

Self-Supervised Multi-Layer Garment Animation Generation Network

Guoqing Han¹ , Min Shi^{1†}, Tianlu Mao², Xinran Wang¹, Dengming Zhu², and Lin Gao²

¹North China Electric Power University

²Institute of Computing Technology, Chinese Academy of Sciences

Abstract

This paper presents a self-supervised multi-layer garment animation generation network. The complexity inherent in multi-layer garments, particularly the diverse interactions between layers, poses challenges in generating continuous, stable, physically accurate, and visually realistic garment deformation animations. To tackle these challenges, we present the Self-Supervised Multi-Layer Garment Animation Generation Network (SMLN). The architecture of SMLN is based on graph neural networks, which represents garment models uniformly as graph structures, thereby naturally depicting the hierarchical structure of garments and capturing the relationships between garment layers. Unlike existing multi-layer garment deformation methods, we model interaction forces such as friction and repulsion between garment layers, translating physical laws consistent with dynamics into network constraints. We penalize garment deformation regions that exceed these constraints. Furthermore, instead of the traditional post-processing method of fixed vertex displacement calculation for handling collision interactions, we add an additional repulsion constraint layer within the network to update the corresponding repulsive force acceleration, thereby adaptively managing collisions between garment layers. Our self-supervised modeling approach enables the network to learn without relying on garment sample datasets. Experimental results demonstrate that our method is capable of generating visually plausible multi-layer garment deformation effects, surpassing existing methods in both visual quality and evaluation metrics.

Keywords: Multi-layer garments animation; Computer Graphics; Self-supervised Learning; Garment deformation;

CCS Concepts

• *Computing methodologies* → *Animation*;

1. Introduction

Animating 3D garments plays a significant role in various domains such as computer animation, gaming, and virtual reality. The quality of garment animations often profoundly impacts the visual experience of an entire animation scene. Although physical simulation can generate reliable garment deformation effects, this method is complex, time-consuming, computationally expensive, and inefficient. Additionally, it often requires animators to manually adjust interpenetrations between multi-layer garments.

Learning-based methods [WSFM19] [SOC19] [PLPM20] can efficiently and quickly generate visually plausible garment deformation effects. However, they often employ data-driven approaches that rely on a large amount of high-quality paired garment sample data for network training to fit the ground truth of garment vertices. Existing multi-layer garment deformation methods [SOTC22] [LGF24] [LKL23] could generate static draping deformations of complex multi-layer garments, but such methods

struggle to capture the physical laws during motion and are not suitable for handling complex interactions between multi-layer garments. Shao et al. [SLD23] proposed a method capable of handling dynamic deformation of multi-layer garments. Their approach is based on training with garment sample datasets generated from physical simulations, and they convert UV patches into particle representations for particle simulation. The network effectively learns the physical laws governing particle motion. However, as the number of garment layers increases, the quantity of sample datasets and the computational cost of the network also significantly increase.

Using self-supervised methods to set constraints for garment deformation, thereby implicitly learning the physical laws of garment deformation, has become a hot research topic in the field of garment animation. However, existing self-supervised garment deformation methods [GBH23] [SOC22] [BME22] mostly focus on learning for single-layer garments, making it difficult to handle the complex interactions among multiple layers of garments during motion. A few works that do not require sample data can be applied to work with multi-layer garments, such as the method proposed by Bertiche [BME20] et al., employs self-supervised approaches to constrain the geometric features of garments. This method can

† Corresponding author: Min Shi. Email: shi_min@ncepu.edu.cn

generate garment deformation effects that fit human body movements without requiring garment sample data. However, they do not consider the complex mechanical constraints between multiple layers of garments, making it difficult to generate dynamic garment deformation effects.

To address these challenges, we present a self-supervised multi-layer garment animation generation method based on a graph neural network architecture [PFSGB20]. We represent the multi-layer garment model as a graph structure, where different layers are connected by virtual edges, facilitating the network's learning of layers relationships and dynamic interactions between garment layers. Using self-supervised learning, we convert dynamic equations adhering to physical laws and geometric constraints into target loss functions for the network to constrain garment deformation. Areas of garment deformation that exceed these constraints are penalized, thereby updating the vertex features of the graph. Additionally, we have incorporated a repulsion constraint layer into the network to simulate collision and contact between layers of garments during motion, ultimately generating garment deformation animations that adhere to physical laws and achieve visual realism. We conducted a series of experiments based on our network architecture and performed qualitative and quantitative analyses comparing our results with state-of-the-art garment deformation methods. We present these comparison results in this paper.

The main contributions of this paper are as follows:

(1) We present a multi-layer garment animation generation network based on Graph Neural Networks. Virtual edges are constructed to connect multi-layer garment graphs, facilitating the learning of interactions between garment layers. Additionally, a repulsion constraint layer is added to the network to handle collision and contact between layers of garments during motion.

(2) we set up a series of loss terms based on self-supervised garment geometric information and repulsive force between multi-layer garments, implicitly learns the deformation law of multi-layer garments, and generates the deformation effect of garments that fit the human body's movements without the need of garment sample data.

2. Related work

2.1. Physics-based simulations

Physics-based simulation methods, by establishing dynamic models [P*95] [VSC01] and solving large sets of dynamic equations, can achieve realistic and detailed garment simulation effects. Physics simulation constantly analyzes the forces acting on garments [TWT*16] [DBDC15] to generate garment deformations.

