}$ of the nearest points on the posed character mesh as mentioned above, the edges of template cloth mesh $\mathcal{E}$, and the adjacency matrix $\mathcal{A}$ to compose mesh $\mathcal{M}_{\mathcal{V}_t} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$. The mesh $\mathcal{M}_{\mathcal{H}_t}(\mathcal{Z}, \mathcal{E}, \mathcal{A})$ is composed of the hidden information $\mathcal{Z}$, the edges of template cloth mesh $\mathcal{E}$, and the adjacency matrix $\mathcal{A}$.

Graph Sequence GRU Model(GSGRU). Graph data such as mesh often consists of interconnected nodes and edges. However, traditional GRU [CVG*14] struggles to effectively process and model graph sequence data. To address this challenge, inspired by [CVG*14; LZBT16], we have designed a graph sequence GRU model(GSGRU) that operates on serialized graphs. The model is capable of capturing the sequential dependencies present in the serialized graphs.

We define $X^{(l)} = \left\{x_1^{(l)}, x_2^{(l)}, \ldots, x_n^{(l)}\right\}$ as the node features of previous layer l on mesh graph $\mathcal{M}_{\mathcal{V}_t}$. n is the number of nodes. $x_i^l \in \mathbf{R}^{F_V}$ represents the features of node i whose dimension is F_V in layer l on mesh graph $\mathcal{M}_{\mathcal{V}_t}$. Similarly, we define $H^{(l)} = \left\{h_1^{(l)}, h_2^{(l)}, \ldots, h_n^{(l)}\right\}$ as the hidden features of previous layer l on mesh graph $\mathcal{M}_{\mathcal{H}_t}$. $h_i^l \in \mathbf{R}^{F_Z}$ represents the hidden features of node i whose dimension is F_Z.

For the node features of node i on mesh graph $\mathcal{M}_{\mathcal{V}_t}$, the node features are updated by follow:

$$a_i^{(l)} = \sum_{j \in \mathcal{N}(i)} e_{ji} \mathbf{W}^p x_j^{(l)} + b \tag{9}$$

where e_{ij} represents the edge weight from node j to node i. node j is the neighbor of node i. $\mathbf{W}^p$ and b are trainable parameters. $\mathcal{N}(i)$ refers to the neighboring nodes of node i.

Then, the updated node feature $a_i^{(l)}$ and hidden feature $h_i^{(l)}$ are fused to obtain the update gate as follow:

$$z_i^l = \sigma\left(\mathbf{W}^z a_i^{(l)} + \mathbf{U}^z h_i^{(l)}\right) \tag{10}$$

where $\mathbf{W}^z$ and $\mathbf{U}^z$ are trainable parameters. σ is the activation function and is formulated as:

$$\sigma(x) = 1/\left(1 + e^{-x}\right) \tag{11}$$

where x is the input of the activation function.

The reset gate is calculated by:

$$r_i^l = \sigma\left(\mathbf{W}^r a_i^{(l)} + \mathbf{U}^r h_i^{(l)}\right) \tag{12}$$

where $\mathbf{W}^r$ and $\mathbf{U}^r$ are trainable parameters.

Then the gate signals are fused with the features:

$$\widetilde{h_i^{(l+1)}} = \tanh\left(\mathbf{W} a_i^{(l)} + \mathbf{U}\left(r_i^l \odot h_i^{(l)}\right)\right) \tag{13}$$

where $\mathbf{W}$ and $\mathbf{U}$ are trainable parameters, the operator $\odot$ represents element-wise multiplication.

The hidden features for the next layer $l+1$ is updated as follows:

$$h_i^{(l+1)} = \left(1 - z_i^t\right) \odot h_i^{(l)} + z_i^l \odot \widetilde{h_i^{(l+1)}} \tag{14}$$

And the output features of the current layer l are calculated as follows:

$$o_i^{(l)} = \mathbf{MLP}(h_i^{(l+1)}) \tag{15}$$

By incorporating both the node features and hidden information of the mesh, the GSGRU can effectively capture and model the intricate details and dynamics of the cloth deformation over mesh sequences.

Mesh Sequence Stage Architecture. The architecture of the mesh sequence stage, as shown in Fig. 5, utilizes the GSGRU model to process serialized mesh data and capture the sequence dependencies. As depicted in Fig. 5, two sets of mesh data are fed into the GSGRU model to extract sequence-relevant features. Subsequently, LayerNorm [BKH16] and PReLU [HZRS15] functions are employed to introduce nonlinearity. Then, several analogous structures follow, wherein the GSGRU, LayerNorm, and PReLU layers from the previous stages are concatenated.

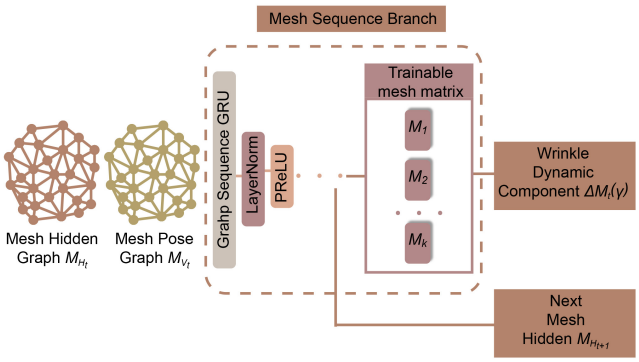

Figure 5: *The mesh sequence stage network: The mesh sequence stage takes mesh node features and mesh hidden information as inputs. And it generates the wrinkle dynamic component and the next hidden information.*

Regarding the initial hidden information $H^{(0)}$ of the mesh graph $\mathcal{M}_{\mathcal{H}0}$ at the initial state $t = 0$, it is initialized as a zero vector. This leads to the initial hidden information of the mesh graph $\mathcal{M}_{\mathcal{H}0}$ being represented as a zero tensor.

