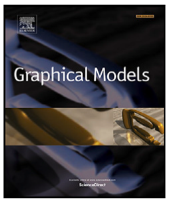
Graphical Models 139 (2025) 101273

Contents lists available at ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod



FastClothGNN: Optimizing message passing in Graph Neural Networks for accelerating real-time cloth simulation

Yang Zhang, Kailuo Yu, Xinyu Zhang *

Software Engineering Institute, East China Normal University, Shanghai, China
Shanghai Key Laboratory of Trustworthy Computing, Shanghai, China
Engineering Research Center for Software/Hardware Co-design Technology and Application (MoE), Shanghai, China

ARTICLE INFO

Keywords:
Cloth simulation
Virtual try-on
Graph Neural Networks
Message passing

ABSTRACT

We present an efficient message aggregation algorithm FastClothGNN for Graph Neural Networks (GNNs) specifically designed for real-time cloth simulation in virtual try-on systems. Our approach reduces computational redundancy by optimizing neighbor sampling and minimizing unnecessary message-passing between cloth and obstacle nodes. This significantly accelerates the real-time performance of cloth simulation, making it ideal for interactive virtual environments. Our experiments demonstrate that our algorithm significantly enhances memory efficiency and improve the performance both in training and in inference in GNNs. This optimization enables our algorithm to be effectively applied to resource-constrained, providing users with more seamless and immersive interactions and thereby increasing the potential for practical real-time applications.

1. Introduction

Cloth simulation has many applications in video games, virtual try-on and fashion design. Over the past few decades, many techniques have been proposed for cloth deformation and simulation. Physics-based approaches are commonly used to animate plausible cloth dynamics [1–6]. Although realistic deformation can be achieved, their high computational costs limit their application in real-time scenarios. Recently, learning-based methods have shown great potential for achieving fast cloth animation [7–11].

Some learning-based methods generate cloth deformation using body poses and model skinning [12,13]. For tight-fitting garments, these methods can achieve plausible results, but have difficulties in handling loose-fitting garments, such as dresses and skirts. Some methods are trained for specific garments [12,14,15], showing weak generalization and applicability. The work in [16–18] tackled these problems using Graph Neural Networks (GNNs). GNNs use local transformation, generate messages, update feature vectors and propagate signals through the mesh. GNN-based methods exhibit satisfactory performance, ensuring realistic simulation for both tight-fitting and loose-fitting garments under various motions. However, GNN-based methods also encounter challenges, such as increased memory demands, which limit batch sizes during training and result in longer training and inference times. The increased inference time diminishes some of the advantages that GNNs have over traditional physics-based methods, especially in real-time interactive applications where quick and smooth responses are crucial. Our goal is to strike a balance between accuracy and efficiency, enabling virtual try-on systems to provide visually realistic results with real-time performance [9,19].

Main Results: In this paper, we aim to enhance the scalability of GNN-based methods, making them suitable for real-time virtual try-on systems. We propose a novel method to accelerate the training and inference processes of GNNs. Our algorithm ensures that each node efficiently aggregates relevant information from its neighborhood, reducing redundant computations, and simplifying the message-passing process. This not only accelerates the training and inference processes, but also enhances the scalability and feasibility of our method for real-time applications. By reducing redundant computations while maintaining model performance, our method represents a significant advancement in the efficiency and effectiveness of GNNs for cloth simulation and virtual try-on systems. Our GNN-based algorithm can accurately predict the deformations of various garments, enabling immersive and responsive virtual try-on experiences.

2. Related work

2.1. Physics-based simulation

For decades, physics-based cloth simulation has captivated graphics researchers and developers due to its potential applications in

* Corresponding author.
E-mail address: xyzhang@sei.ecnu.edu.cn (X. Zhang).

https://doi.org/10.1016/j.gmod.2025.101273
Received 30 December 2024; Received in revised form 9 May 2025; Accepted 19 May 2025
Available online 8 June 2025

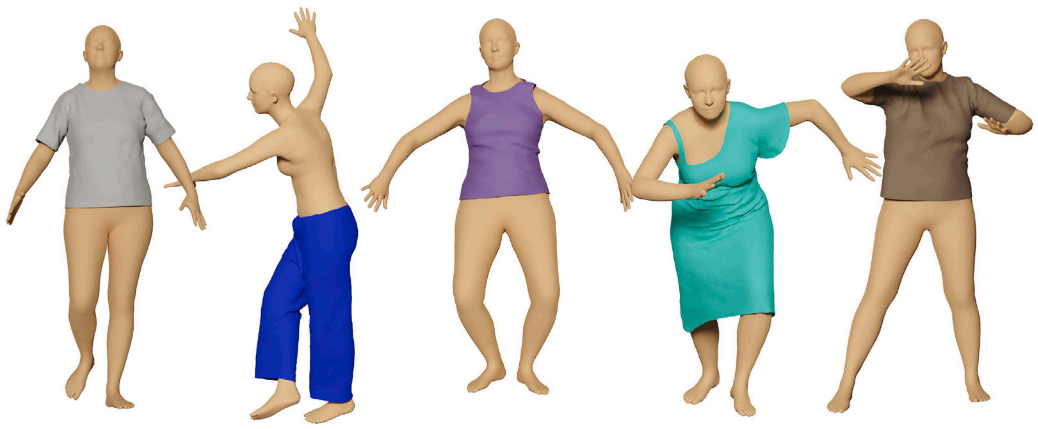


Fig. 1. Our method generates compelling results on unseen poses. Our system and algorithm optimize message passing and aggregation in GNNs specially designed for real-time cloth simulation in fashion and virtual try-on. Our extensive experiments demonstrate that the new algorithm can accurately predict the deformation behaviors for a variety of garments during real-time inference, while significantly reducing computational resources in practical VR applications.

entertainment and fashion. The pioneering work in [1] introduced an implicit integration method that allows for stable cloth simulation with large time steps. Researchers have proposed various models to represent cloth deformations, including mass–spring models [2,5], continuum-based models [1,3,20,21], and yarn-based models [22,23]. These methods often produce highly realistic simulations, generalize to different garments, and handle body-garment collisions. However, they fail to achieve the combined robustness and performance required for real-time applications. Position-based dynamics [4,5,24–26] can achieve faster simulations. In recent years, many studies utilized GPUs to overcome the computational bottlenecks of cloth simulation [27–29]. However, the high cost of high-performance GPUs also limits their application. Thus, while many algorithms provide realistic and accurate solutions, achieving efficient real-time cloth simulation remains an open challenge.