Vassilev et al. [VSC01] proposed a spring-mass model based on velocity direction correction and collision detection in image space. In each iteration, they check for collisions and the elongation of springs, correcting them when their length exceeds a pre-defined threshold to limit stretching. Fierz et al. [FSH11] proposed a numerical integration method for stable dynamic simulation of objects. Despite these methods generating high-precision garment deformation simulations, they require high computational and time costs. As the number of garment layers increases, the efficiency

of simulation decreases further. Tang et al. [TWT*16] proposed a GPU-based cloth simulation method, while Russell Gillette et al. [GPV*15] introduced coarse-to-fine approaches aimed at enhancing simulation efficiency. However, when simulating deformation across multiple layers of clothing, each layer's dynamic equation system must be independently recalculated, necessitating significant time and computational costs.

2.2. Learning-based models

Santesteban et al. [SOTC22] conducted early research on the un-tangling of multi-layer garments, but it could only be achieved in standard postures and required pre-processing to learn each garment, making it challenging to apply to dynamic scenarios. Li et al. [LGF24] proposed ISP, representing garment models using UV maps of 2D patches and implicit 2D signed distance field (SDF) values to avoid the problem of SDF being unsuitable for 3D open surfaces, thereby addressing the static draping problem of multi-layer garments. Lee et al. [LKL23] introduced the ClothCombo method, which predicts vertex features for each vertex and uses a Graph Neural Network (GNN) to simulate interactions between garments. This method appropriately deforms garments to prevent interpenetration and effectively simulates the static draping effects of complex multi-layer garments. However, these methods do not consider the temporal aspects of garment deformation, making it difficult to generate dynamic garment deformation effects.

In the realm of garment animation deformation, Bertiche et al. [BMTE21] employ Graph Neural Networks to animate any garment, independent of its topology, vertex order, or connectivity. They supplement supervised learning with implicit constraints to handle collision penetration, generating dynamic garment deformation effects through a combination of ground truth and implicit constraints. The Swingar method [LSZK23] proposed by Li et al. also uses a graph neural network structure and incorporates spectral information to focus on the details of garment wrinkles. However, their network [BMTE21] [LSZK23] training still requires a large amount of sample data. Shao et al. [SLD23], on the other hand, utilize the rotational invariance and additivity of physical systems to capture and handle interactions between components within garments, between different garments, and with driving factors. Their method converts garment UV patches into particle representations for deformation simulation, reducing network computation while necessitating a large dataset of garment samples for training.

Using a self-supervised approach, constraints are set for garment deformation to implicitly learn the physical laws governing the deformation of multi-layer garments. Bertiche et al. [BME20] proposed a method based on self-supervised learning, adding collision loss constraints to address penetration issues in multi-layer garment deformation. However, their approach does not consider mechanical constraints between garments in the network, making it difficult to generate dynamic deformation effects. The PGN-Cloth method [PWL*23] proposed by Peng et al. also encounters the same challenges. The NCS method [BME22] builds upon the PBNS [BME20] framework by adding a temporal motion module to extract temporal motion characteristics of garment deformation, thereby generating continuous dynamic garment deformations. However, like the SNUG method [SOC22], NCS [BME22]

