DOI: 10.1111/cgf.14493
EUROGRAPHICS 2022 / R. Chaine and M. H. Kim
(Guest Editors)
Volume 41 (2022), Number 2

N-Cloth: Predicting 3D Cloth Deformation with Mesh-Based Networks

Y. D. Li[1] M. Tang[1] Y. Yang[1] Z. Huang[1] R. F. Tong[1] S. C. Yang[3] Y. Li[3] and D. Manocha[2]

[1]Zhejiang University, China
[2] University of Maryland at College Park, America
[3] Tencent

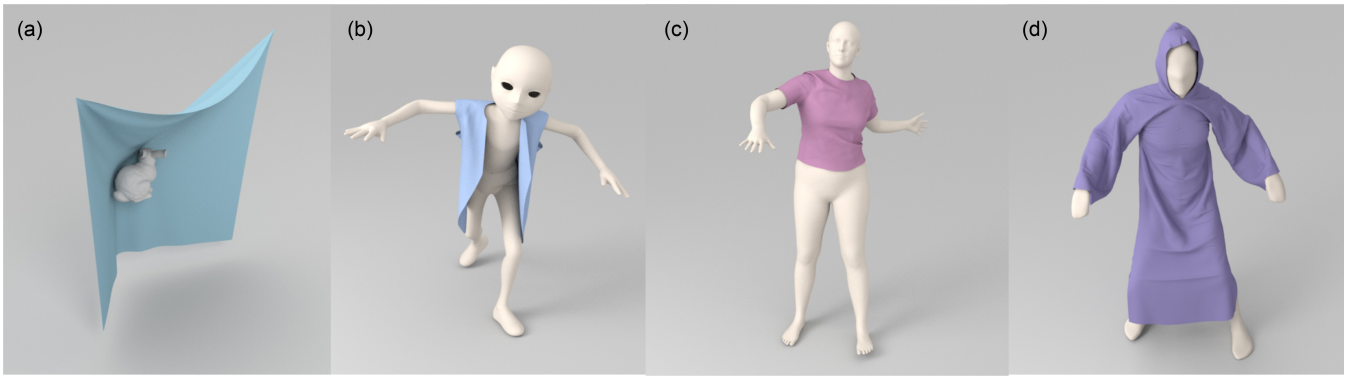

Figure 1: *Given the initial template of the cloth mesh and the target obstacle mesh, our network can predicate a plausible target 3D cloth mesh for general scenes. We highlight (a) the final cloth mesh wrapped around a bunny; (b) a jacket draped on a non-SMPL human body; (c) t-shirt deformation on a SMPL human body; (d) a human dressed in a robe represented by* $100K$ *triangles. All predicted meshes are different from the datasets used for training. Our approach runs at* $30-45$*fps on an NVIDIA GeForce RTX 3090 GPU.*

Abstract

We present a novel mesh-based learning approach (N-Cloth) for plausible 3D cloth deformation prediction. Our approach is general and can handle cloth or obstacles represented by triangle meshes with arbitrary topologies. We use graph convolution to transform the cloth and object meshes into a latent space to reduce the non-linearity in the mesh space. Our network can predict the target 3D cloth mesh deformation based on the initial state of the cloth mesh template and the target obstacle mesh. Our approach can handle complex cloth meshes with up to $100K$ *triangles and scenes with various objects corresponding to SMPL humans, non-SMPL humans or rigid bodies. In practice, our approach can be used to generate plausible cloth simulation at* $30-45$ *fps on an NVIDIA GeForce RTX 3090 GPU. We highlight its benefits over prior learning-based methods and physically-based cloth simulators.*

CCS Concepts

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

1. Introduction

Generating plausible cloth simulation has been an active research area for many decades. The driving applications include video games and VR, computer animation, special effects, the fashion industry, virtual try-on applications, etc. There is extensive literature on simulating cloth deformation using physically-based and data-driven methods.

Physically-based methods treat cloth simulation as a deformable modeling problem and solve is using techniques from scientific computing and geometric computing. These methods also perform collision handling for accurate simulation. The resulting algorithms can generate high-fidelity simulations and can be accelerated by exploiting GPU parallelism. However, they are mostly limited to offline simulations and are not considered fast or practical for in-

teractive applications such as games and VR. There has been considerable interest in developing data-driven or learning-based approaches for interactive simulation. The data-driven methods use a large number of pre-computed simulated clothing samples to synthesize cloth deformation. Recently, many learning-based methods have been proposed for draping cloth or adjusting cloth deformation to human motion [PLP20; SOC19; LMR*15; BME20a; BME20b; WCC*21; STOC21]. While these methods can predict clothing deformation in 3D space at interactive rates, they may not work well for arbitrary scenarios with different types of objects exerting force on the cloth. In practice, these learning methods are mainly limited to predicting the deformation of clothes that conform to human movements. Moreover, many of these algorithms are limited to SMPL-based human body models [LMR*15] or virtual try-on applications. It is not clear whether these learning methods can extend to other types of irregular fabrics such as a table cloth wrapping around an arbitrary obstacle like a bunny. Often these methods also require some pre-processing such as skinning [GCS*19; GCP*20], which can introduce artifacts into subsequent network training.