After the feature extraction on the mesh sequence, we use a vertex level MLP to learn the global embedding $\mathcal{Q}_t = \{Q_1, Q_2, Q_2, \cdots, Q_k\}$ and a set of mesh matrices are trained as

$M_t = \{M_1, M_2, M_3, \cdots M_k\}$. Then he wrinkle dynamic component $\Delta_{M_t}(\gamma)$ is calculated as:

$$\Delta_{M_t}(\gamma) = \sum_{j=0}^{j=k} Q_j M_j \tag{16}$$

Considering both the node features and hidden information of the mesh, the mesh sequence stage can effectively capture the wrinkle dynamics of the cloth deformation.

Incorporating the mesh sequence stage into our three-stage network, we are able to capture both coarse and fine details of the cloth deformation, resulting in a final dynamic deformation that exhibits realistic wrinkle patterns and enhances the overall visual quality of the cloth simulation.

3.4. Loss function

To optimize the parameters of our network architecture, we utilize the Mean Squared Error (MSE) loss that quantifies the difference between the predicted deformed cloth mesh and the ground truth. The specific form of the loss function is defined as follows:

$$\mathcal{L} = \frac{1}{N} \sum_{j=1}^{N} \left\| x_p^j - x_g^j \right\|_2, \tag{17}$$

where x_p^j is the predicted position of vertex j in the cloth mesh. x_g^j is the corresponding ground truth position. N is the number of cloth vertices. $\|\cdots\|_2$ is the L_2 distance.

4. Dataset and Implementation

In this section, we provide details about the dataset generation process and discuss some implementation aspects of our method.

4.1. Dataset

To capture the dynamic deformation of loose-fitting garments, we create two specific types of clothing: a loose skirt and a loose robe. The skirt has 7964 vertices and 15176 triangle faces, while the robe has 9738 vertices and 19168 triangle faces. These garments have fewer constraints imposed by the character's body and allow for more pronounced dynamic deformation during character movement.

To generate action sequences for our dataset, we obtained character motion FBX files from the Mixamo website https://www.mixamo.com/. These FBX files contain predefined animations for various actions, such as walking, running, and jumping. We chose a range of action sequences to ensure diversity in the types of movements and poses exhibited by the characters. We use 45 sequences to train our network and left 4 sequences for testing. We set the hip joint of the character as the coordinate origin for all poses in the action sequences. This step helps align the positions of the characters across different frames and eliminates any translation differences that may exist in the original FBX files. Then, we extracted the joint transformation matrices of the character at a frame rate of 30 FPS for each pose of the action sequences. The template mesh and skinning weights are extracted from the FBX files.

We employed the ArcSim simulator [NSO12; NPO13; PNDO14] to generate both static and dynamic cloth deformations for our experiments. For the static deformations, we performed pose interpolation to generate the intermediate static deformations between adjacent keyframes. During the static deformation simulation, we allowed the cloth to relax to eliminate any dynamic effects. For the dynamic deformations, we performed regular cloth simulations using the ArcSim simulator.

4.2. Implementation

Our network training was conducted on a standard PC with the following specifications: Ubuntu 22.04 LTS operating system, Intel Core i7 CPU running at 4.2 GHz, 8 GB of RAM, and an NVIDIA GeForce RTX 3090 GPU. The implementation of our network was done using PyTorch 1.7.0 and Python 3.8.8.

The training process for our network is divided into three stages, aligning with the architecture of our network. Each stage focuses on training a specific component of the network to achieve the desired cloth deformation. The first stage involves learning the skinning weights and joint increments of the cloth to obtain a static skinning model. Once the static stage is trained, we move on to the second stage. In this stage, we fix the skinning weights and joint increments obtained from the first stage and focus on training to capture the coarse dynamic component of the cloth deformation. Then, we proceed to the third stage. In this stage, we fix the weights obtained from the previous two stages and train to capture the wrinkle dynamic component of the cloth deformation.

During the training process, we set the learning rate to $1e-3$ and employed the Adam optimizer [KB14] to update the parameters of our neural network.

4.3. Penetration Handling

We employ a post-processing optimization method similar to the one described in [WSFM19] to address the issue of penetrations between the cloth and the character in the predicted deformations. This optimization step is performed after the prediction is generated, and its purpose is to refine the predicted deformed cloth mesh to reduce penetrations and achieve collision-free configurations. The deformed cloth mesh is refined by minimizing the following function:

$$E_B = \sum_{i \in V_{pene}} \left\| v_i - \left(v_i^B + \varepsilon n_i^B \right) \right\|, \tag{18}$$

Here, V_{pene} signifies the collection of vertices in the predicted cloth mesh that have penetrated. For each penetrated vertex v_i, the corresponding closest point vertex v_i^B and its associated normal n_i^B are computed based on the character mesh. The term E_B quantifies the error stemming from penetration vertices found in both the cloth and character meshes. Additionally, ε denotes a minor adjustment employed to displace the penetrated vertices away from the character mesh.