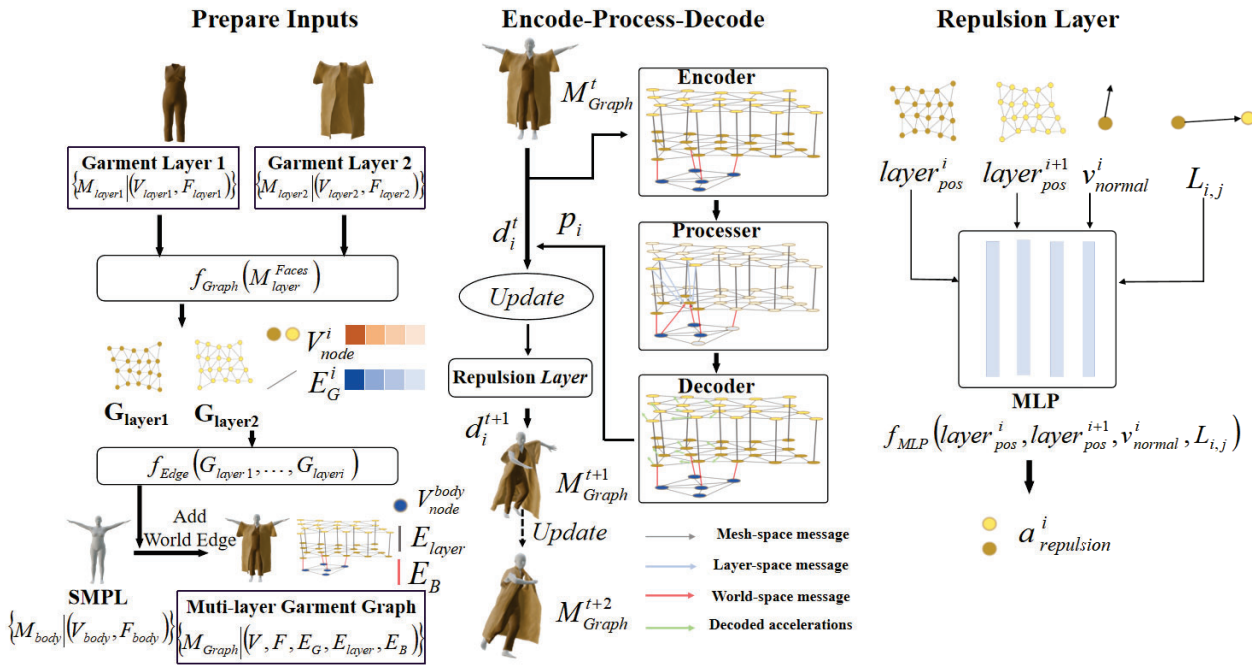


Figure 1: Overview of SMLN. In the Prepare Input stage, we process the garment mesh data. Using the $f_{Graph}(\cdot)$ function and the $f_{Edge}(\cdot)$ function, we convert each layer of the garment mesh into a graph representation and construct virtual edges for connection. This generates a multi-layer garment graph M_{Graph} and initializes vertex attributes in the graph. During the Encoder-Processor-Decoder stage, we process the graph vertex features, decode the dynamic changes of the garments, and generate additional acceleration changes for the vertices through the repulsion layer to handle interactions.

also defines all garment vertex weights using the SMPL [LMR*23] human body model and is trained specifically for single-layer garments. Grigorev et. al. [GBH23] adopts a graph neural network architecture and utilizes self-supervised constraints to learn the dynamics of garment deformation and its interaction with the human body. However, their method is unable to simulate interactions and collision penetration between garment layers, thus making it difficult to generate realistic multi-layer garment deformation effects. The ContourCraft method [GBB*24], based on a graph neural network, introduces a cross-contour loss to handle interpenetration between garments, thus generating visually plausible garment deformation effects.

In summary, some learning-based methods consider the penetration between multi-layer garments, but they typically rely only on sample vertex constraints or collision loss constraints to handle contact between different layers of garments. Our approach enhances this by introducing an additional repulsion constraint layer, which simulates the repulsive forces between garment layers. The network layer updates additional acceleration values for collision vertices, thereby simulating the interaction between different layers of garments.

3. Overview

We will provide a detailed description of the proposed multi-layer garment animation generation network. Based on a given human motion sequence and a multi-layer garment template in T-pose, the network predicts and generates multi-layer garment deformation

animations. Our network architecture is based on a Graph Neural Network and specifically includes three main components: the garment graph generation module, the Encode-Process-Decode graph network learning module, and the garment repulsion layer. The overall structure is illustrated in Fig. 1.

Given the multi-layer garment models $M_{layer1}, \dots, M_{layerm}$, each layer garment includes vertex coordinates V_{layer} and face information F_{layer} . Through the garment graph generation module, we convert each layer garment into a graph structure representation G_{layer} , which includes multi-dimensional vertex information and edge connectivity. Virtual garment edges are constructed to connect the garment graphs of each layer and the human body world, generating the multi-layer garment graph M_{Graph} .

The multi-layer graph M_{Graph} is input into the graph network learning module, where we construct geometric and frictional loss terms to learn dynamic features during the deformation process. Through the garment repulsion layer, the current layer's vertex information and adjacent layer's vertex information are used as inputs. This layer updates additional acceleration values for collision vertices, thereby simulating the interaction between different layers of garments.

3.1. Garment Graph Generation

The garment graph generation module primarily serves as the pre-processing part for network input data. In this module, we convert each layer of garment, which contain vertex coordinate information

V_{pos} and face information F_{layer} , into a graph representation structure. We initialize and add multi-dimensional information such as velocity, mass, friction coefficient, etc., on the vertices of the graph, thereby generating vertex attribute vectors as shown in Eq. 1-4.

$$G_{\text{layer}}^i = f_{\text{Graph}}(F_{\text{layer}}^i) \quad (1)$$

$$V_{\text{node}}^{\text{velocity}} = V_{\text{pos}}^{\text{cur}} - V_{\text{pos}}^{\text{prev}} \quad (2)$$

$$V_{\text{mass}} = \frac{D_{\text{fabric}} \times S_{\text{triangle}}}{3} \quad (3)$$

$$V_{\text{node}}^{\text{attribute}} = \left[V_{\text{node}}^{\text{velocity}}, V_{\text{mass}}, V_{\text{pos}}, \mu \right] \quad (4)$$

Where $f_{\text{Graph}}(\cdot)$ takes garment face information as input to generate the connectivity between graph vertices. We use a method based on human body skinning to generate the initial vertex positions $V_{\text{prev}}, V_{\text{curr}}, V_{\text{targ}}$ after the first three frames of rigid deformation for the garments, which are used to initialize the vertex velocity attributes. Each vertex's mass property V_{mass} is initialized by multiplying the vertex density D_{fabric} by the triangle area S_{triangle} and dividing by 3. In Eq. 4, μ represents the friction coefficient.