Main Results: We present a novel learning-based method (N-Cloth) to interactively predict cloth deformations in 3D. Our approach is designed for general scenes represented using triangle meshes and makes no assumption about the topology of the cloth or the shape/topology of the obstacle. Moreover, the simulation environment may consist of arbitrary rigid or deforming objects (e.g., humans in motion) that apply forces on the cloth and can result in complex deformations. Our learning method predicts the target3D cloth mesh deformation based on the initial state of the cloth mesh template and the target obstacle mesh.

A key aspect of our learning-based approach is the use of a network that directly uses the input meshes and does not require pre-processing (e.g., mesh skinning). We extend the classic encoder-decoder architecture [RHBL07] with two major components: a graph-convolution-based encoder network and a fusion network. The first network transforms the input cloth and object meshes into latent vectors of a latent space and greatly reduces the input data size. This enables our algorithm to handle complex objects in the scenes defined using triangle meshes (e.g., with up to $100K$ triangles). Our fusion network is used to derive the deforming mesh from the input cloth meshes and obstacle meshes in the latent space. This increases the accuracy of our overall learning-based method in terms of predicting arbitrary 3D cloth deformations. The connections between the outputs of an obstacle encoder and a cloth decoder are introduced to generate more accurate cloth deformations. Moreover, our learning method can also generate detailed features like wrinkles.

We qualitatively and quantitatively analyze the performance of the proposed mesh-based network in a variety of scenarios. These include cloth meshes corresponding to many types and topologies. Furthermore, the obstacles in the scene correspond to rigid objects or a human body. In practice, our approach can generate plausible cloth deformations for all these scenarios, even when the predicted meshes are different from the datasets used in training. We also compare the performance with TailorNet [PLP20], a SMPL-based network, and observe lower error with respect to the ground truth.

The novel aspects of our learning-based approach include:

- **A novel mesh-based network for various scenes:** Our approach can handle arbitrary obstacle meshes. This is in contrast to recent learning-based methods that are mainly limited to parametric human models [PLP20; SOC19; LMR*15; BME20a; BME20b; WCC*21; STOC21].
- **End-to-end neural network:** Our method can predict cloth deformation given the initial cloth template and the target obstacle mesh. We do not perform any pre-processing (e.g., skinning computations [GCS*19; GCP*20]).
- **Interactive speed:** Our approach can predict cloth deformation at $30-45$ fps on an NVIDIA GeForce RTX 3090 GPU. We observe $5-8$X and $220-300$X speedups over prior GPU-based [TWL*18] and CPU-based physics-based simulators [NSO12; NPO13], respectively.
- **Plausible Results:** We have evaluated the accuracy of our approach on a large number of complex cloth deformations and observe plausible results. Compared with TailorNet, our method can predict the cloth mesh with more wrinkles.
- **Lower Memory Overhead:** Our approach can handle cloth meshes with up to $100K$ triangles on commodity GPUs.

2. Related Work

In this section, we give a brief overview of prior work on cloth deformation using physically-based simulation and learning-based methods.

2.1. Physically-based Cloth Simulation

Physically-based algorithms use explicit Euler integration [Pro95], implicit Euler integration [BW98], iteration optimization [LBOK13; LBK17], or projective dynamics [BML*14] to calculate the cloth deformation under external/internal forces. Many techniques have been proposed for robust collision handling [BFA02; BEB12; TTWM14; GLM05]. Impulse-based methods and impact zones [BFA02; HVTG08; Pro97; TWL*18] are used for penetration handling. Recently many techniques have been proposed to accelerate these simulations using one or more GPUs [TWT*16; LTT*20]. In practice, accurate cloth simulators can generate high–fidelity simulations, and we regard them as the ground truth for our learning approach.

2.2. Data-driven Approaches

Many data-driven approaches have been proposed for cloth deformation synthesis [FYK10; WHRO10]. By combining high-quality wrinkles with a coarse cloth simulation [ZWCM21; CMM*20], visually plausible results can be generated at interactive rates. These methods commonly need to simulate coarse deformed meshes. Furthermore, these methods need to precompute a large dataset, and their generalizability to arbitrary scenarios tends to be limited [dASTH10; KKN*13; ZBO13].

2.3. Learning-based Algorithms

Recently, learning-based algorithms have been proposed for predicting cloth deformation in 3D. Using the synthetic training