2.2. Learning-based methods

In recent years, learning-based methods have proven their feasibility in real-time cloth simulation [9,13,15,19,30,31]. Some existing methods rely on supervised learning [8,13,16,32,33], training with large dataset of physics-based simulation. The work in [8,34] fused the features of body and garments, making predictions based on pose and generating the result with linear blend skinning (LBS). The influence of three factors on cloth deformation: pose, shape, and style (garment geometry) are studied in [13]. Their reliance on skinning limits their ability to handle loose-fitting garments. Recently, virtual bone-driven motion networks [14] was used to generate loose garment deformation. [35] proposed a joint-specific feature learning approach to mitigate spurious correlations in loose-fitting garment animation, improving animation quality. Loose garments were handled by learning a generative space of plausible garment geometries and mapping motion-dependent dynamic deformations [36]. [37] introduced a neural-network-based approach for simulating loose-fitting garments by combining physics-based rope chain simulations with neural skinning and shape inference. [38] proposed a neural garment dynamic super-resolution method that enhances low-resolution garment simulations by learning super-resolution features from coarse garment dynamics and body interactions to generate high-frequency wrinkle details. Although these methods achieved good results for specific tasks, their resource-intensive nature hinders their practical application for large-scale scenarios.

A pioneer work, PBN [39], used an unsupervised method to learn garment deformations. PBN casts garment deformation as an energy optimization problem, deriving losses based on physics terms. SNUG [15] built upon PBN by incorporating elements that learn dynamic behavior of garment. SNUG uses previous states to predict cloth deformation on the target body pose in a self-supervised way. The work in [40] disentangles static and dynamic cloth subspace to control over the level of motion in predictions. Despite these methods not requiring real-world data, they can only predict deformations for specific garments and are limited to handling tight-fitting clothes. This also limits their scalability.

Methods based on Graph Neural Networks(GNNs) garnered significant attention from researchers. MeshGraphNets [16] employed a message-passing network to learn the dynamics of physical systems, including fluids and cloth, from mesh-based simulations. [41] introduces a bi-stride pooling strategy to effectively handle long-range interactions in large-scale mesh-based physical systems. HOOD [17,18] extended this approach specifically to cloth simulation, employing a self-supervised method to learn cloth deformations. Leveraging the generalization capabilities of GNNs, HOOD can handle a variety of garments and predict the deformations of loose-fitting clothes, such as skirts. Despite the feasibility of being applied in cloth simulation, GNNs require extensive computations, posing significant challenges regarding computational resources and memory demands. HOOD, for instance, experiences a drastic slowdown in inference when dealing with complex cloth meshes, particularly when the number of mesh nodes exceeds ten thousands, which limits its real-time application.

Recently, many GNN works that focus on using sub-sampled graph to increase the efficiency of message passing. GraphSAGE [42] utilizes a sampling and aggregation approach, where each node aggregates feature information from a fixed-size sample of its neighbors, enabling efficient and scalable node representation learning. [43] leverages random walks to sample neighbors and combines them with graph convolutions for scalable recommendation tasks. [44] divides the large graph into smaller, densely connected subgraphs (clusters) and performs training on these subgraphs.

In this paper, we analyze the message-passing mechanism of MeshGraphNets and identify a substantial amount of redundant message exchanges. Building on this insight, we propose an optimized message-passing method that enhances both training and inference speeds while significantly reducing memory requirements. We adopt an unsupervised training strategy, drawing inspiration from SNUG and HOOD, which obviates the need for supervised labels.

3. Algorithm overview and preliminary

An overview of our algorithm is given in Fig. 2. We use MeshGraphNets [16], an encoder-processor-decoder architecture for simulating mesh deformations. Here, we provide some preliminary on MeshGraphNets and graph neural networks.

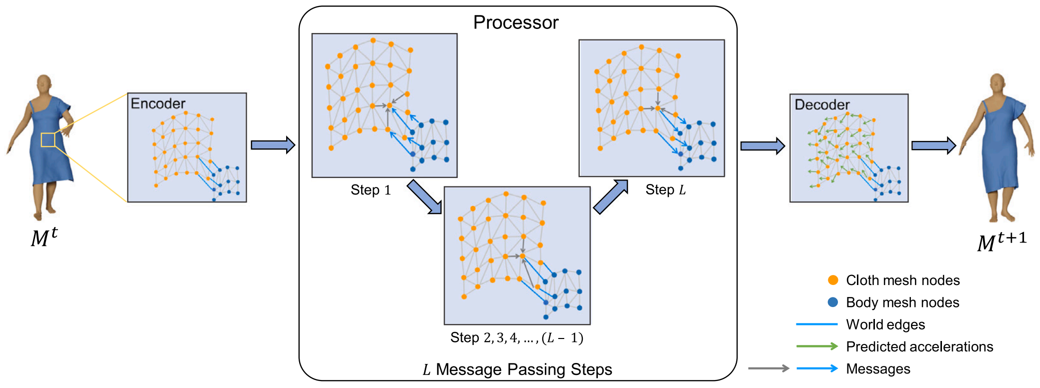


Fig. 2. An overview of our encoder-processor-decoder architecture. The encoder converts the features of nodes and edges in a given cloth mesh into a graph. The processor updates these features through multiple message-passing iterations, during which neighboring nodes for information transmission are determined via random sampling. Message exchanges between cloth nodes and body nodes occur only during the first and last message-passing iterations. The decoder then interprets the features of each node to determine their acceleration. We train the model using self-supervised learning by optimizing the potential energy.

MeshGraphNets encode a mesh and its simulation state into a graph. Then through message passing, it can predict the acceleration of each node in the mesh. Let a mesh $\mathcal{M}$ (garment and body) be encoded into a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}^{\mathcal{M}}, \mathcal{E}^{\mathcal{W}})$, where $\mathcal{V}$ are graph nodes (i.e., corresponding to the garment mesh nodes), $\mathcal{E}^{\mathcal{M}}$ are graph edges (i.e., corresponding to the garment mesh edges), and $\mathcal{E}^{\mathcal{W}}$ are the world edges (i.e., corresponding to the edges connecting two nodes, the one from garment mesh and the other from body). For each garment node, $\mathcal{E}^{\mathcal{W}}$ identifies the closest vertex on the body model. If their distance is less than a specified threshold, we create an edge between them. Then garment features are encoded into graph nodes and edges. Node features include node type, velocity, normal vector and mass. Edge features describe the relative position relationship between two nodes. Then these features are encoded into latent vectors.