After generating each layer's garment graph, $f_{\text{Edge}}(\cdot)$ constructs virtual garment edges between the layers and world edges representing garment interactions with the human body. The values of the edge vectors are normalized and added as edge attribute features to the multi-layered graph. The specific formulas are shown in Eq. 5-8:

$$E_{\text{layer}}^{\text{index}} = \sum_{i=0}^{\text{layernum}} \text{connect} \left(G_{\text{layer}}^i, G_{\text{layer}}^{i+1} \right) \quad (5)$$

$$E_{\text{layer}}^{\text{attribute}} = \left\| x_{\text{v}}^i - x_{\text{v}}^j \right\|^2 \quad (6)$$

$$E_{\text{B}}^{\text{index}} = \text{connect} \left(G_{\text{Multi-layer}}, V_{\text{Body}} \right) \quad (7)$$

$$E_{\text{B}}^{\text{attribute}} = \left\| x_{\text{v}}^i - x_{\text{b}}^j \right\|^2 \quad (8)$$

Where $\text{connect}(\cdot)$ function connects the nearest neighboring vertex pairs between adjacent garment layers, generating edge-vertex pair indices $E_{\text{layer}}^{\text{index}}$, and calculates the Euclidean norm to add to the edge attributes $E_{\text{layer}}^{\text{attribute}}$. Similarly, it creates human body world edges $E_{\text{B}}^{\text{index}}$ and their edge attributes $E_{\text{B}}^{\text{attribute}}$. The garment graph generation module preprocesses the data, generating the multi-layer graph $\{M_{\text{Graph}} | (V, F, E_{\text{G}}, E_{\text{layer}}, E_{\text{B}})\}$.

3.2. Encode-Process-Decode Graph Network

This module's main task is to use the dynamic features d_i^t generated from the given current frame graph M_{Graph}^t and the previous frame graph M_{Graph}^{t-1} to predict the next frame's dynamic features d_i^{t+1} for M_{Graph}^{t+1} . The dynamic features d_i^t mainly consist of three parts, $\{d | (\text{position}, \text{velocity}, \text{acceleration})\}$. The processing flow of the Encoder-Processor-Decoder is shown in Fig. 1.

The encoder takes the current multi-layer graph M_{Graph}^t as input and encodes the vertex attributes generated by the garment graph generation module to generate vertex features. The edges in the graph are processed as follows: the mesh edges of each garment layer are encoded to generate $E_{\text{G}}^{\text{feature}}$, which captures internal forces driving deformation patterns within the garments. Virtual edges between garment layers are encoded to generate $E_{\text{layer}}^{\text{feature}}$, which represents hierarchical features used to learn interactional deformations between garment layers. Finally, world edges between the human body and garments are encoded to generate $E_{\text{B}}^{\text{feature}}$, which captures interactional deformation patterns due to collisions with the human body. This is represented as Eq.9-12.

$$V_{\text{node}}^{\text{feature}} = \text{Encoder}_{\text{node}} \left(V_{\text{node}}^{\text{attribute}} \right) \quad (9)$$

$$E_{\text{G}}^{\text{feature}} = \text{Encoder}_{\text{edge}}^{\text{mesh}} \left(E_{\text{G}}^{\text{attribute}} \right) \quad (10)$$

$$E_{\text{layer}}^{\text{feature}} = \text{Encoder}_{\text{edge}}^{\text{layer}} \left(E_{\text{layer}}^{\text{attribute}} \right) \quad (11)$$

$$E_{\text{B}}^{\text{feature}} = \text{Encoder}_{\text{edge}}^{\text{world}} \left(E_{\text{B}}^{\text{attribute}} \right) \quad (12)$$

Encoders $\text{Encoder}_{\text{node}}$, $\text{Encoder}_{\text{edge}}^{\text{mesh}}$, $\text{Encoder}_{\text{edge}}^{\text{layer}}$, and $\text{Encoder}_{\text{edge}}^{\text{world}}$ are composed of four independent MLP layers, which encode vertex features and different types of edge features through MLP encoding. The processor updates multi-layer graph features based on feature values, learning dynamic deformation of garments.

The processor consists of multiple independent message passing blocks. To accelerate the computation process, we introduce anchor points on the graph vertices. Based on these anchor points, we perform upsampling and downsampling of the graph to reduce computational complexity and assist in aggregating neighborhood information, thereby enhancing feature extraction capabilities. The processor then sends the aggregated and updated graph vertex features to the decoder for feature decoding.

The decoder consists of MLP that decodes the vertex features containing physical properties, predicting the acceleration change p_i of the vertices in the current frame of garments, thereby predicting the dynamic features d_i^{t+1} of the next frame. Since the vertex properties (such as velocity, $v = (v_{\text{pos}}^{\text{curr}} - v_{\text{pos}}^{\text{prev}}) / t$) are based on the distance difference between adjacent frames, the time step interval is $t = 1$. Without considering the influence of the repulsion force layer, the dynamic features of the garments in the next frame are shown in Eq. 13:

$$d_i^{t+1} = d_i^t + p_i \quad (13)$$

3.3. The Garment Repulsion Layer and Network Loss Settings

The garment repulsion layer simulates the effect of a thrust between garment layers to update additional acceleration values for collision vertices, adaptively handling collision contacts between multi-garment layers. The repulsion force network layer comprises two functional modules: the computation module and the $f_{\text{MLP}}(\cdot)$

prediction module. It uses $f_{\text{MLP}}(\cdot)$ to generate additional acceleration values for vertices that have experienced collision penetration, specifically as represented in Eq.14 and Eq.15. The inputs for the computation module are the vertex positions of Layer i , Layer $i+1$, and the accelerations predicted by f_{MLP} , which are used to update the vertex position information influenced by repulsive forces. After adding the additional acceleration, the dynamic characteristics of the next frame of garments are represented as shown in Eq.16.

$$E_r = f_{E_r}(L_{ij}, n) \quad (14)$$

$$a = f_{\text{MLP}}(|L_{ij}|, E_r) \quad (15)$$

$$d_i^{t+1} = d_i^t + p_i + a \quad (16)$$

Where, we use the edge vector L_{ij} to simulate repulsive forces, calculating the magnitude of the edge vector and its projection with the normal vector using $f_{E_r}(\cdot)$ to simulate the dynamic force variation E_r in the direction of the vertex normal.