5. Results

In this section, we showcase the prediction results of our network and provide a comprehensive comparison with the results obtained

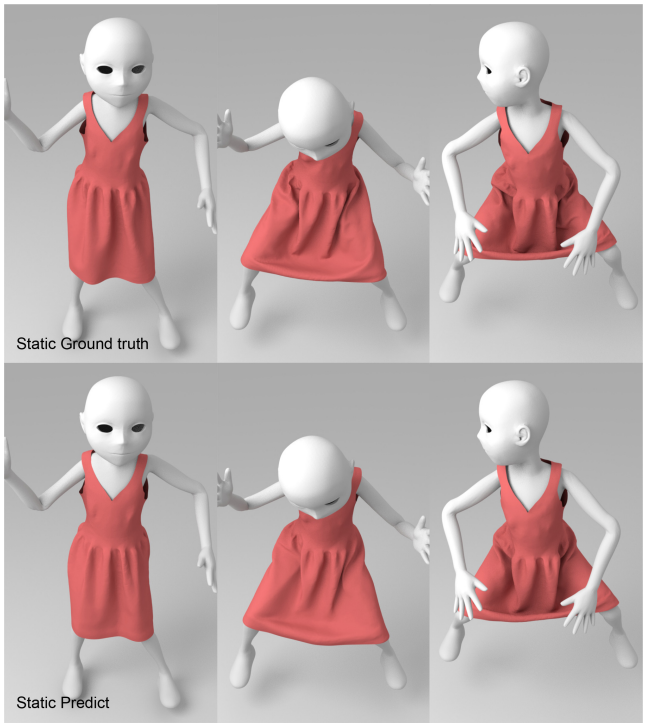

Figure 6: *The predicted static deformed skirt results on test unseen data: The top row shows the ground truth of the static deformation, while the bottom row highlights the predictions of our network.*

from static prediction deformation, coarse dynamic deformation, and wrinkle dynamic deformation.

5.1. Predicted Deformation

Fig. 6 and Fig. 7 visually demonstrate the outcomes of the different stages in our network for predicting cloth deformations. The static prediction results obtained from the static stage are shown for the skirt in Fig. 6. Fig. 7 presents the dynamic prediction results of our network, showcasing the contributions of the skeleton sequence stage and the mesh sequence stage. The second row illustrates the outputs of the skeleton sequence stage, which introduces coarse dynamic effects on top of the static deformation. The third row displays the prediction results of the mesh sequence stage, which further refines the dynamic effects by adding detailed wrinkles to the coarse dynamic deformation.

Fig. 8 demonstrates the static and dynamic prediction results of our network on the robe. The dynamic prediction results exhibit more realistic deformations in the sleeves and the hem of the robe compared to the static prediction.

Halimi et al.[HLB*23] utilize a recurrent variant of a graph network[PFSB20] to enhance the level of detail in the initial coarse cloth mesh. Specifically, Halimi et al. [HLB*23] create recurrent graph vertices by concatenating the current state's force with the first-order differences in the sequence configurations. This approach enables them to incorporate sequence information into

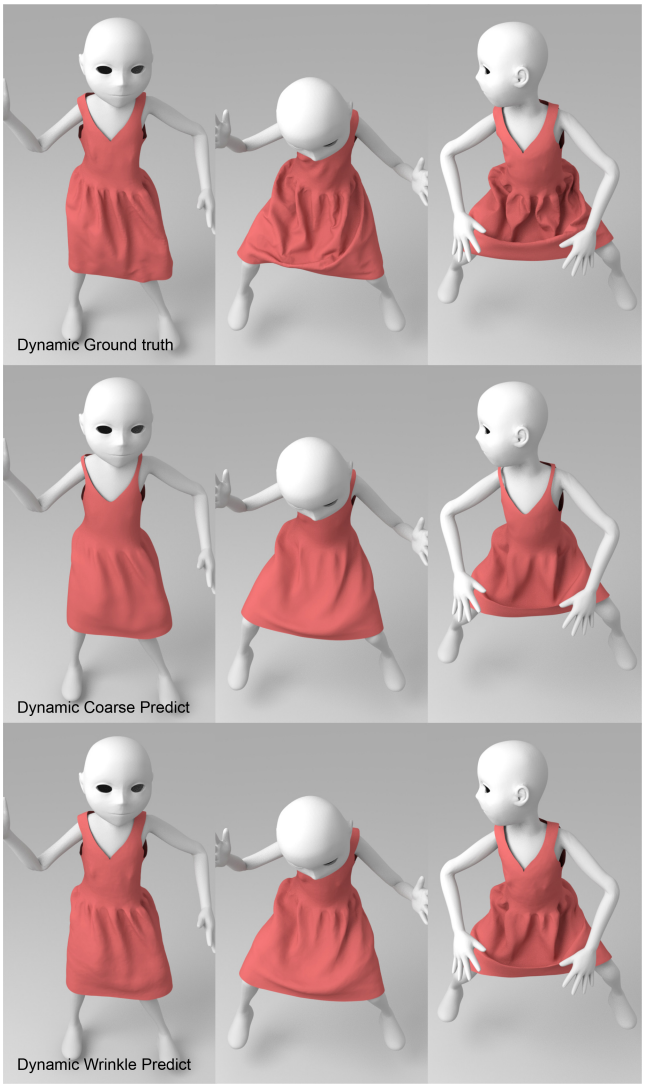

Figure 7: *The predicted dynamic deformed skirt results on test unseen data: The top row shows the ground truth of the dynamic deformation. The second row shows the coarse prediction. The bottom row shows the wrinkle prediction.*

a single graph to extract features. In comparison to Halimi et al. [HLB*23], our GSGRU architecture takes a step further. It operates on two distinct sequence graphs with hidden information, to dynamically enhance the final cloth deformation.