To allow nodes exchange information with their neighboring nodes, MeshGraphNets use multiple message passing steps. Each step contains its own distinct set of network parameters and is sequentially applied to the output of the preceding block. Garment edges, world edges and nodes are updated by

$$\mathbf{e}_{ij}^{\mathcal{M}} \leftarrow f^{\mathcal{M}}(\mathbf{e}_{ij}^{\mathcal{M}}, \mathbf{v}_i, \mathbf{v}_j), \tag{1}$$

$$\mathbf{e}_{ij}^{\mathcal{W}} \leftarrow f^{\mathcal{W}}(\mathbf{e}_{ij}^{\mathcal{W}}, \mathbf{v}_i, \mathbf{v}_j), \tag{2}$$

$$\mathbf{v}_i \leftarrow f^{\mathcal{V}}\left(\mathbf{v}_i, \sum_j \mathbf{e}_{ij}^{\mathcal{M}}, \sum_j \mathbf{e}_{ij}^{\mathcal{W}}\right), \tag{3}$$

where $f^{\mathcal{M}}$, $f^{\mathcal{W}}$ and $f^{\mathcal{V}}$ are Multilayer Perceptrons (MLPs) [45]. $\mathbf{v}$ is a graph node. $\mathbf{e}^{\mathcal{M}} \in \mathcal{E}^{\mathcal{M}}$ and $\mathbf{e}^{\mathcal{W}} \in \mathcal{E}^{\mathcal{W}}$ are garment edge and world edge, respectively. Edge features are first updated, then nodes gather information from all edges adjacent to them. After L steps of message passing, a decoder uses an MLP to transform the latent node features to be node accelerations, which are used to update node positions and velocities.

4. Our algorithm

We will now illustrate our algorithm in details. In Section 4.1, we introduce a dynamic subgraph sampling method to optimize computational efficiency. Based on this approach, in Section 4.2, we propose an optimized message passing strategy to reduce redundant information exchange. In Section 4.3, we outline our unsupervised training scheme using a physics-based loss function to model garment deformations without relying on any ground-truth data.

4.1. *Dynamic subgraph sampling*

In MeshGraphNets, each node in the graph represents a point in the mesh, and an edge represents a connection between two points. For each node in the graph, MeshGraphNets require message passing across all its neighboring nodes. This may become computational bottleneck as the number of nodes and edges increases, particularly in high-resolution cloth simulations. We propose a subgraph sampling method to solve this problem. By sampling only a subset of neighbors for each garment node, our method significantly reduces the computational overhead compared to MeshGraphNets, making it more scalable for large and complex garment meshes.

Inspired by GraphSAGE [42], we introduce a sampling technique that aggregates features from a randomly sampled sub-set of neighboring nodes. During each step of message passing, we sample a subset of neighbors for each garment node $\mathbf{v}$ in $\mathcal{G}$, randomly selecting a subset of $\mathbf{v}$'s neighbors to form a subgraph, as shown in Fig. 3.

Unlike Edge Dropout [46–48] that randomly discards a portion of edges in the graph to improve generalization and reduce overfitting, our approach primarily aims at reducing redundant computations and improving the efficiency of message passing.

Once the subgraphs are sampled, we aggregate the features of each garment node $\mathbf{v}$ with the features of its sampled neighbors onto the garment edges connecting them. This aggregation process only involves the sampled nodes and their corresponding sampled edges. This can be described as:

$$\mathbf{e}_{\mathcal{N}(ij)}^{\mathcal{M}} \leftarrow f^{\mathcal{M}}(\mathbf{e}_{\mathcal{N}(ij)}, \mathbf{v}_{\mathcal{N}(i)}, \mathbf{v}_{\mathcal{N}(j)}), \tag{4}$$

where $\mathcal{N}(ij)$ is the set of sampled edges, $\mathcal{N}(i)$ and $\mathcal{N}(j)$ are the sampled nodes. After updating the edge features, we update the features of the garment nodes connected by these sampled edges. For a node $\mathbf{v}$, we aggregate the features from all sampled edges connected to $\mathbf{v}$:

$$\mathbf{v}_{\mathcal{N}(i)} \leftarrow f^{\mathcal{V}}(\mathbf{v}_{\mathcal{N}(i)}, \sum_j \mathbf{e}_{\mathcal{N}(ij)}^{\mathcal{M}}), \tag{5}$$

Sampling a subset of neighbors for each node drastically decreases the amount of data processed. Aggregating features from fewer neighbors also means fewer arithmetic operations, which expedites both the forward and backward passes, leading to overall quicker training and inference. Moreover, memory usage is also significantly reduced. During the forward pass, fewer activations need to be stored, lowering the overall memory footprint. Modern deep learning frameworks dynamically manage memory allocation, and with fewer activations, the peak memory usage is reduced. Consequently, our method can efficiently handle large and complex cloth simulations, offering significant improvements over traditional methods by leveraging the benefits of localized computation through node sampling.

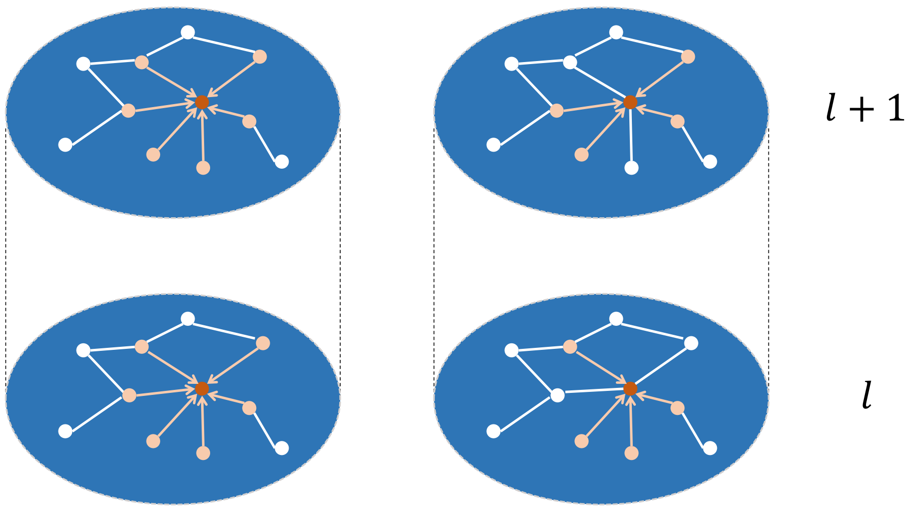


Fig. 3. Comparison with MeshGraphNets [16] in the process of message passing. Left: MeshGraphNets aggregates the features of all its neighboring nodes for each node during every message-passing step. Right: Our algorithm aggregates the features of a randomly sampled subset of neighboring nodes.

4.2. Message reduction

We also observe the following problem related to the message passing frequency between body nodes and garment nodes in existing methods [16,18].

Redundant Message Passing. In our model, each body node connects to a single garment node. Although multiple message passing steps aim to gather information from distant nodes, each body node merely aggregates the initial features of its connected garment node. Consequently, subsequent message passes cause the garment node to repeatedly receive the same initial feature from the body node, resulting in redundant information after the first message passing step.

Less efficient Feature Update for Body Nodes. Each garment node sends its features to the connected body node during each message passing step. Given that each body node is connected to only one garment node, it receives features solely from this garment node. Consequently, the body node's feature is heavily dependent on its connected garment node. Since the body node's feature is merely relayed back to the same garment node, this information loop becomes inefficient.