The initial construction of the edge vector set requires a complete traversal, resulting in a time complexity of $O(N^2)$, assuming that each layer contains N vertices. However, during the collision handling process, we only deal with edge vectors that are below the threshold, with a time complexity of less than $O(N)$.

The norm of the edge vector $|L_{ij}|$ and E_r are then fed into $f_{\text{MLP}}(\cdot)$ to generate additional acceleration for the collided vertices.

Additionally, the network converts the geometric features of the garments and the dynamic equations into optimization objectives for the network. The loss function is shown in Eq.17:

$$L = \lambda_s L_{\text{str}} + \lambda_b L_{\text{bend}} + \lambda_g L_g + \lambda_f L_{\text{friction}} + \lambda_r L_{\text{repulsion}} \quad (17)$$

Where, L_{str} represents the stretch loss, L_{bend} denotes the curvature loss, L_g is the gravity loss, L_{friction} is the friction loss, and $L_{\text{repulsion}}$ is the repulsion loss, $\lambda_s, \lambda_b, \lambda_g, \lambda_f, \lambda_r$, corresponding to their weights. L_{str} and L_{bend} are geometric loss constraints that adhere to physical laws, where L_{str} enforces constraints on edge lengths between garment vertices and L_{bend} enforces constraints on the curvature of garment faces. Based on the self-supervised loss settings of HOOD [GBH23] and SNUG [SOC22], the network is structured with the stretch loss, as shown in Eq.18-19

$$E_{\text{density}} = \text{traces}\left(\left(S^T G_r\right)\right) \quad (18)$$

$$L_{\text{str}} = \sum_{\text{triangles}} F_{\text{area}} E_{\text{density}} \quad (19)$$

where F_{area} represents the triangle area, and E_{density} is the stretching energy density. The calculation of E_{density} is shown in Eq.18, where G_r denotes the Green strain tensor, which can be computed from the deformation gradient matrix of triangles. S^T is the linear transformation of the Green strain tensor G_r , reflecting the elastic stretching of garments.

Set the curvature constraint L_{bend} between garment faces to constrain the geometric features of the garment, as shown in Eq.20.

$$L_{\text{bend}} = \sum_{n=1}^{n_k} \|\Delta(F_n)\|^2 \quad (20)$$

Where $\Delta(\cdot)$ is the Laplacian operator, n_k represents the number of faces, and F_n is the normal vector of the face. L_{bend} constrains the garment patches to prevent unnatural bending deformations.

To enable the network to generate physically realistic multi-layer garment deformations, we set the gravity loss constraint L_g , friction loss constraint L_{friction} , and repulsion loss constraint $L_{\text{repulsion}}$ as follows in Eq. 21.

$$L_g = \lambda \cdot V_z \quad (21)$$

where $\lambda = g \cdot v_{\text{mass}}$, and V_z represents the vertical position of vertices in world space coordinates, used to calculate the gravitational potential energy of garment mesh points.

$$L_{\text{fri}} = \sum_{i=1}^{V_{\text{node}}} \mu V_{\text{mass}}^i g \cos \theta \cdot \text{distance} \quad (22)$$

In the friction loss constraint, $\mu = 0.5$ represents the friction coefficient, g denotes the gravitational acceleration, which are constant terms. V_{mass} represents the vertex mass, $\cos(\theta)$ denotes the cosine of the angle between the adjacent face normal and the vertex normal, indicating the direction of frictional force. distance represents the displacement of the vertex in the direction of the adjacent face normal, calculated between the current frame t and the previous frame $t-1$.

The repulsion loss $L_{\text{repulsion}}$ is used to prevent multi-layer garments from penetrating each other. It applies repulsive forces between vertex pairs from different layers to maintain a certain distance between vertices of different layers. The specific calculations are shown in Eq. 23-25.

$$d_{ij}^{(kl)} = x_i^{(k)} - x_j^{(l)} \quad (23)$$

$$f_{ij}^{(kl)} = \begin{cases} \left(\epsilon - d_{ij}^{(kl)}\right) \cdot \frac{x_i^{(k)} - x_j^{(l)}}{d_{ij}^{(kl)}} & d_{ij}^{(kl)} < \epsilon \\ 0, & d_{ij}^{(kl)} \geq \epsilon \end{cases} \quad (24)$$

$$L_{\text{repulsion}} = \sum_{k>l} \sum_i \sum_j \left(f_{ij}^{(kl)}\right)^3 \quad (25)$$

Where $d_{ij}^{(kl)}$ represents the distance between the i -th vertex of the k -th layer garment and the j -th vertex of the l -th layer garment. $x_i^{(k)}$ denotes the position of the i -th vertex of the k -th layer garment. $f_{ij}^{(kl)}$ indicates the magnitude of the repulsive force between vertices, where ϵ represents a threshold.