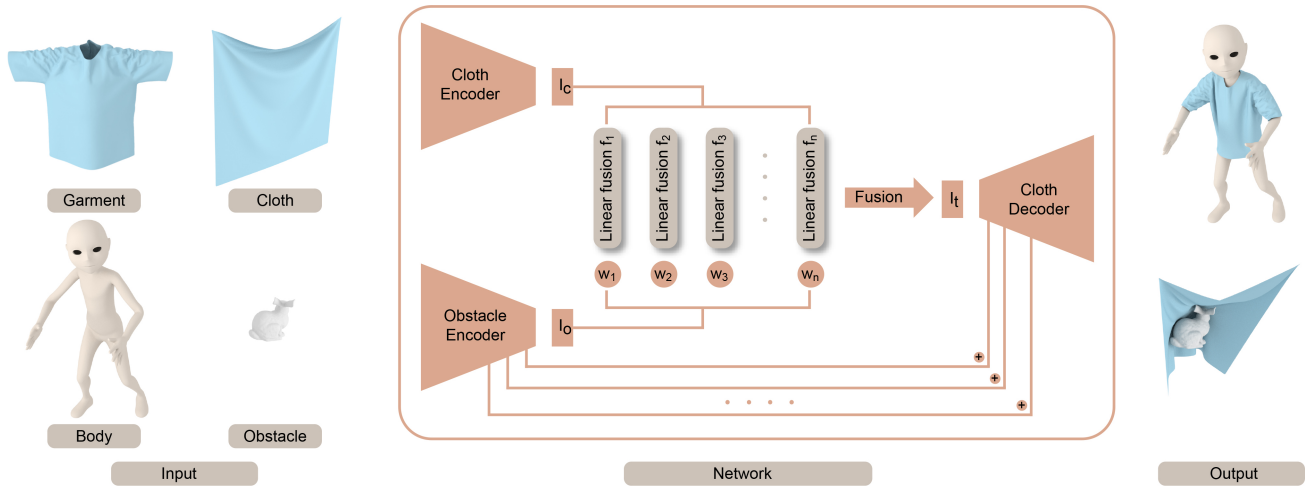

Figure 2: *Our mesh-based network architecture:The initial cloth mesh template is encoded into a vector* $\mathbf{l}_c$ *in the latent space by a cloth encoder. The obstacle mesh in the target state is encoded as vector* $\mathbf{l}_o$*, which is used as a weighting factor in the latent space by the obstacle encoder. With a fusion network, cloth vector* $\mathbf{l}_c$ *is weighted and fused by n functions with different weights to compute the latent vector* $\mathbf{l}_t$*.* $f_i, f_2, ..., f_n$ *are linear fusion functions with trainable parameters. Finally, the latent vector is restored to the deformed cloth mesh in the target state by the cloth decoder. The output of each layer in the obstacle encoder is connected to the cloth decoder by a linear function to add more obstacle impact.*

data generated using physics-based simulators, learning-based approaches can predict cloth deformation at interactive rates on commodity GPUs. A large number of learning-based algorithms [PLP20; SOC19; LMR*15; BME20a; BME20b; WCC*21; STOC21; WSFM19; CPA*21] have been designed for specific or parametric obstacle models such as SMPL-based [LMR*15] or skeleton-based human body models. This makes it difficult to use these methods in environments with arbitrary rigid or deformable objects that can interact with the cloth.

Some learning-based algorithms are not limited to SMPL models. These approaches aim to process human bodies with skeleton. Holden et al. [HDDN19] obtain the vector of the vertex attributes of in the subspace through PCA and divide the deformation into linear and nonlinear to make assumptions and to predict the subsequent deformation. The GarNet network architecture proposed by Gundogdu et al. [GCS*19; GCP*20] predicts the cloth deformation from the target posture with DQS (dual quaternion skinning) preprocessing [KCZO07] from the initial state and uses it as the final cloth deformation. [ZCWM21] learns to generate rendered characters and cloth on posed skeleton joints. All these methods focus on human bodies with skeleton and can not handle scenes without human skeletons. Our aim is to find an approach capable of processing many scenes with human meshes and other rigid body meshes.

3. Our Approach

3.1. Overview

Our goal is to predict the target deformed cloth mesh based on the target obstacle mesh and the initial cloth mesh. We assume that they are represented as 3D meshes. The initial cloth mesh provides a template for deformation. The target obstacle mesh is used to guide deformation. We do not make any assumptions about the initial topology of the cloth, though it remains fixed during the simulation or deformation. Thus, our network is an end-to-end method for predicting the cloth deformation. Formally, our approach can be described as:

$$M_t^c = \mathcal{N}_\theta(M_i^c, M_t^o), \tag{1}$$

where M_t^c is the predicted deformed target cloth mesh and M_t^o is the target obstacle mesh. M_i^c is the initial, undeformed cloth mesh template which is undeformed and constant for one specific kind of cloth. During the prediction, the topologies of the cloth mesh and the obstacle mesh are invariable. $\mathcal{N}_\theta$ is the mesh-based network and θ represents the network parameters obtained by training.

We extend the classic encoder-decoder neural network architecture [RHBL07] to generate the 3D cloth deformation. An overview of our approach is shown in Fig. 2. The deformation in mesh space is nonlinear and too complicated to be modeled. The nonlinearity of the mesh deformation is transformed to linear fusion in latent space through a cloth encoder and an obstacle encoder. Thus, we obtain the latent representation of the target cloth mesh by linear fusion. The target cloth mesh is obtained by a cloth decoder. We will describe the network in more detail.

3.2. Encoder Network

We use the encoder network to transform the input cloth mesh M_i^c and the obstacle mesh M_t^o from the 3D mesh space into a latent space. Our goal is to handle arbitrary triangle cloth meshes. A triangle mesh is similar to graph data. Therefore, the networks for processing graph data are applicable to resolving relevant mesh problems. Referring to [GJ19], graph convolution is used in this

paper to perform feature extraction on a mesh with abundant triangles. Both the cloth encoder and the obstacle encoder have similar network architectures.The first network block of our encoder network is shown in Fig. 3. Both the cloth encoder and the obstacle encoder have several network blocks similar to this initial block. Then we elaborate on the various parts of the first network block in the encoder. As shown in Fig. 3, we use the GCN layer [KW16] to extract features from the geometry information (vertex coordinates) and topology information (edge connectivity) of the mesh. In this manner, the GCN layer maintains the topology of the mesh. The GCN layer can be formulated as in [KW16]:

$$X^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}^{(l)}\tilde{D}^{-\frac{1}{2}}X^{(l)}W^{(l)}\right), \tag{2}$$

where $\tilde{A}^{(l)} = A^{(l)} + I$, $A^{(l)}$ is the adjacency matrix of layer l. I is the identity matrix for adding self-loops. $X^{(l+1)}$ and $X^{(l)}$ are the feature matrices of the layer $l+1$ and l, respectively. $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, and $W^{(l)}$ is a trainable weight matrix for layer l. Since the output of the