These observations highlight a critical inefficiency in the message passing process, where redundant information is exchanged between body nodes and garment nodes. To address this, we propose an optimized message passing strategy tailored specifically for garment and body nodes. First, we conduct message passing from body nodes to garment nodes only during the initial step. This ensures garment nodes receive the initial features from body nodes early in the process. After this initial step, we focus solely on message passing among garment nodes. Conversely, we perform message passing from garment nodes to body nodes only during the final step. At this stage, garment nodes aggregate comprehensive information through multiple steps of intra-garment node message passing. This simplifies the message passing process and reduces redundant information exchange, as previously analyzed in Section 4.1. Additionally, our method reduces overall computational costs and memory usage.

4.3. Model training

We use a fully unsupervised training scheme and a physics-based loss function derived from the incremental potential for implicit time stepping [49]. Inspired by SNUG [15] and HOOD, we use optimization-based formulation to define a loss

$$\mathcal{L}_{total} = \mathcal{L}_{strain} + \mathcal{L}_{bending} + \mathcal{L}_{collision} + \mathcal{L}_{gravity} + \mathcal{L}_{inertia} + \mathcal{L}_{friction}. \qquad (6)$$

We use $\mathcal{L}_{strain}$ to model the response of the material to in-plane deformation, which is defined by the Saint Venant Kirchhoff material model [50]. $\mathcal{L}_{bending}$ models the energy caused by the dihedral angle between two adjacent triangles. $\mathcal{L}_{collision}$ is crucial for ensuring valid predictions by pushing outfit vertices outside the body. We penalize the negative distance between garment nodes and their closest points on the body mesh once it falls below a specified threshold. $\mathcal{L}_{gravity}$ models the effect of gravity of garment deformations. $\mathcal{L}_{inertia}$ models the inertia of the garment. $\mathcal{L}_{friction}$ penalizes the sliding friction between the garment and the body.

Here, the key is to convert an optimization problem of solving implicit Euler equations in a physics-based method into a loss function. Since an optimization problem involves finding a minimum value and our training process continuously minimizes the loss value, this transformation is straightforward. Using this loss function during training, our model can learn garment deformations without any ground-truth data.

5. Experiments and evaluation

In this section, we provide some details of our implementation, including training, experiments, comparison and evaluation.

5.1. Training

We conducted our training on a machine equipped with an Intel Core i5-13600KF CPU and NVIDIA GeFroce RTX 3090 GPUs. For consistency, we use the same set of 52 sequences with 6519 frames from the AMASS dataset [51] as used in [15,18]. Additionally, we set aside four full sequences for validation. To evaluate the loss function from any starting point, we provide the garment mesh state for the two previous time steps. The geometries are approximated using linear blend skinning combined with the diffused body model. This can avoid penetration between the skinned garment meshes and the body. To reduce the influence of inertia, we apply a coefficient within the range (0,1].

To improve the diversity in body shapes, we randomly sample the shape parameter from a uniform distribution $\mathcal{U}(-3,3)$. Similar to the approach in HOOD, we utilize a variety of garment meshes including shirts, tank tops, long-sleeve tops, shorts, pants, and dresses. All of our experiments use meshes with irregular triangular topology, where each node typically has 6 or 7 neighboring nodes. This strategy enables learning from thousands of different body shapes and a wide range of garments. During training, we sample the material parameters within

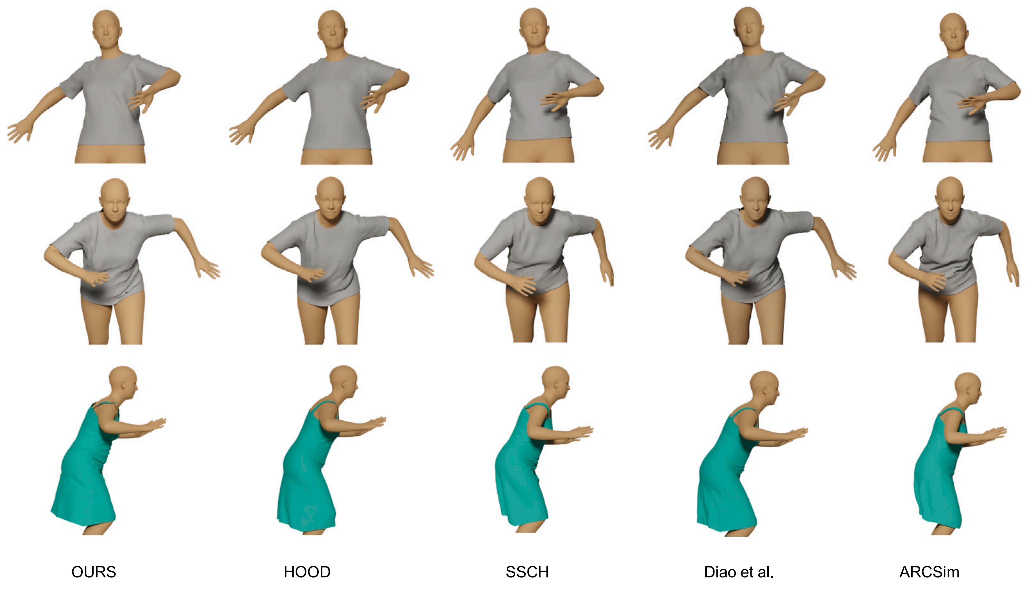


Fig. 4. Comparison with the state-of-art methods, including HOOD [18], SSCH [52], Diao et al. [37] and physics-based simulation from ARCSim [3]. Our method effectively generates realistic cloth deformation for unseen poses, demonstrating its robustness and ability to accelerate real-time cloth simulation using optimizing message passing in GNNs.

Table 1
Ablation on training speed and memory usage. Our method significantly reduced both training time and memory usage, thereby accelerating the model's convergence. Consequently, it allows for processing more data per training iteration under limited computational resources.

	Training time (h)	Cost/iteration (s)	Memory
Baseline	52	1.71	22.8 GB
Only sampling	43	1.33	19.2 GB
Only reduction	40	1.29	15.1 GB
Ours	30	0.98	9.8 GB

an appropriate range and apply these parameters to compute the loss terms. This allows the training to learn the dynamic behavior of fabrics with different material properties. For handling collisions, we set a threshold $\epsilon = 4$ mm. Particle masses are computed according to their individual vertex area and fabric density.