5.2. Runtime and Model Size

Our three-stage network exhibits excellent real-time performance in predicting dynamic cloth deformations. When running on an NVIDIA GeForce RTX 3090 GPU, the network can generate predictions within a time duration of 3.65ms. And the generation time of our network is 9.66ms on an Intel Core i7-7700 CPU. The runtime cost of each stage within our network is presented in Tab. 1. It's evident that the static stage incurs the lowest runtime cost,

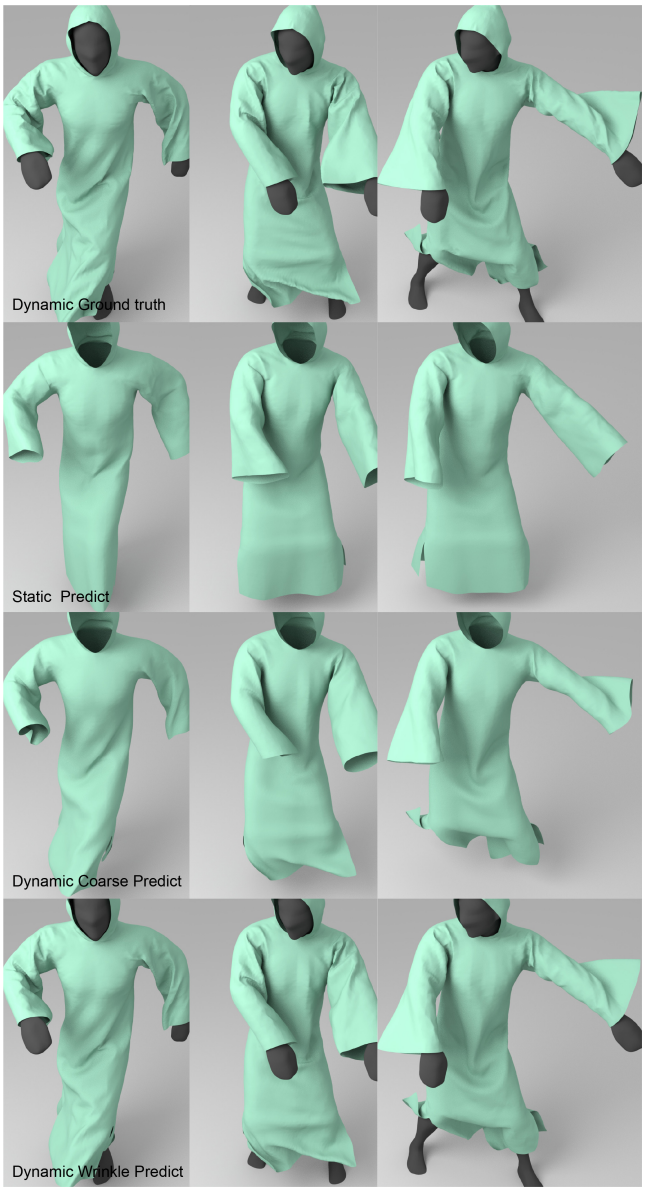

Figure 8: *The predicted outcomes of the deformed robe in various character poses: The top row illustrates the ground truth of dynamic deformation. The second row displays the static prediction. The third row presents the coarse dynamic prediction. The bottom row showcases the dynamic prediction emphasizing wrinkles.*

whereas the mesh sequence stage registers the highest runtime cost. This discrepancy arises from the fact that the static stage exclusively utilizes static skeleton information, resulting in a reduced parameter count. In contrast, both the skeleton sequence stage and the mesh sequence stage integrate sequence information, leading to a higher parameter complexity. Among these stages, the mesh sequence stage, due to its incorporation of mesh data, carries the most significant parameter load, consequently elongating its execution time to the slowest among them. The real-time capability of our network opens up possibilities for a wide range of applications, such as virtual reality, gaming, virtual try-on systems, and virtual character animation.

Table 1: *The runtime cost of each stage network on CPU and GPU.*

Stage	CPU Runtime(s)	GPU Runtime(s)
Static Stage	1.92E-4	1.84E-4
Skeleton Sequence Stage	1.05E-3	3.73E-4
Mesh Sequence Stage	8.41E-3	3.10E-3
Total Architecture	9.66E-3	3.65E-3

Correspondingly, the memory footprint of each stage within our network is illustrated in Tab. 2. The memory footprint of our total network architecture amounts to 130.5MB. The memory footprint of the static stage is 18.9MB. The memory footprint of the skeleton sequence stage and the mesh sequence stage are 40.6MB and 71.0MB respectively. As evident, the memory footprint of the mesh sequence stage surpasses the collective memory footprints of both the static stage and the skeleton sequence stage. The inclusion of the mesh sequence stage notably amplifies the parameter count within our approach. Nonetheless, these supplementary parameters endow the mesh sequence stage with the capability to refine the final deformation outcomes by introducing finer details.

Table 2: *The runtime cost of each stage network on CPU and GPU.*

Stage	Memory Footprint(MB)
Static Stage	18.9
Skeleton Sequence Stage	40.6
Mesh Sequence Stage	71.0
Total Architecture	130.5

5.3. Application on Unity

Real-time cloth deformation prediction has various applications in fields such as computer graphics, virtual try-on, virtual reality, gaming, and animation. The deployment of the network in real-time scenarios opens up new possibilities for interactive and realistic cloth deformation in various fields, enabling more immersive experiences and enhancing the efficiency of virtual clothing-related workflows.