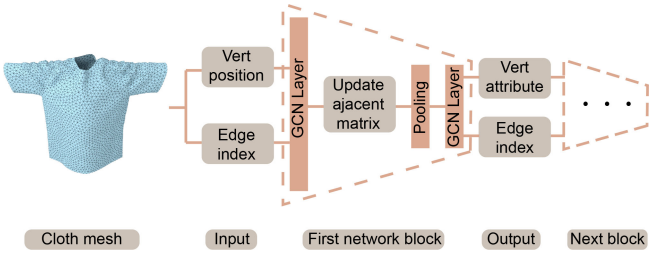

Figure 3: *The first network block of our encoder network: Our network takes geometry information (vertex coordinates) and topology information (edge connectivity) of meshes as inputs and performs data down-sampling by outputting meshes with reduced geometry and topology information.*

GCN layer has the same topology as the input, this formulation can result in a large number of training parameters for high-resolution meshes and thereby exceed the GPU memory budget. To reduce the model parameters and complex non-linearity in 3D space, we use top-k-pooling [GJ19] to perform data down-sampling by outputting meshes with reduced topology information, i.e., with fewer vertices and connectivity information among them.

Top-k-pooling will pick k vertices from the original vertex set and discard other vertices to perform down-sampling. This process may result in multiple isolated point sets, which may not work well for subsequent GCN layers because these layers extract features in terms of information related to the of the vertex. Thus, we recalculate the connectivity of mesh vertices before using a top-k-pooling layer to improve triangle connectivity. Therefore, we calculate the square of the adjacency matrix A as follows [GJ19]:

$$A_u^{(l)} = A^{(l)}A^{(l)}, \tag{3}$$

where $A_u^{(l)}$ is the new adjacency matrix for next top-k-pooling computation. The new adjacency matrix $A_u^{(l)}$ corresponds to introducing more vertices around a given vertex and will reduce the isolated points after pooling.

Fig. 4 shows the connectivity between a single vertex and surrounding vertices in the mesh where the adjacency matrix is $A^{(l)}$ or $A_u^{(l)}$. The connectivity between vertices is strengthened with the adjacency matrix $A_u^{(l)}$.

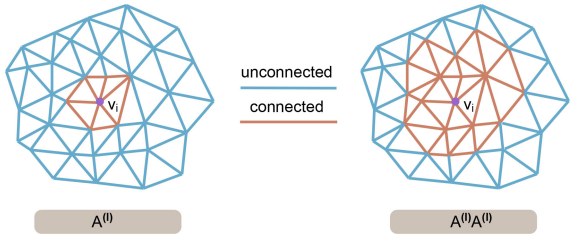

Figure 4: *The connectivity of vertex v_i with adjacency matrices $A^{(l)}$ (a) and $A_u^{(l)}$ (b). The connectivity between vertices is strengthened with the adjacency matrix $A_u^{(l)}$.*

3.3. Fusion Network

The output of the cloth encoder and the obstacle encoder are $\mathbf{I}_c$ and $\mathbf{I}_o$, respectively. $\mathbf{I}_c$ and $\mathbf{I}_o$ are vectors in the latent space extracted from the cloth mesh and the obstacle mesh, respectively. $\mathbf{I}_c$ and $\mathbf{I}_o$ are expressed as follows:

$$\begin{aligned} \mathbf{I}_c &= \mathbf{E}_c(M_i^c) = (x_1, x_2, x_3, \cdots, x_m), \\ \mathbf{I}_o &= \mathbf{E}_o(M_t^o) = (w_1, w_2, w_3, \cdots, w_n), \end{aligned} \tag{4}$$

where $\mathbf{E}_c$ and $\mathbf{E}_o$ represent the cloth encoder network and the obstacle encoder network, respectively. m and n are the dimensions of $\mathbf{I}_c$ and $\mathbf{I}_o$, respectively. $x_1, x_2, ..., x_m$ and $w_1, w_2, ..., w_n$ are the component of latent vectors $\mathbf{I}_c$ and $\mathbf{I}_o$, respectively.

We use the fusion network to generate $\mathbf{I}_t$, which corresponds to the vector of the target cloth mesh in the latent space from $\mathbf{I}_c$ and $\mathbf{I}_o$. Our formulation of the fusion network is inspired by prior work in image processing and 3D character control. Rocco et al. [RAS17] added a correlation layer to the network for geometric matching between 2D images, and Holden et al. [HKS17] proposed a phase-functioned network the weights of which are updated by a phase cyclic function for 3D character control.