4. Experiment

Train Data. We train the network in a self-supervised manner by constructing loss terms that adhere to physical laws, hence requiring only human motion data for preparation. The human motion data needed for training is based on the SMPL parametric human body model representation. We extracted this data from the virtual try-on dataset [SOC19], selecting various movements such as walking, running, torso actions, and dancing, totaling 6,000 frames. Our

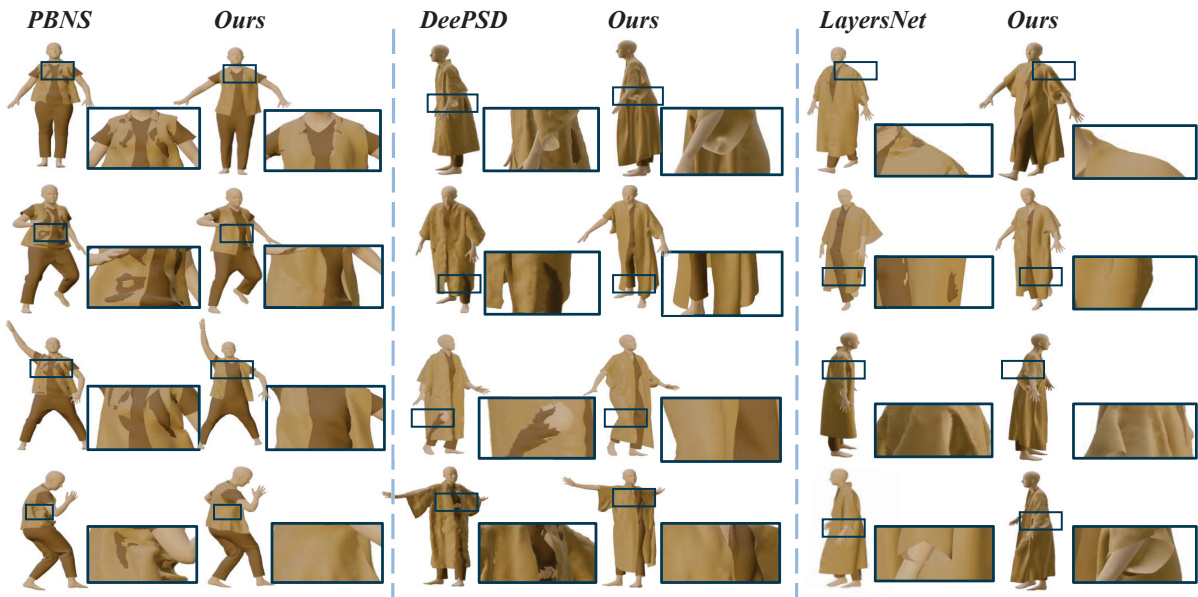


Figure 2: Qualitative comparisons with other garment animation generation methods, including the PBNS, DeePSD, and LayersNet methods. The PBNS method is only applicable to tight-fitting garments, so we use sleeveless tops that are on the tighter side for comparison. The comparison with DeePSD and LayersNet is made using more loose-fitting long shirts. It can be seen that our method is able to generate more reasonable garment deformation effects.

training dataset covers enough basic movements to meet common action requirements.

Implementation Details. In our network, both the encoder and decoder are composed of MLP. The vertex encoder has an input dimension of [24, 128], the garment edge encoder has an input dimension of [12, 128], and both the human body edge encoder and the garment virtual edge encoder have an input dimension of [9, 128]. The decoder MLP features have a dimension of [128, 3]. The repulsion layer consists of three MLP layers, each using ReLU as its activation function. The input dimension of this layer is set to [4, 32]. The network is configured with a learning rate of $lr = 5 \times 10^{-7}$ and a batch size of 1. In the network’s loss constraints, the stretch loss weight $\lambda_s = 1.0$, and the bend loss weight $\lambda_b = 1.0$. The gravity loss weight $\lambda_g = 1.0$, with a gravity acceleration $g = 9.81$. The friction loss weight $\lambda_f = 1.5$, with a fixed friction coefficient $\mu = 0.5$. The repulsion loss weight $\lambda_r = 1 \times 10^3$.

4.1. Comparison with Related Methods

To validate the superiority of our proposed method in generating garment deformation compared to existing methods, we use the test data for driving garment deformation prediction. We conduct both quantitative and qualitative analyses comparing our method with state-of-the-art methods suitable for multi-layer garments including PBNS [BME20], DeePSD [BMTE21], and LayersNet [SLD23].

The used evaluation metrics are L-Coll and H-Coll from LayersNet [SLD23], as shown in Tab.1. L-Coll assesses the penetration degree between different layers of garments, while H-Coll evaluates the penetration degree between garments and the human body. $Bend_{Loss}$ [BME20] is used to indicate the degree of bending of gar-

ment pieces, measuring whether the deformation of the garment is natural. The qualitative comparison results are illustrated in Fig.2.

Metrics	$Bend_{Loss} \downarrow$	$Collision_{Loss} \downarrow$	L-Coll (%) \downarrow	H-Coll (%) \downarrow
PBNS	0.45	1.01	11.77	0.44
DeePSD	1.09	2.45	2.51	11.39
LayersNet	0.51	0.92	1.08	0.23
Ours	0.57	0.18	0.99	0.11

Table 1: Quantitative Comparison Results.

From Tab. 1, it can be seen that our method outperforms the DeePSD method in all metrics. In comparison with PBNS and LayersNet, our method is similar in terms of the bending indicator of garment pieces, meaning it can generate reasonable bending deformations. Additionally, it surpasses PBNS and LayersNet in other penetration metrics. The results in Fig. 2 demonstrate the effectiveness of our method, showing that it can produce more reasonable deformation effects compared to other methods.