We have deployed our network in Unity. We train the network and save the trained weights. Then in the Unity project, we write scripts in C# to load the saved network weights. This step involves initializing the network architecture and setting the weights to the corresponding layers and parameters. To predict the dynamic deformation of clothes for the posed character, the joint transformation information is extracted from the character in the Unity scene. We feed the joint transformation information of the posed character into the loaded network model. Then the vertex positions of the cloth mesh are updated in the Unity scene.

We have the character model, skirt mesh, and other necessary assets set up in the Unity Editor in Fig. 9. The necessary scripts

and components are attached to the appropriate game objects to handle the prediction and visualization.

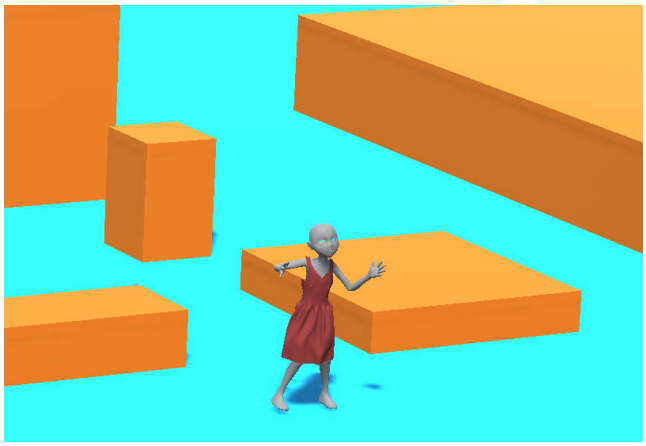

Figure 9: *The application of our network on Unity scene: We deployed our network in Unity to predict the wrinkle dynamic cloth deformation in real-time.*

6. Comparison

In this section, we compare the results of our network with prior learning-based methods and conduct ablation experiments to assess the effectiveness of your network.

6.1. Qualitative Comparisons

We have qualitatively compared the predicted cloth deformations from our network with the results from prior learning-based methods, as shown in Fig. 10.

The PBNS [BME21] is based on unsupervised learning for static deformation prediction. The results may lack physical realism and introduce unnatural deformations in certain parts of the cloth, such as the sleeves and skirt. The CTSN method [LTY*23], which specializes in static deformation prediction based on the skeleton, may exhibit limitations in capturing dynamic effects in its predicted results. The NeuralClothSim [BME22] tackles the issue of limited dynamic effects by integrating a sequence-based dynamic loss function into the unsupervised learning approach. This enables the generation of cloth deformations with more dynamic details. However, since ground truth data is not available, there might still be instances of unnatural deformations, especially in the middle part of the robe and the lower skirt region. The VirtualBone method [PMJ*22] aims to predict virtual bones for cloth deformation by leveraging the SSDR method for virtual bone extraction. However, since this method relies on the effectiveness of the SSDR extraction, the results may exhibit more noticeable unnatural creases in the skinning effect, when using the same number of virtual bones as in our method. In contrast to these methods, our approach excels at capturing dynamic details in clothing deformation, taking into account the motion sequence of the character. This enables us to generate more realistic and natural cloth deformations, resulting in an overall higher-quality simulation.

6.2. Quantitative Comparisons

Evaluating the accuracy and fidelity of the predictions of our network in comparison to prior approaches is crucial. It allows us to assess the performance and effectiveness of our network, and to determine its advantages over existing methods.

Metrics such as point-to-point distance error and other relevant metrics can provide quantitative measurements of the similarity between predicted cloth deformations and ground truth data. Thus we utilize the following error metrics to evaluate and compare the prediction results of our network with other methods:

$$\begin{aligned} \mathcal{E}_d &= \frac{1}{N}\sum_{i=1}^{N}\left\|x_p^i - x_g^i\right\|, \\ \mathcal{E}_n &= \frac{1}{N}\sum_{i=1}^{N}\arccos\left(\frac{(n_p^i)^T n_g^i}{\|n_p^i\|\,\|n_g^i\|}\right), \end{aligned} \tag{19}$$

where x_p^i is the predicted position of vertex i. x_g^i is the ground truth. The position evaluation metric $\mathcal{E}_d$ measures the discrepancy between the predicted results and the ground truth by calculating the point-to-point distance error between x_p^i and x_g^i. The provided normal evaluation metric $\mathcal{E}_n$ is used to assess the similarity of the normal vectors n_p^i and n_g^i between the predicted mesh and the ground truth.

We analyze the prediction errors for our method as well as previous methods using the mentioned evaluation metrics. The results are summarized in Tab. 3, which presents the mean and variance of the prediction errors across different methods.

Table 3: *We compare the mean and standard deviations of prediction errors between different methods.*

Evaluation	mean $\mathcal{E}_d$(m)	std $\mathcal{E}_d$(m)	mean $\mathcal{E}_n$(°)	std $\mathcal{E}_n$(°)
PBNS	7.401E-2	1.068E-2	35.72	7.24
CTSN	3.055E-2	1.055E-2	25.52	3.58
NeuralClothSim	5.429E-2	1.508E-2	32.14	7.41
VirtualBones	3.562E-2	1.509E-2	20.44	4.07
Our Method	5.889E-3	1.891E-3	7.39	1.69

Compared with these metrics across different methods, it can be concluded that the prediction results of our network have a higher level of accuracy. The lower mean error and variance values indicate that our method consistently produces more precise predictions.

6.3. Ablation Experiments

To validate our network architecture, we conducted ablation experiments by modifying or removing specific components or stages within the network.