We perform the fusion by linearly weighting a set of linear fusion functions $\{f_1, f_2, f_3, \cdots, f_n\}$, which all take $\mathbf{I}_c$ as an input. Here $\mathbf{I}_o$ is used as the linear weight. The linear fusion functions are defined as follows:

$$f_i(\mathbf{I}_c) = \begin{bmatrix} \alpha_i^1 \\ \alpha_i^2 \\ \cdots \\ \alpha_i^m \end{bmatrix} \left[x_1 + x_2 + \cdots + x_m\right], \tag{5}$$

where $\{\alpha_i^1, \alpha_i^2, \cdots, \alpha_i^m\}$ is a set of trainable coefficients and $i \in [1, n]$. The overall fusion process can be expressed by the following formula:

$$\begin{aligned} \mathbf{I}_t &= \sum_{i=1}^{n} w_i f_i(\mathbf{I}_c) \\ &= w_1 f_1(\mathbf{I}_c) + w_2 f_2(\mathbf{I}_c) + w_3 f_3(\mathbf{I}_c) + \cdots + w_n f_n(\mathbf{I}_c). \end{aligned} \tag{6}$$

We use this formulation to obtain the vector $\mathbf{I}_t$ of the target cloth mesh in the latent space. Thus, we obtain a linear latent space where the deformation can be expressed as a linear fusion. In practice, we observe that our linear formulation can predict plausible results, and the errors are rather small (see Section 4).

In the mesh space, the influence of the obstacle mesh on the deformed cloth mesh may be complex and non-linear (e.g., due to collisions between the cloth and the obstacle). With our fusion network, we are able to model the non-linearity as weighted combinations of latent vectors and obtain the parameters of the combination function by training. In addition, the dimensions m and n of $\mathbf{I}_c$ and $\mathbf{I}_o$, respectively, also govern the accuracy of our predicted deformation. In our implementation, we set $m = 96$ and $n = 80$ for most benchmarks. We choose these dimensions by experiments and find that increasing them does not obviously improve the results.

3.4. Decoder Network

We use the decoder network to generate a cloth mesh in the world space from I_t. The problem of recovering the cloth mesh from the latent space has been investigated by [CZY20; CMM*20; CGY*21]. Although these methods use graph convolution networks, the resulting decoder networks use the same sampling information as the encoder networks. In addition, there is almost no deformation between the input mesh and the output mesh. However, this formulation does not work well when the target cloth mesh involves a large deformation relative to the initial mesh. In our formulation, we assume the cloth mesh maintain the same topology during deformation. As a result, we reduce the problem to only computing the geometric information of the deformed mesh (i.e., the vertex coordinates).

To compute the coordinates of the deformed vertices, we use a Multilayer Perceptron (MLP) network to decode the feature vector $\mathbf{I}_t$ of the target deformed cloth. In each layer of cloth decoder, we apply dropout regularization by randomly disabling 20% of the hidden neurons to avoid overfitting the training data. The output of our decoder network is a one-dimensional vector that will be reshaped to $(num, 3)$ as vertex coordinates of the target cloth mesh, where num is the number of vertices of the target cloth mesh. In addition, we add the output of each layer of the obstacle encoder to the input of the corresponding layer of the decoder through a linear layer connection. In this way, we increase the effects of obstacles on the cloth decoder and find improved results.

3.5. Loss Functions

Loss function is a key component of our learning-based algorithm. We use different loss terms to achieve plausible results and overcome penetrations. We use the position information of the deformed cloth meshes as the ground truth and calculate the MSE loss between it and the network prediction output. The MSE loss on the positions can be expressed as:

$$\mathcal{L}_p = \frac{1}{N}\sum_{i=1}^{N}\left\|x_p^i - x_g^i\right\|_2, \tag{7}$$

where x_p^i is the position of vertex i on the predicted cloth mesh, x_g^i is the position of vertex i on the ground mesh, N is the number of vertices, and $\|...\|_2$ is the L^2 distance.

In addition, we also use a new type of loss to remove penetrations between the generated cloth and the obstacle. Our goal is to generate non-colliding cloth meshes. The penetration loss between the cloth mesh and the obstacle mesh can be expressed as:

$$\mathcal{L}_e = \frac{1}{N}\sum_{i=1}^{N}\left(d_\varepsilon - \min\left(\left(x_p^i - x_o^i\right)\cdot\mathbf{n}_o^i, d_\varepsilon\right)\right), \tag{8}$$

where x_p^i is the position of vertex i on the cloth mesh and x_o^i is the nearest point to x_p^i on the obstacle mesh. $\mathbf{n}_o^i$ is the normal vector of x_o^i on the obstacle mesh. d_ε is the minimum distance of penetration. N is the number of vertices of the cloth mesh. As shown in Fig. 9, the penetration loss can greatly overcome penetrations between a cloth mesh and a human body. We build the AABB tree of the obstacle to find the nearest point on the obstacle mesh. Fig. 10 shows the effectiveness of the penetration loss on obstacles with different numbers of triangles (0.36k, 0.64k, and 2.75k).

To prevent self-penetrations in the generated cloth mesh, we use the following loss function:

$$\mathcal{L}_s = \frac{1}{N}\sum_{i=1}^{N}\left(d_\varepsilon - \min\left(\left(x_p^i - x_p^j\right)\cdot\mathbf{n}_p^j, d_\varepsilon\right)\right), \tag{9}$$

where x_p^j is the nearest vertex to x_p^i on the cloth mesh. $i \neq j$ and $i, j \in [1, N]$. We concatenate two pieces of cloth with opposite normal vectors of vertices to validate the self-penetration loss in Fig. 11. However, the experiments reveal that Eq. 9 is limited and cannot handle all self-penetrations.