We set the initial learning rate to 0.0001. To ensure stable and gradual convergence, we halve the learning rate every 30,000 training steps. This annealing process helps in fine-tuning the model parameters over time, preventing abrupt changes that could destabilize the learning process. We employ an auto-regressive training method to improve the model's predictive capability over multiple time steps. This method starts by predicting the accelerations for the next single time step. Gradually, the number of predicted steps increases every 3000 iterations, allowing the model to learn dependencies over longer sequences. The number of predicted steps increases up to a maximum of 5. This incremental approach helps better capture the temporal dynamics of cloth deformation.

In each message passing step, we randomly sample a subset of each node's neighboring edges. Given that our irregular triangle meshes average 6 to 7 neighbors per node, we uniformly sample 4 neighbors per node in each iteration. Our network architecture comprises 21 message passing steps and standard MLPs within the encoder, processor, and decoder stages, with each MLP featuring 2 hidden layers and ReLU activations.

To evaluate the efficacy of our approach, we also trained a baseline model that did not incorporate our optimized message-passing technique. Our approach focuses on optimizing message passing, which means it can be applied to any scenario that utilizes MeshGraphNets. While existing methods, such as HOOD [18], have also introduced optimizations to MeshGraphNets' message passing, we use the original message passing scheme from MeshGraphNets as our baseline for a more intuitive comparison. It is important to note that our method can be seamlessly integrated with HOOD, resulting in a significant efficiency boost. This baseline serves as a reference to demonstrate the performance improvements brought by our method. The training comparison is given in Table 1.

5.2. *Comparison*

Fig. 5 presents a quality comparison between our method and the baseline over a sequence of continuous motions. Both models were trained for the same number of iterations and the same message-passing steps. Fig. 5 demonstrates that our method achieves results identical to the baseline.

We further compare our method with HOOD [18], SSCH [52], Diao et al. [37], shown in Fig. 4. As a reference, we also include the physics-based simulation results from ARCSim [3]. The poses used in Fig. 4 were unseen during training. Using our optimized message passing, the trained model achieves excellent cloth deformation results. Both our model training and the baseline used the same number of iterations and the same message-passing steps. The simulation results achieved using our algorithm are almost identical to the baseline. Our optimized approach not only enhances efficiency but also maintains the high-quality cloth deformations, making it a robust solution for real-time applications. Since the ground truth in our work is derived from physically-based methods, and our approach relies on learning-based techniques, a direct numerical comparison between the two is not entirely feasible. Physically-based methods follow physical equations to simulate cloth dynamics, which makes them deterministic in their results, while learning-based methods, such as ours, aim to infer these dynamics through pattern recognition and approximation. It is important to note that nearly all learning-based methods in cloth simulation rely heavily on visual comparisons to evaluate the accuracy and realism of the generated results against the physically-based ground truth.

Fig. 1 demonstrates additional results produced by our method for unseen poses, different body shapes and a variety of garments including t-shirt, top, dresses, pants and shorts. Leveraging the generalization capability of GNNs, our method can generate complex deformations merely using a single model. In contrast, most existing learning-based methods require training a specific model for each type of garment. Please watch the accompanying video for the simulation produced using our algorithm.

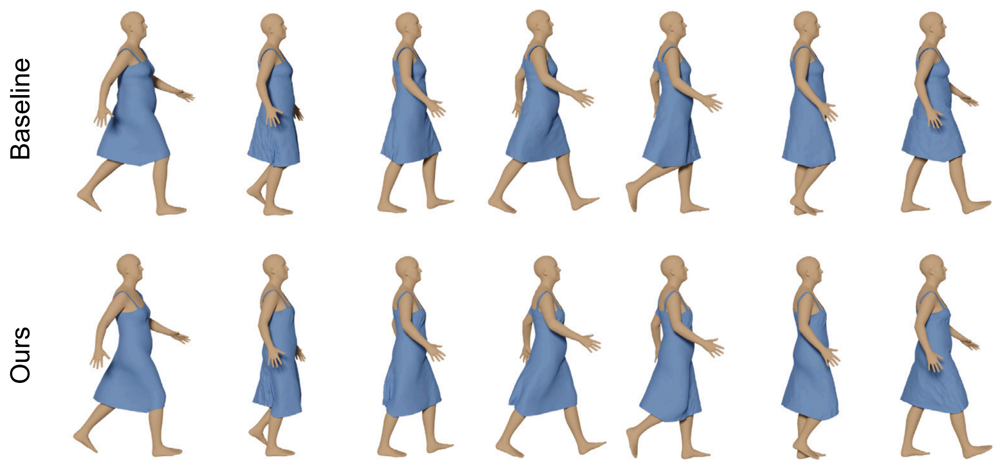


Fig. 5. Comparison with the baseline.

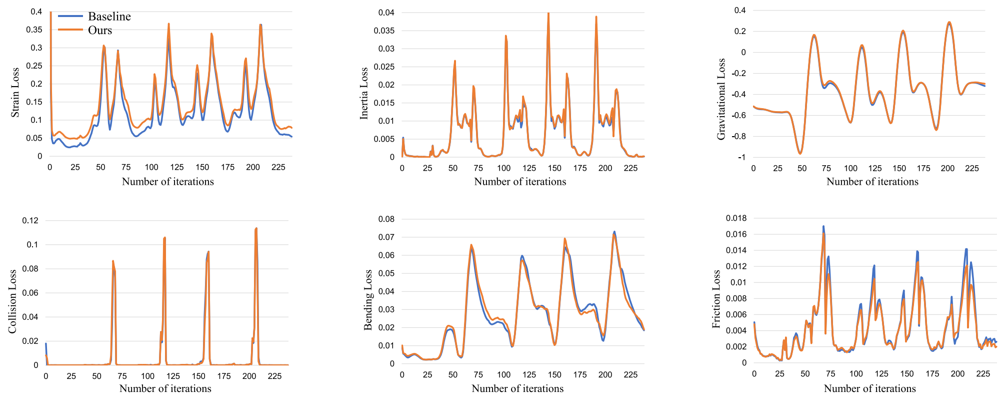


Fig. 6. Quantitative evaluation in terms of loss metrics. Both our method and the baseline were trained using the same number of iterations and the same batch size. Our method exhibits a slightly higher strain loss compared to the baseline. However, in other loss metrics, our method shows almost no difference from the baseline, highlighting its efficiency without compromising accuracy.

Though both MeshGraphNets and our approach aim at optimizing message passing in Graph Neural Networks, they exhibit essential distinction in learning paradigm. MeshGraphNets employs a supervised learning approach, where the model is trained on large datasets with predefined labels or ground truth, making the learning process more straightforward in terms of optimizing for a known target. In contrast, we adopt an unsupervised learning framework and do not rely on labeled data or predefined ground truth. Instead, it leverages a self-supervised loss function derived from physical principles, enabling the model to learn the underlying dynamics of cloth deformation without direct supervision. Therefore, this approach is inherently more challenging as it demands careful formulation of the loss functions to guide the model towards accurate predictions without explicit feedback during training.