Fig. 11 provides the results of our ablation Experiments. The ground truth is shown as a reference in Fig. 11 (a). We then removed the learned joint increments in the static stage, leaving only the optimization of skinning weights. The prediction results of the static stage without joint increments are presented in Fig. 11 (b). It can be observed that without joint increments, the quality of

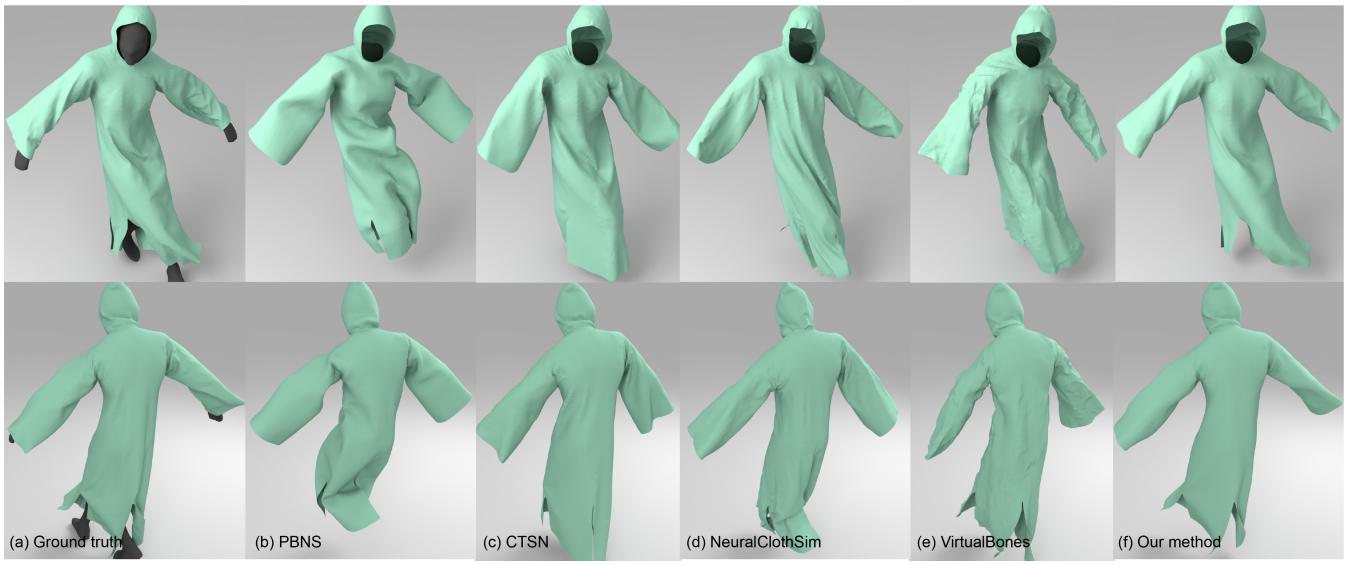

Figure 10: *Comparison of results between our network and prior methods: The first column is the ground truth of the dynamic cloth. The second and third columns are the static results of [BME21] and [LTY*23]. The fourth and fifth columns are the dynamic results of [PMJ*22] and [BME22]. The last column is the result of our method.*

the static skinning results deteriorates, indicating the importance of incorporating learned joint increments for accurate predictions. Furthermore, we removed the skinning weight increments in the static stage, using fixed clothing skinning weights instead. The static skinning results obtained using only fixed skinning weights are depicted in Fig. 11 (c). It is evident that relying solely on fixed skinning weights leads to poor static deformation results. Fig. 11 (d) and Fig. 11 (e) showcase the coarse dynamic and wrinkle dynamic prediction results of our network, respectively. Our network effectively adds coarse dynamics and wrinkle dynamics to the static skinning model, resulting in more realistic and detailed deformations. Lastly, Fig. 11 (f) illustrates the dynamic deformation predicted by our network with fixed skinning weights. Although the predicted results incorporate dynamic information, the presence of fixed skinning weights introduces artifacts and inaccuracies.

7. Conclusions

We present a three-stage network to predict the dynamic deformation of cloth dressed on skeleton-based characters. Our method uses three stages to generate the final prediction. The static stage learns the static cloth deformation, while the skeleton sequence stage and the mesh sequence stage learn to add coarse and wrinkle dynamic components respectively to the static skinning model. The deployment of our network in Unity showcases its immense potential and significance across a wide range of applications. By integrating our network into the Unity environment, we are able to leverage its capabilities in real-time simulations, virtual reality experiences, video games, and other interactive multimedia platforms.

Similar to previous learning-based methods, our network also faces challenges in achieving collision-free predictions. The accurate simulation of cloth interacting with complex objects or characters in real-time is a complex task, and ensuring collision-free predictions is an ongoing research challenge. To address this issue, we plan to explore learning-based collision handling methods in our future work.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China under Grant No.: 61972341, Grant No.: 61972342, Grant No.: 61832016, and the Tencent-Zhejiang University joint laboratory.

References

[BFA02] BRIDSON, ROBERT, FEDKIW, RONALD, and ANDERSON, JOHN. "Robust Treatment of Collisions, Contact and Friction for Cloth Animation". *ACM Trans. Graph.* 21.3 (2002), 594–603 2, 3.

[BKH16] BA, JIMMY LEI, KIROS, JAMIE RYAN, and HINTON, GEOFFREY E. "Layer normalization". *arXiv preprint arXiv:1607.06450* (2016) 6.