The overall loss function used to predict the cloth deformation is:

$$\mathcal{L} = \mathcal{L}_p + \lambda\mathcal{L}_e + \mu\mathcal{L}_s, \tag{10}$$

where λ and μ are blending coefficients. In practice, we use $\lambda = \mu = 1$ and observe good results for all the benchmarks with these values. Since the predictions tend to be random at the beginning of the training, Eq. 8 and Eq. 9 may result in inaccurate predictions. We use Eq. 10 to train the network at the beginning and add Eq. 8 and Eq. 9 during the final training process. The number of parameters of our network and the computed gradients will hinder the scalability of training on large meshes. Therefore, Eq. 7 is computed on a GPU, while Eq. 8 and Eq. 9 are computed on a CPU.

3.6. 3D Cloth Prediction

With the trained network, we can obtain the predicted cloth mesh by inputting the initial cloth mesh and the target obstacle mesh. Since the initial cloth mesh representation has a fixed topology for a specific cloth, we only need to input the target obstacle mesh; our network is used to predict the deformed target cloth mesh.

4. Implementation and Performance

In this section, we describe our implementation and highlight the results on many benchmarks. We also compare the performance with prior physics-based simulators and learning-based methods.

4.1. Implementation

We have implemented our algorithm on a standard PC (Ubuntu 18.04.4 LTS/Intel I7 CPU@4.2G Hz/8G RAM, NVIDIA GeForce RTX 3090 GPU). We perform both network training and cloth prediction on the same platform. Our implementation uses PyTorch 1.7.0 and Python 3.8.8 as the underlying development environment.

Datasets: Our mesh-based network can handle various types of cloth and obstacles. We evaluate its performance on many different cloth meshes and obstacle meshes. We consider three types of benchmark scenarios to evaluate our approach:

No.	Obstacle	Cloth	Cloth triangles	Obstacle triangles	Key frames	Mean error
1	Bunny	Curtain	16384	12000	4839	2.47E-03
2	QMan	Tshirt	16148	14664	9247	3.46E-03
3	SMPL pose		16116	55104	2163	2.45E-03
4	SMPL shape		16116	55104	2779	3.15E-03
5	Andy	Dress	15176	12999	9059	3.78E-03
6		Jacket	11054	12999	9563	4.18E-03
7		Pants	16118	12999	8433	3.59E-03
8		Tshirt	12392	12999	9521	3.20E-03
9	QMan	Robe_20k	19168	14664	8829	4.86E-03
10		Robe_100k	100404	14664	8702	5.02E-03

Figure 5: *We evaluate the performance of our mesh-based network on benchmarks with various characteristics: different types of obstacles, cloths with different shapes, different resolutions, etc. We highlight the number of triangles for the cloth and the obstacles, the number of key-frames, and the mean vertex position error(in meters) for all unseen test frames between the predictions and ground truth simulated by a physics-based simulation ArcSim [NSO12].*

- **Rigid obstacles:** We lead a rigid bunny model through a hanging cloth from different positions in the scene. The deformation of the cloth is obtained by the simulator ArcSim [NSO12; NPO13; PNDO14]. At each position, we relax the cloth for a period of time to generate a quasi-static deformation. We also simulate the results of the cloth on the bunny under different rotations and scales. The size of the bunny is scaled from 0.5 to 1.7. To ensure the uniqueness of the cloth covering on the bunny, we mark each side of the cloth as 1 or -1. We also label the side that is in contact with the bunny according to the vertex attributes of the bunny.
- **SMPL human body model:** For the SMPL humans, we choose the representative data provided by TailorNet [PLP20]. Considering that the SMPL bodies in Tailornet have various shapes and postures, we select two types of data. One is the SMPL body data with 2779 different postures in a fixed body shape, The other is the SMPL body data with 9 different body shapes in several fixed postures. This selection of data also facilitates comparison of results from our network and Tailornet. For these human bodies, we generate their triangle meshes from the SMPL parameters.
- **Non-SMPL human body model:** We also generate non-SMPL humans, including a child Andy and an adult male Qman. We upload the humans with canonical poses to the website https://www.mixamo.com/ and download about 90 different action sequences. For these action sequences, we use the physics-based simulator ArcSim [NSO12; NPO13; PNDO14] to generate clothes on them. To eliminate dynamic effects, we perform linear interpolation between the adjacent poses and relax the cloth on each pose for a period of time to ensure the cloth is as static as possible. We transform all the human body meshes to the origin of the coordinates to eliminate the absoluteness of the position. The relative coordinates will enhance the generalizability of the network.

Our network can also predict different types of clothes. The child, Andy, wears different types of clothes, including a t-shirt, pants, a jacket and a dress. The jacket and dress are loose, and their deformation is different from the t-shirt and pants. These clothing simulations are also obtained by ArcSim [NSO12; NPO13; PNDO14], as above.

We evaluate the accuracy of our predicted meshes by measuring the mean error of each benchmark (as shown in Fig. 5) with the following equation:

$$\mathcal{E} = \frac{\sum_{j=1}^{M} \frac{\sum_{i=1}^{N} \left\| x_P^{i,j} - x_G^{i,j} \right\|}{N}}{M}, \tag{11}$$

where M is the number of animation key frames. N is the number of vertices in the 3D mesh, and $x_P^{i,j}$ and $x_G^{i,j}$ are the positions of vertex i on frame j. The unit of our mean error is meters. The details for each scene are shown in Fig 5.

Network Training: Following [PLP20] and [WSFM19], the dataset is split for training and testing. For test data, we select 800 bunny models with different positions, rotations and scales that have not been seen during training. For the SMPL humans, we use the training and testing split provided in TailorNet [PLP20]. For other action sequences obtained from https://www.mixamo.com/, 90 action sequences are used during training, which produce approximately 9,000 samples. To demonstrate the generalizability of our network, during the test, we download 10 other action sequences that were unseen during training from https://www.mixamo.com/ and predict the results of these new action poses.