5.3. Quantitative evaluation

In Fig. 6, we compare the loss terms for the two methods for a test sequence from the AMASS dataset. Our method shows a slightly higher strain loss compared to the baseline, primarily due to the use of random sampling for neighboring nodes during each message passing iteration. Other loss metrics remain nearly identical. Considering the performance improvements, such as faster training, faster inference and reduction in memory usage, the slight increase in strain loss is negligible. Thus, our method offers a compelling balance between performance and precision, making it highly suitable for real-time applications where computational efficiency is crucial.

In our experiments, we investigated the effect of varying the sample numbers on the performance of our model. As described in Section 4.1, the sampling number indicates the number of neighboring nodes with which each node exchanges information during each message-passing step. We trained models with different sampling numbers for the same number of iterations to evaluate their impact on the loss metrics. Fig. 7 presents the comparison. The inference loss gradually decreases as the sampling number increases. However, once the sampling number is 4, loss reduction becomes less significant. Considering both the quality and training time, we select 4 as the optimal sampling number for each message-passing iteration.

5.4. Ablation

We performed ablation studies to isolate the impact of each optimization technique. Specifically, we compared the training and inference performance of four models: a baseline model with no optimizations, a model with only Dynamic Subgraph Sampling, a model with only Message Reduction, and a model incorporating both Dynamic Subgraph Sampling and Message Reduction.

Table 1 provides a comparison of training time, iteration speed, and memory usage for these four models. The results highlight the effectiveness of each optimization technique in accelerating training and reducing memory consumption. Notably, the combined model (i.e., a model incorporating both Dynamic Subgraph Sampling and Message Reduction) outperforms all other methods in terms of training speed and memory efficiency.

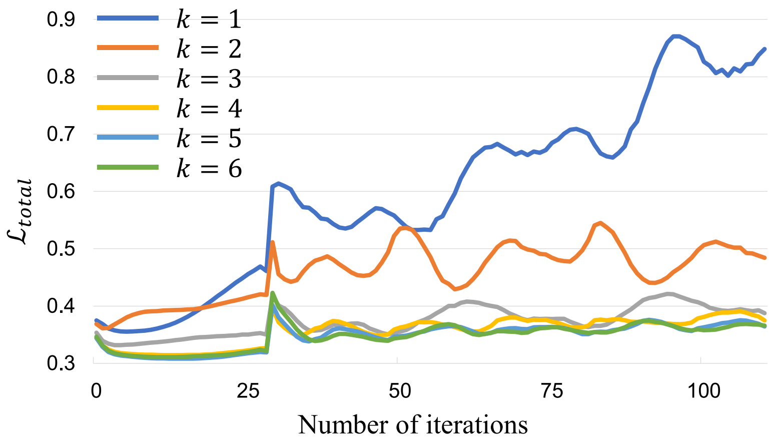


Fig. 7. Comparison under different number of sampling. With the increasing number of sampling, the total loss $\mathcal{L}_{total}$ decreases. The decrease becomes less obvious when the number of sampling is great than 4.

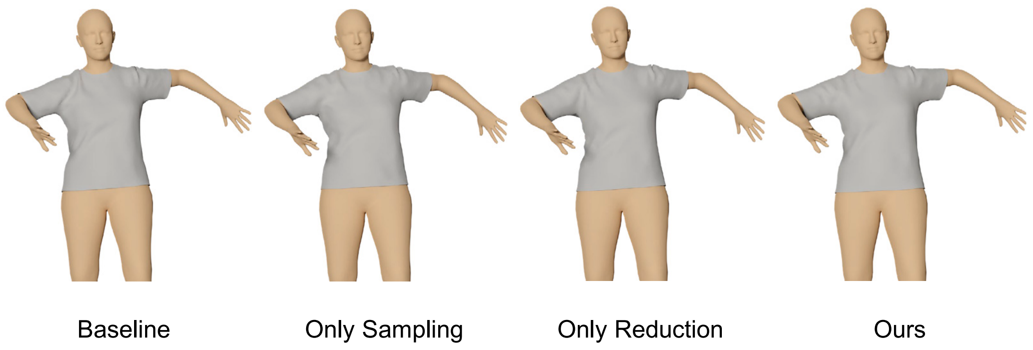


Fig. 8. Ablation study on quality.

Table 2
Ablation on inference for different garments. Our method significantly improves inference speed while maintaining almost the same loss as the baseline.

Garments	Method	Speed (FPS)	Memory	$\mathcal{L}_{total}$
Tshirt	Baseline	21.89	716 MB	0.448
	Only sampling	28.63	491 MB	0.458
	Only reduction	32.22	557 MB	0.449
	Ours	39.18	373 MB	0.461
Pants	Baseline	24.07	692 MB	0.396
	Only sampling	31.01	480 MB	0.405
	Only reduction	34.74	552 MB	0.398
	Ours	41.32	362 MB	0.409
Dress	Baseline	10.69	1098 MB	0.603
	Only sampling	13.85	714 MB	0.621
	Only reduction	15.78	793 MB	0.605
	Ours	20.06	532 MB	0.625

Table 2 presents a comparison of inference performance across different garments (T-shirt, dress, and pants) based on inference speed, memory usage, and loss value. The evaluation was conducted on four test sequences from the AMASS dataset, comprising 598 unseen frames during training. These garments consist of 4000 vertices for the T-shirt, 3000 vertices for the pants, and 12,000 vertices for the dress, respectively. Our method demonstrated a significant improvement in inference speed while maintaining a total loss that is nearly equivalent to the baseline. Given optimizing inference speed, we can achieve real-time simulations, such as virtual try-on and interactive garment design. Fig. 8 visually demonstrates the individual contributions of the *only sampling* and *only reduction* components to the overall quality of the cloth simulation.

Table 3
Comparison of performance metrics between HOOD and the integrated method (HOOD + Ours).

	Training time (h)	Speed (FPS)	Memory	$\mathcal{L}_{total}$
HOOD	45	18.55	1121 MB	1.19
HOOD + Ours	28	32.37	641 MB	1.25

HOOD employs hierarchical message passing to accelerate the propagation of messages within the network. Table 3 compares the performance metrics of HOOD with those of the integrated method (HOOD + Ours). By integrating our approach with HOOD, we demonstrate that it seamlessly enhances the hierarchical message passing mechanism, leading to significant performance improvements. Specifically, the combined method achieves faster training, higher processing speed, and reduced memory consumption, while maintaining a similar level of accuracy. These results demonstrate the effectiveness of our approach in enhancing HOOD's message passing efficiency, making it more scalable and suitable for real-time cloth simulation.