[BME21] BERTICHE, HUGO, MADADI, MEYSAM, and ESCALERA, SERGIO. "PBNS: physically based neural simulation for unsupervised garment pose space deformation". *ACM Transactions on Graphics (TOG)* 40.6 (2021), 1–14 3, 10, 11.

[BME22] BERTICHE, HUGO, MADADI, MEYSAM, and ESCALERA, SERGIO. "Neural Cloth Simulation". *ACM Transactions on Graphics (TOG)* 41.6 (2022), 1–14 2, 3, 10, 11.

[BML*14] BOUAZIZ, SOFIEN, MARTIN, SEBASTIAN, LIU, TIANTIAN, et al. "Projective Dynamics: Fusing Constraint Projections for Fast Simulation". *ACM Trans. Graph. (SIGGRAPH)* 33.4 (July 2014), 154:1–154:11. ISSN: 0730-0301 2.

[BMTE21] BERTICHE, HUGO, MADADI, MEYSAM, TYLSON, EMILIO, and ESCALERA, SERGIO. "DeePSD: Automatic deep skinning and pose space deformation for 3D garment animation". *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, 5471–5480 2.

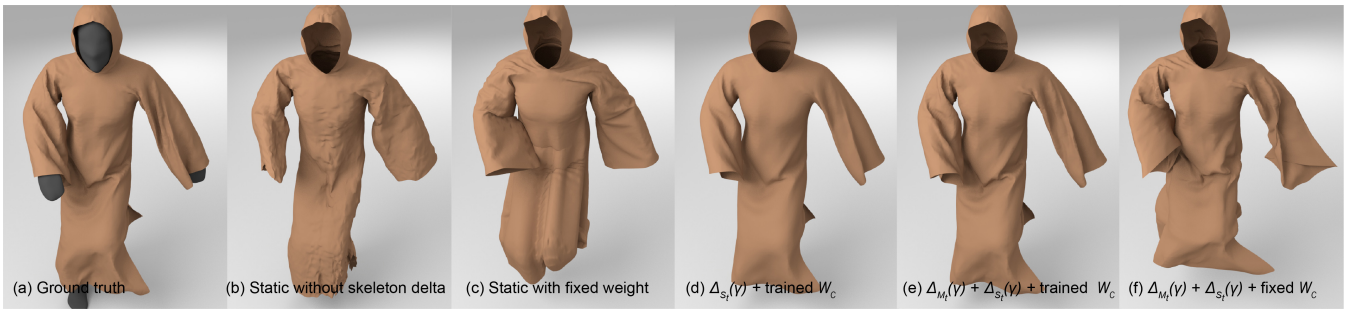

Figure 11: *The ablation experiments of our three-stage network: We removed the learned joint increments and skinning weight increments respectively to show the network prediction. The coarse and wrinkle dynamic predictions of our network are also shown.*

[BW98] BARAFF, DAVID and WITKIN, ANDREW. "Large steps in cloth simulation". *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 1998, 43–54 2.

[CVG*14] CHO, KYUNGHYUN, VAN MERRIËNBOER, BART, GULCEHRE, CAGLAR, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". *arXiv preprint arXiv:1406.1078* (2014) 2, 5, 6.

[GCP*20] GUNDOGDU, ERHAN, CONSTANTIN, VICTOR, PARASHAR, SHAIFALI, et al. "GarNet++: Improving Fast and Accurate Static 3D Cloth Draping by Curvature Loss". *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020) 3.

[GCS*19] GUNDOGDU, ERHAN, CONSTANTIN, VICTOR, SEIFODDINI, AMROLLAH, et al. "GarNet: A two-stream network for fast and accurate 3d cloth draping". *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, 8739–8748 3.

[HDDN19] HOLDEN, DANIEL, DUONG, BANG CHI, DATTA, SAYANTAN, and NOWROUZEZAHRAI, DEREK. "Subspace neural physics: Fast data-driven interactive simulation". *Proceedings of the 18th annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2019, 1–12 3.

[HLB*23] HALIMI, OSHRI, LARIONOV, EGOR, BARZELAY, ZOHAR, et al. "PhysGraph: Physics-Based Integration Using Graph Neural Networks". *arXiv preprint arXiv:2301.11841* (2023) 3, 8.

[HVTG08] HARMON, DAVID, VOUGA, ETIENNE, TAMSTORF, RASMUS, and GRINSPUN, EITAN. "Robust Treatment of Simultaneous Collisions". *ACM Trans. Graph.* 27.3 (2008), 23:1–23:4 2, 3.

[HZRS15] HE, KAIMING, ZHANG, XIANGYU, REN, SHAOQING, and SUN, JIAN. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". *Proceedings of the IEEE international conference on computer vision*. 2015, 1026–1034 6.

[KB14] KINGMA, DIEDERIK P and BA, JIMMY. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980* (2014) 7.

[KCŽO07] KAVAN, LADISLAV, COLLINS, STEVEN, ŽÁRA, JIŘÍ, and O'SULLIVAN, CAROL. "Skinning with dual quaternions". *Proceedings of the 2007 symposium on Interactive 3D graphics and games*. 2007, 39–46 3.

[KW16] KIPF, THOMAS N and WELLING, MAX. "Semi-supervised classification with graph convolutional networks". *arXiv preprint arXiv:1609.02907* (2016) 3.

[LD12] LE, BINH HUY and DENG, ZHIGANG. "Smooth skinning decomposition with rigid bones". *ACM Transactions on Graphics (TOG)* 31.6 (2012), 1–10 2.

[LMR*15] LOPER, MATTHEW, MAHMOOD, NAUREEN, ROMERO, JAVIER, et al. "SMPL: A skinned multi-person linear model". *ACM transactions on graphics (TOG)* 34.6 (2015), 1–16 2, 3.