4.2. Generalization Experiment

To validate the network’s generalization performance, we select human motion sequences and unseen multi-layer garment styles, for predicting garment deformation. Both qualitative and quantitative analyses are conducted. The experimental results are shown in Tab. 2, and the garment deformation effects are illustrated in Fig. 3.

In Tab. 2, we select the $Collision_{Loss}$, L-coll, and H-coll metrics for quantitative analysis to measure whether the interlayer interactions of the network have good generalization capabilities and whether it can generate reasonable garment deformations for the unseen garments. From Tab. 2 and Fig. 3, it can be seen that our

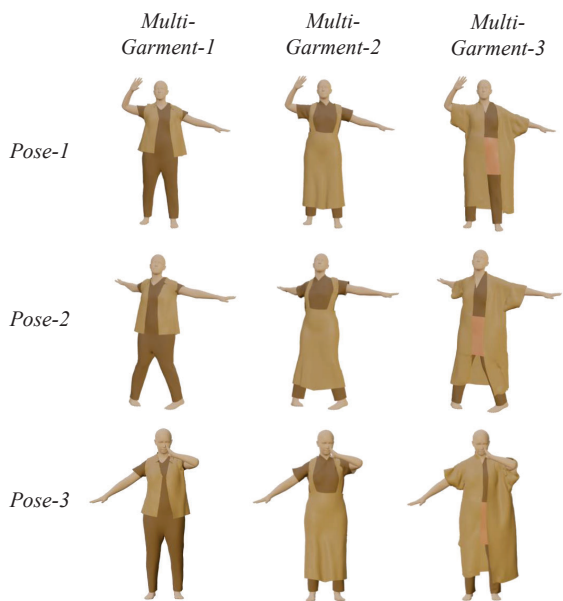


Figure 3: Generalization Results. Each row of the figure represents the garment deformation of different unseen garments under the same motion sequence, and each column represents the garment deformation of the unseen garments under different motion sequences.

Metrics	$Collision_{Loss} \downarrow$	L-Coll(%) \downarrow	H-Coll(%) \downarrow
Multi-Garment-1	0.25	0.22	0.04
Multi-Garment-2	0.24	0.19	0.02
Multi-Garment-3	0.84	0.92	0.06

Table 2: Quantitative Comparison Results for Unseen Garments.

method maintains a low garment penetration rate and human body penetration rate when applied to the unseen garments, producing reasonable garment deformation effects.

4.3. Ablation Study

To validate the effectiveness of the garment repulsion layer and the loss settings between multi-layer garments, we conducted ablation experiments and performed quantitative and qualitative analyses.

Metrics	$Collision_{Loss} \downarrow$	L-Coll(%) \downarrow	H-Coll(%) \downarrow
W/o loss and repulsion	3.95	5.51	0.05
W/o repulsion	2.03	1.92	0.02
Ours	0.69	0.95	0.02

Table 3: Quantitative Comparison Results for Ablation Study. The table presents the comparison results between the baseline method, the method with only the loss term constraint, and our method.

From Tab. 3, it can be seen that the first row shows the results without the garment repulsion layer and repulsion loss term. The second row shows the results with only the loss term constraint.

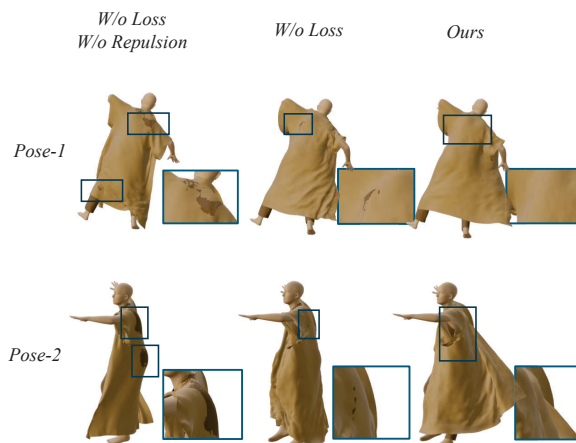


Figure 4: Ablation Study Results. The figure shows the comparison results of baseline, method that using only loss term constraints, and our method under different motion sequences. It can be seen that the loss term and garment repulsion layer are effective for us to simulate the multi-layer clothing deformation.

The last row shows the results with both the repulsion layer and the loss term constraint. From the results in Tab. 3 and Fig. 4, it is evident that compared to the baseline, both the garment penetration rate and human body penetration rate are further improved. The addition of the repulsion layer further reduces the penetration rate between garments.

4.4. Sample dressing animation and Runtime performance

The method presented in this paper is well-suited for predicting multi-layer garment deformation and generating animations. We have selected various types of human motion postures and used the method described in this paper for deformation simulation and animation generation. The results are demonstrated in Fig. 5, which displays animations of virtual characters dressed in different pose sequences. Please refer to the supplementary video for visual comparisons of dynamic deformation results.

Additionally, we compared the animation generation efficiency of the physics-based ARCSim method with our method. In our previous work, we compared the prediction speeds with ArcSim, which takes 3230 ms per frame in a 100-frame single-layer animation. Our method operates at 210 ms per frame in multi-layer animations, showing a significant improvement in efficiency over the ArcSim method.