5.5. *Interactive applications in VR*

We deployed our network in Unity3D and tested our algorithm using virtual try-on. To predict the dynamic deformation of clothing for posed characters, we retrieve joint transformation and load the pre-trained network model. The vertex positions of the cloth mesh are updated accordingly within Unity3D. In VR headset devices, we can render the model's inference results in an interactive VR environment, allowing users to observe the real-time cloth deformation (see Fig. 9).

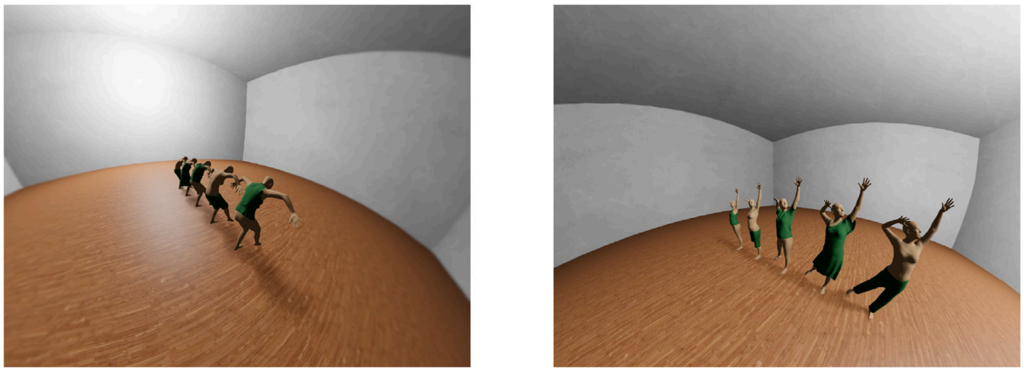


Fig. 9. Fashion and Virtual Try-on in VR.

6. Conclusion

We introduced a novel method for optimizing message passing in GNNs to accelerate real-time cloth simulation. Our efficient message aggregation algorithm significantly reduces computational overhead, resulting in substantial improvements. Our experimental results show that, without compromising the quality of cloth deformation, our algorithm demonstrates a significant reduction in memory usage during training and shows computational improvements in both training and inference. The performance improvement is particularly beneficial for scenarios with hardware limitations, enabling more extensive and complex simulations without requiring expensive computational resources.

Limitations and Future Work. Our method has a few limitations. First, inference may be less efficient when handling cloth with tens of thousands of nodes. Addressing this challenge (i.e., processing higher-resolution cloth) is a valuable research direction. Second, our method cannot handle cloth self-collisions, which sometimes may result in self-penetration. Addressing garment self-collision in a self-supervised manner is a promising direction. Third, when a human body moves at high speed, body-cloth penetrations may occur, necessitating continuous collision detection to address this issue. Our experiments utilize the message passing scheme of MeshGraphNets as a baseline due to its simplicity and clarity. While some recent methods leverage hierarchical message passing, we anticipate that our method can be readily adapted to such schemes. Future work will explore this integration. Furthermore, handling multi-layer garments will significantly enhance realism in complex clothing scenarios. This necessitates modeling inter-layer collisions and interactions, a challenging yet crucial advancement for comprehensive cloth simulation. One potential approach involves connecting mesh nodes of different garment layers via auxiliary edges, analogous to the body-garment node linking in our current model.

CRediT authorship contribution statement

Yang Zhang: Writing – original draft, Software, Resources. **Kailuo Yu:** Writing – original draft, Software, Resources. **Xinyu Zhang:** Writing – review & editing, Project administration, Methodology, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Science and Technology Major Project (2021ZD0114501).

Data availability

Data will be made available on request.

References

[1] D. Baraff, A. Witkin, Large steps in cloth simulation, in: ACM SIGGRAPH, 1998, pp. 43–54.
[2] K.-J. Choi, H.-S. Ko, Stable but responsive cloth, ACM Trans. Graph. 21 (3) (2002) 604–611.
[3] R. Narain, A. Samii, J.F. O'Brien, Adaptive anisotropic remeshing for cloth simulation, ACM Trans. Graph. 31 (6) (2012) 152:1–152:10.
[4] S. Bouaziz, S. Martin, T. Liu, L. Kavan, M. Pauly, Projective dynamics: fusing constraint projections for fast simulation, ACM Trans. Graph. 33 (4) (2014) 154:1–154:11.
[5] T. Liu, A.W. Bargteil, J.F. OBrien, L. Kavan, Fast simulation of mass-spring systems, ACM Siggraph 32 (6) (2013) 214:1–214:7.
[6] J. Yu, Z. Wang, Super-resolution cloth animation with spatial and temporal coherence, ACM Trans. Graph. 43 (4) (2024) 105:1–105:14.
[7] Q. Ma, J. Yang, A. Ranjan, S. Pujades, G. Pons-Moll, S. Tang, M.J. Black, Learning to dress 3d people in generative clothing, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 6469–6478.
[8] E. Gundogdu, V. Constantin, A. Seifoddini, M. Dang, M. Salzmann, P. Fua, GarNet: A two-stream network for fast and accurate 3D cloth draping, in: IEEE/CVF International Conference on Computer Vision, 2019, pp. 8739–8748.
[9] I. Santesteban, M.A. Otaduy, D. Casas, Learning-based animation of clothing for virtual try-on, Comput. Graph. Forum 38 (2) (2019) 355–366.
[10] E. Gundogdu, V. Constantin, S. Parashar, A. Seifoddini, M. Dang, M. Salzmann, P. Fua, GarNet++: Improving fast and accurate static 3D cloth draping by curvature loss, IEEE Trans. Pattern Anal. Mach. Intell. 44 (1) (2020) 181–195.
[11] H. Bertiche, M. Madadi, E. Tylson, S. Escalera, DeePSD: Automatic deep skinning and pose space deformation for 3D garment animation, in: IEEE/CVF International Conference on Computer Vision, 2021, pp. 5471–5480.
[12] R. Li, B. Guillard, E. Remelli, P. Fua, DIG: Draping implicit garment over the human body, in: Asian Conference on Computer Vision, 2022, pp. 2780–2795.
[13] C. Patel, Z. Liao, G. Pons-Moll, TailorNet: Predicting clothing in 3D as a function of human pose, shape and garment style, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 7365–7375.
[14] X. Pan, J. Mai, X. Jiang, D. Tang, J. Li, T. Shao, K. Zhou, X. Jin, D. Manocha, Predicting loose-fitting garment deformations using bone-driven motion networks, in: ACM Siggraph, 2022, pp. 11:1–11:10.
[15] I. Santesteban, M.A. Otaduy, D. Casas, SNUG: Self-supervised neural dynamic garments, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 8140–8150.
[16] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, P.W. Battaglia, Learning mesh-based simulation with graph networks, in: International Conference on Learning Representations, 2021, pp. 1–18.
[17] L. Tiwari, B. Bhowmick, S. Sinha, GenSim: Unsupervised generic garment simulator, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 4168–4177.
[18] A. Grigorev, M.J. Black, O. Hilliges, HOOD: Hierarchical graphs for generalized modelling of clothing dynamics, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 16965–16974.
[19] H. Bertiche, M. Madadi, S. Escalera, Cloth3D: Clothed 3D humans, in: European Conference on Computer Vision, 2020, pp. 344–359.
[20] H. Wang, J.F. O'Brien, R. Ramamoorthi, Data-driven elastic models for cloth: modeling and measurement, ACM Trans. Graph. 30 (4) (2011) 71:1–71:12.
[21] M. Li, D.M. Kaufman, C. Jiang, Codimensional incremental potential contact, ACM Trans. Graph. 40 (4) (2021) 170:1–170:24.