Moreover, the obstacle and cloth meshes in our scene have different vertices and topologies. Thus, networks used for different scenarios have different numbers of parameters. Therefore, we train an exclusive network for each scenario. The training time for each benchmark varies from 24 hours to 7 days.

To accelerate the convergence of the network, we normalize the input and output vertex positions to zero mean and unit variance for all frames. During training, we uniformly reduce the learning rate from $1e-3$ to $1e-5$. We use an Adam optimizer [KB14] to train the parameters of the neural network.

Penetrations: Our learning-based method uses the penetration loss function highlighted in Eq. 8, which is designed to prevent cloth-object penetrations or cloth self-collisions. In our benchmarks, we do not observe any deep or noticeable penetrations, though the learning-based method does not guarantee a non-penetrating final mesh. In our physics-based simulator, we use a large repulsion thickness (i.e., 1mm) so that the training data is not only collision-free but there is some distance between non-adjacent mesh ele-

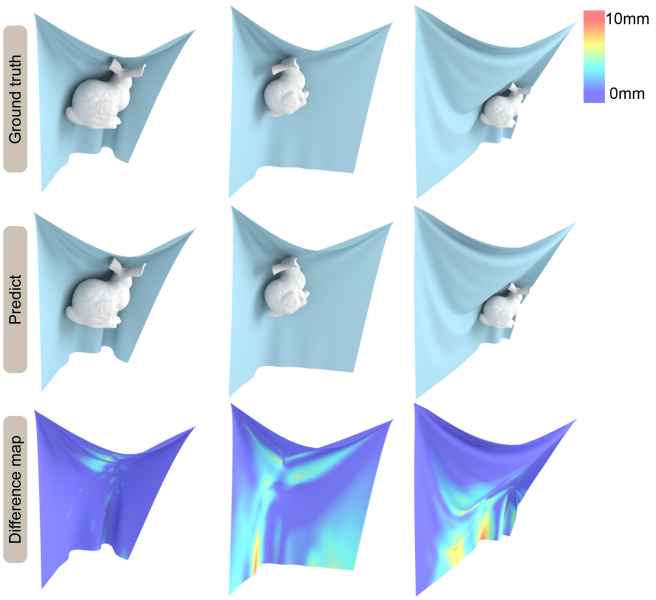

Figure 6: *The hanging cloth draping on bunnies in different positions, rotations, and scales that are unseen in training. Compared to the ground truth computed using ArcSim (top row), our predicted meshes (second row) result in visually plausible results. The deviation between the ground truth mesh and our predicted mesh is shown in the bottom row, with the error bounded by 10mm.*

ments. This use of repulsion distance further reduces the chances of self-penetrations or collisions in the predicted cloth mesh. If the predicted mesh has a few collisions, we can solve them by simple post-processing.

4.2. Results on Diverse Scenes

In this section, we highlight the performance of our method on different benchmarks and compare the accuracy with physically-based simulation results. All predictions are performed on new test sets that are different from the training data.

Fig. 6 highlights our results on scenes with obstacles unseen during training corresponding to moving, rotating, or scaling rigid bodies. Our network results in favorable generalization to obstacles with unseen locations, rotations, and scales. Our approach makes no assumption about the topology of the obstacles or the cloth. We also compare the accuracy with ArcSim (an accurate physics-based simulator) and observe a high level of similarity between our predicted 3D mesh and the ground truth mesh. The mean deviation error between the vertices is less than 5mm in our benchmarks. Fig. 7 highlights our results on different human body models. We use the same cloth mesh corresponding to a t-shirt on different human models. In Fig. 7, all the human bodies are represented with triangle meshes. All predictions are on an unseen test set. For example, the predictions of Andy and Qman are on the new action sequences downloaded from https://www.mixamo.com/. The results of SMPL are on the test data split from TailorNet. For all these benchmarks, our predicted results are visually close to the

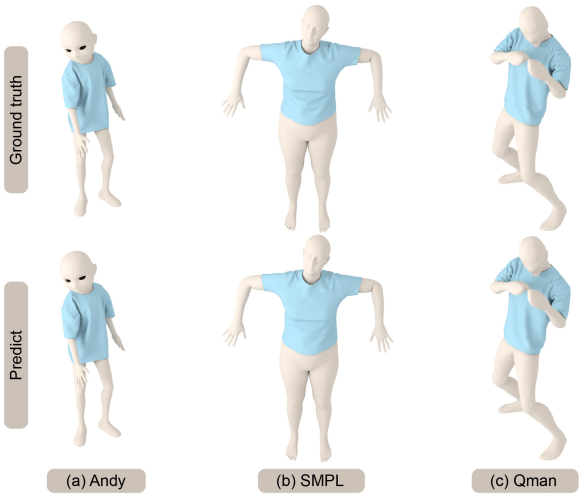

Figure 7: *We show the deformation on a cloth mesh corresponding to a t-shirt on different unseen human models. All these human models are represented using triangle meshes. The human model (b) is generated from SMPL parameters. For all these benchmarks, our predictions are visually close to the ground truth.*