5. Conclusion and discussion

This study investigates the generation of multi-layer garment animations using a self-supervised approach. Based on a graph neural network architecture and incorporating network constraints with physics-based loss functions, we learn the deformation patterns of



Figure 5: Example of SMLN for multi-layer garment animation.

garments under external forces such as friction and repulsion during motion. Our method effectively handles the complex interactions of multi-layer garments during movement, requires no garment sample data, and demonstrates strong generalization capabilities. In the final experimental section of this paper, we validate the effectiveness of our approach. Compared to other state-of-the-art methods, our network shows superiority in garment animation quality, penetration loss, and other metrics.

Limitations: While our method effectively learns collision interactions between layers of garments, it encounters unavoidable self-penetration issues among individual single-layer garments. Addressing and improving self-penetration problems within single-layer garments in a self-supervised manner will be a focus of our future research.

References

- [BME20] BERTICHE H., MADADI M., ESCALERA S.: Pbn: Physically based neural simulator for unsupervised garment pose space deformation. *arXiv preprint arXiv:2012.11310* (2020). 1, 2, 6
- [BME22] BERTICHE H., MADADI M., ESCALERA S.: Neural cloth simulation. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–14. 1, 2
- [BMTE21] BERTICHE H., MADADI M., TYLSON E., ESCALERA S.: Deepsd: Automatic deep skinning and pose space deformation for 3d garment animation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 5471–5480. 2, 6
- [DBDC15] DAVIET G., BERTAILS-DESCOUBES F., CASATI R.: Fast cloth simulation with implicit contact and exact coulomb friction. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2015), pp. 197–197. 2
- [FSH11] FIERZ B., SPILLMANN J., HARDERS M.: Element-wise mixed implicit-explicit integration for stable dynamic simulation of deformable objects. In *Proceedings of the 2011 ACM SIGGRAPH/eurographics symposium on computer animation* (2011), pp. 257–266. 2
- [GGB*24] GRIGOREV A., BECHERINI G., BLACK M., HILLIGES O., THOMASZEWSKI B.: Contourcraft: Learning to resolve intersections in neural multi-garment simulations. In *ACM SIGGRAPH 2024 Conference Papers* (2024), pp. 1–10. 3
- [GBH23] GRIGOREV A., BLACK M. J., HILLIGES O.: Hood: Hierarchical graphs for generalized modelling of clothing dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 16965–16974. 1, 3, 5
- [GPV*15] GILLETTE R., PETERS C., VINING N., EDWARDS E., SHEFFER A.: Real-time dynamic wrinkling of coarse animated cloth. In *Proceedings of the 14th ACM SIGGRAPH/eurographics symposium on computer animation* (2015), pp. 17–26. 2
- [LGF24] LI R., GUILLARD B., FUA P.: Isp: Multi-layered garment draping with implicit sewing patterns. *Advances in Neural Information Processing Systems* 36 (2024). 1, 2
- [LKL23] LEE D., KANG H., LEE I.-K.: Clothcombo: Modeling inter-cloth interaction for draping multi-layered clothes. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–13. 1, 2
- [LMR*23] LOPER M., MAHMOOD N., ROMERO J., PONS-MOLL G., BLACK M. J.: Smpl: A skinned multi-person linear model. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 2023, pp. 851–866. 3
- [LSZK23] LI T., SHI R., ZHU Q., KANAI T.: Swinger: Spectrum-inspired neural dynamic deformation for free-swinging garments. *IEEE Transactions on Visualization and Computer Graphics* (2023). 2
- [P*95] PROVOT X., ET AL.: Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In *Graphics interface* (1995), Canadian Information Processing Society, pp. 147–147. 2
- [PFSGB20] PFAFF T., FORTUNATO M., SANCHEZ-GONZALEZ A., BATTAGLIA P.: Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations* (2020). 2
- [PLPM20] PATEL C., LIAO Z., PONS-MOLL G.: Tailornet: Predicting clothing in 3d as a function of human pose, shape and garment style. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 7365–7375. 1
- [PWL*23] PENG T., WU W., LIU J., LI L., MIAO J., HU X., HE R., LI L.: Pgn-cloth: Physics-based graph network model for 3d cloth animation. *Displays* 80 (2023), 102534. 2
- [SLD23] SHAO Y., LOY C. C., DAI B.: Towards multi-layered 3d garments animation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 14361–14370. 1, 2, 6
- [SOC19] SANTESEBAN I., OTADUY M. A., CASAS D.: Learning-based animation of clothing for virtual try-on. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 355–366. 1, 5
- [SOC22] SANTESEBAN I., OTADUY M. A., CASAS D.: Snug: Self-supervised neural dynamic garments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 8140–8150. 1, 2, 5
- [SOTC22] SANTESEBAN I., OTADUY M., THUREY N., CASAS D.: Ulnet: Untangled layered neural fields for mix-and-match virtual try-on. *Advances in Neural Information Processing Systems* 35 (2022), 12110–12125. 1, 2
- [TWT*16] TANG M., WANG H., TANG L., TONG R., MANOCHA D.: Cama: Contact-aware matrix assembly with unified collision handling for gpu-based cloth simulation. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 511–521. 2
- [VSC01] VASSILEV T., SPANLANG B., CHRYSANTHOU Y.: Fast cloth animation on walking avatars. In *Computer Graphics Forum* (2001), vol. 20, Wiley Online Library, pp. 260–267. 2
- [WSFM19] WANG T. Y., SHAO T., FU K., MITRA N. J.: Learning an intrinsic garment space for interactive authoring of garment animation. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–12. 1