[22] G. Cirio, J. Lopez-Moreno, D. Miraut, M.A. Otaduy, Yarn-level simulation of woven cloth, ACM Trans. Graph. 33 (6) (2014) 207:1–207:11.

[23] J.M. Kaldor, D.L. James, S. Marschner, Efficient yarn-based cloth with adaptive contact linearization, in: ACM SIGGRAPH, ACM, 2010, pp. 105:1–105:10.

[24] M. Macklin, M. Müller, N. Chentanez, T.-Y. Kim, Unified particle physics for real-time applications, ACM Trans. Graph. 33 (4) (2014) 153:1–153:12.

[25] M. Müller, B. Heidelberger, M. Hennix, J. Ratcliff, Position based dynamics, J. Vis. Commun. Image Represent. 18 (2) (2007) 109–118.

[26] M. Müller, Hierarchical position based dynamics, in: Workshop on Virtual Reality Interaction and Physical Simulation, VRIPHYS, 2008, pp. 1–12.

[27] H. Wang, A chebyshev semi-iterative approach for accelerating projective and position-based dynamics, ACM Trans. Graph. 34 (6) (2015) 246:1–246:9.

[28] Z. Wang, L. Wu, M. Fratarcangeli, M. Tang, H. Wang, Parallel multigrid for nonlinear cloth simulation, Comput. Graph. Forum 37 (7) (2018) 131–141.

[29] L. Wu, B. Wu, Y. Yang, H. Wang, A safe and fast repulsion method for GPU-based cloth self collisions, ACM Trans. Graph. 40 (1) (2020) 5:1–5:18.

[30] Y.D. Li, M. Tang, X.R. Chen, Y. Yang, R.F. Tong, B.L. An, S.C. Yang, Y. Li, Q.L. Kou, D-Cloth: Skinning-based cloth dynamic prediction with a three-stage network, Comput. Graph. Forum 42 (7) (2023) e14937.

[31] Y. Li, M. Tang, Y. Yang, R. Tong, S. Yang, Y. Li, B. An, Q. Kou, CTSN: Predicting cloth deformation for skeleton-based characters with a two-stream skinning network, Comput. Vis. Media 10 (3) (2024) 471–485.

[32] T.Y. Wang, T. Shao, K. Fu, N.J. Mitra, Learning an intrinsic garment space for interactive authoring of garment animation, ACM Trans. Graph. 38 (6) (2019) 220:1–220:12.

[33] M. Zhang, T.Y. Wang, D. Ceylan, N.J. Mitra, Dynamic neural garments, ACM Trans. Graph. 40 (6) (2021) 235:1–235:15.

[34] M. Habermann, L. Liu, W. Xu, M. Zollhoefer, G. Pons-Moll, C. Theobalt, Real-time deep dynamic characters, ACM Trans. Graph. 40 (4) (2021) 94:1–94:16.

[35] Y. Jin, D. Omens, Z. Geng, J. Teran, A. Kumar, K. Tashiro, R. Fedkiw, A neural-network-based approach for loose-fitting clothing, 2024, arXiv:2404.16896. URL https://arxiv.org/abs/2404.16896.

[36] M. Zhang, D. Ceylan, N.J. Mitra, Motion guided deep dynamic 3d garments, ACM Trans. Graph. 41 (6) (2022) 219:1–219:12.

[37] J. Diao, J. Xiao, Y. He, H. Jiang, Combating spurious correlations in loose-fitting garment animation through joint-specific feature learning, Comput. Graph. Forum 42 (7) (2023) e14939.

[38] M. Zhang, J. Li, Neural garment dynamic super-resolution, in: ACM Siggraph Asia, 2024, pp. 122:1–122:11.

[39] H. Bertiche, M. Madadi, S. Escalera, PBNS: Physically based neural simulation for unsupervised garment pose space deformation, ACM Trans. Graph. 40 (6) (2021) 198:1–198:14.

[40] H. Bertiche, M. Madadi, S. Escalera, Neural cloth simulation, ACM Trans. Graph. 41 (6) (2022) 220:1–220:14.

[41] Y. Cao, M. Chai, M. Li, C. Jiang, Efficient learning of mesh-based physical simulation with bi-stride multi-scale graph neural network, in: Proceedings of the 40th International Conference on Machine Learning, Vol. 202, 2023, pp. 3541–3558.

[42] W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, Adv. Neural Inf. Process. Syst. 30 (2017) 1025–1035.

[43] R. Ying, R. He, K. Chen, P. Eksombatchai, W.L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 974–983.

[44] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, C.-J. Hsieh, Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks, in: ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 257–266.

[45] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536.

[46] Y. Rong, W. Huang, T. Xu, J. Huang, Dropedge: Towards deep graph convolutional networks on node classification, 2019, arXiv preprint arXiv:1907.10903.

[47] P.A. Papp, K. Martinkus, L. Faber, R. Wattenhofer, DropGNN: Random dropouts increase the expressiveness of graph neural networks, Adv. Neural Inf. Process. Syst. 34 (2021) 21997–22009.

[48] T. Fang, Z. Xiao, C. Wang, J. Xu, X. Yang, Y. Yang, Dropmessage: Unifying random dropping for graph neural networks, AAAI Conf. Artif. Intell. 37 (4) (2023) 4267–4275.

[49] S. Martin, B. Thomaszewski, E. Grinspun, M. Gross, Example-based elastic materials, in: ACM Siggraph, 2011, pp. 72:1—-72:8.

[50] J. Montes, B. Thomaszewski, S. Mudur, T. Popa, Computational design of skintight clothing, ACM Trans. Graph. 39 (4) (2020) 105:1–105:12.

[51] N. Mahmood, N. Ghorbani, N.F. Troje, G. Pons-Moll, M.J. Black, AMASS: Archive of motion capture as surface shapes, in: IEEE/CVF International Conference on Computer Vision, 2019, pp. 5442–5451.

[52] I. Santesteban, N. Thuerey, M.A. Otaduy, D. Casas, Self-supervised collision handling via generative 3D garment models for virtual try-on, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 11763–11773.