[LTT*20] LI, CHENG, TANG, MIN, TONG, RUOFENG, et al. "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 (2020), 1–15 3.

[LTY*23] LI, YUDI, TANG, MIN, YANG, YUN, et al. "CTSN: Predicting Cloth Deformation for Skeleton-based Characters with a Two-stream Skinning Network". *arXiv preprint arXiv:2305.18808* (2023) 10, 11.

[LZBT16] LI, YUJIA, ZEMEL, RICHARD, BROCKSCHMIDT, MARC, and TARLOW, DANIEL. "Gated Graph Sequence Neural Networks". *Proceedings of ICLR'16*. 2016 6.

[MLTM88] MAGNENAT-THALMANN, NADIA, LAPERRIRE, RICHARD, THALMANN, DANIEL, and MONTRÉAL, UNIVERSITÉ DE. "Joint-Dependent Local Deformations for Hand Animation and Object Grasping". *In Proceedings on Graphics interface '88*. 1988, 26–33 4.

[NPO13] NARAIN, RAHUL, PFAFF, TOBIAS, and O'BRIEN, JAMES F. "Folding and crumpling adaptive sheets". *ACM Transactions on Graphics (TOG)* 32.4 (2013), 1–8 7.

[NSO12] NARAIN, RAHUL, SAMII, ARMIN, and O'BRIEN, JAMES F. "Adaptive anisotropic remeshing for cloth simulation". *ACM transactions on graphics (TOG)* 31.6 (2012), 1–10 7.

[PFSB20] PFAFF, TOBIAS, FORTUNATO, MEIRE, SANCHEZ-GONZALEZ, ALVARO, and BATTAGLIA, PETER. "Learning Mesh-Based Simulation with Graph Networks". *International Conference on Learning Representations*. 2020 8.

[PLP20] PATEL, CHAITANYA, LIAO, ZHOUYINGCHENG, and PONS-MOLL, GERARD. "TailorNet: Predicting clothing in 3d as a function of human pose, shape and garment style". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, 7365–7375 2, 3.

[PMJ*22] PAN, XIAOYU, MAI, JIAMING, JIANG, XINWEI, et al. "Predicting loose-fitting garment deformations using bone-driven motion networks". *ACM SIGGRAPH 2022 Conference Proceedings*. 2022, 1–10 2, 3, 10, 11.

[PNDO14] PFAFF, TOBIAS, NARAIN, RAHUL, DE JOYA, JUAN MIGUEL, and O'BRIEN, JAMES F. "Adaptive tearing and cracking of thin sheets". *ACM Transactions on Graphics (TOG)* 33.4 (2014), 1–9 7.

[Pro95] PROVOT, XAVIER. "Deformation Constraints in a Mass-spring Model to Describe Rigid Cloth Behavior". *Proc. of Graphics Interface*. 1995, 147–154 2.

[SOC19] SANTESTEBAN, IGOR, OTADUY, MIGUEL A., and CASAS, DAN. "Learning-Based Animation of Clothing for Virtual Try-On". *Computer Graphics Forum (Proc. Eurographics)* (2019). ISSN: 1467-8659. DOI: 10.1111/cgf.13643 2, 3.

[SOC22] SANTESTEBAN, IGOR, OTADUY, MIGUEL A, and CASAS, DAN. "Snug: Self-supervised neural dynamic garments". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, 8140–8150 3.

[STOC21] SANTESTEBAN, IGOR, THUEREY, NILS, OTADUY, MIGUEL A, and CASAS, DAN. "Self-Supervised Collision Handling via Generative 3D Garment Models for Virtual Try-On". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, 11763–11773 3.

[TTWM14] TANG, MIN, TONG, RUOFENG, WANG, ZHENDONG, and MANOCHA, DINESH. "Fast and Exact Continuous Collision Detection with Bernstein Sign Classification". *ACM Trans. Graph. (SIGGRAPH Asia)* 33 (6 Nov. 2014), 186:1–186:8 2.

[TWL*18] TANG, MIN, WANG, TONGTONG, LIU, ZHONGYUAN, et al. "I-Cloth: Incremental collision handling for GPU-based interactive cloth simulation". *ACM Transactions on Graphics (TOG)* 37.6 (2018), 1–10 2, 3.

[TWT*16] TANG, MIN, WANG, HUAMIN, TANG, LE, et al. "CAMA: Contact-Aware Matrix Assembly with Unified Collision Handling for GPU-based Cloth Simulation". *Computer Graphics Forum* (2016). ISSN: 1467-8659. DOI: 10.1111/cgf.12851 3.

[VSGC20] VIDAURRE, RAQUEL, SANTESTEBAN, IGOR, GARCES, ELENA, and CASAS, DAN. "Fully Convolutional Graph Neural Networks for Parametric Virtual Try-On". *Computer Graphics Forum (Proc. SCA)* (2020) 3.

[WSFM19] WANG, TUANFENG Y, SHAO, TIANJIA, FU, KAI, and MITRA, NILOY J. "Learning an intrinsic garment space for interactive authoring of garment animation". *ACM Transactions on Graphics (TOG)* 38.6 (2019), 1–12 3, 7.

[WWW22] WU, BOTAO, WANG, ZHENDONG, and WANG, HUAMIN. "A GPU-based multilevel additive schwarz preconditioner for cloth and deformable body simulation". *ACM Trans. Graph.* 41.4 (2022), 63:1–63:14 3.