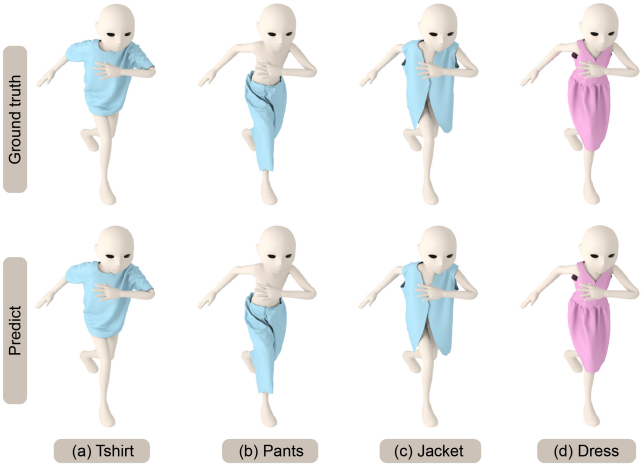

Figure 8: *We highlight the performance of our network on different clothing types corresponding to a t-shirt, pants, a jacket, and a dress with unseen actions. We observe that our predicted mesh is close to the ground truth and generates similar wrinkles and folds.*

ground truth. Figs. 8 highlights our results on cloths of different types. For all these benchmarks, our algorithm can generate plausible results that match the ground truth meshes. For more benchmarks about cloths with different resolutions and multiple disjoint obstacles, please check out our supplementary material.

Fig. 9 highlights the benefits of our penetration handling approach based on a loss function. By adding the penetration term into the loss function, our algorithm tends to alleviate the penetrations and self-collision artifacts. Although no penetration is unavailable in the predictions on the test set, it can reduce the de-

Figure 9: *With the penetration term in our loss function (shown in Eq. 8), our learning algorithm can significantly reduce the number of penetrations (shown on the right) compared to the result without penetration loss(shown in the middle).*

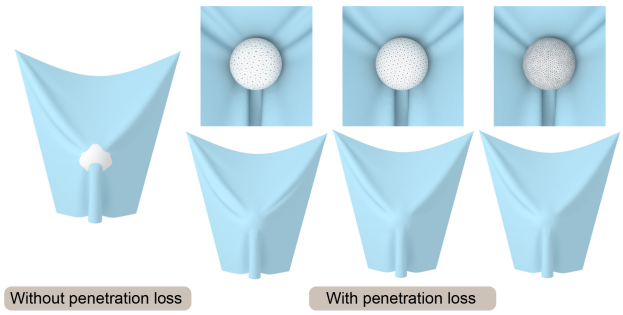

Figure 10: *The results of Eq. 8 with different obstacle discretizations.*

gree of penetration and the subsequent processing work. Fig. 10 and Fig. 11 highlight the benefits of Eq. 8 (with different obstacle discretizations) and Eq. 9, where (self-)penetrations are effectively alleviated by these loss functions.

To sum up, our network can not only handle SMPL and non-SMPL human bodies, but also rigid obstacles. Our network can also process various types of clothes without providing skin models for those clothes. Compared with the previous method, our network can handle more scenarios. The predictions also show that our network can satisfactorily generalize to new, unseen data. Even when trained to predict a static deformed cloth mesh, our network generates a series of deformed cloth with fine temporal coherence on an obstacle sequence (shown in the video).

5. Comparisons

In this section, we qualitatively and quantitatively compare the performance of our network with prior learning-based methods.

5.1. Diverse Scenarios

In Table 1, we list the characteristics of different learning-based methods. We highlight the capabilities of different methods in terms of the kind of obstacles they can handle (e.g., SMPL models only or rigid objects). Compared to prior methods, our approach makes no assumptions about the type or topology of the

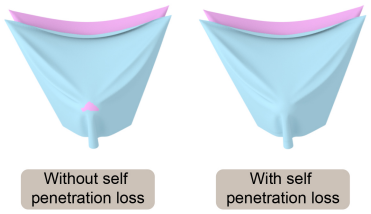

Figure 11: *The results with and without the self-penetration term in the loss function (shown in Eq. 9).*

Table 1: *We compare the characteristics of our approach with prior learning-based methods. Some of these learning-based methods are limited to parametric human models (e.g., SMPL).*

Method	SMPL body	Non-SMPL body	Triangle mesh
TailorNet[PLP20]	✔	✗	✗
DeePSD [BME20a]	✔	✗	✗
[SOC19]	✔	✗	✗
GarNet [GCS*19]	✔	✔	✗
[WSFM19]	✔	✔	✗
[BME20b]	✔	✗	✗
DRAPE [GRH*12]	✔	✗	✗
Our method (N-Cloth)	✔	✔	✔

cloth or the obstacles in the scene. Most previous methods [PLP20; BME20a; SOC19; BME20b] are based on the SMPL model, which limits results to the SMPL human model. Other methods [GCS*19; GRH*12; WSFM19] are limited to human models represented using joints. Although they can handle the non-SMPL human body, they are unable to process other obstacles such as a bunny. [HDDN19] is a complimentary method that uses PCA and subspace-only physics simulation. However, it recurrently inputs the previous prediction and accumulates errors. This makes the predicted cloth mesh appear flat with fewer wrinkles. Our mesh-based method overcomes these limitations and can handle multiple, disjoint obstacles. Our predictions have no accumulation errors and can retain fine details like wrinkles and folds.

5.2. Qualitative Comparisons

We perform a detailed comparison of our method with TailorNet [PLP20], as the code and dataset are easily available. In Fig. 12, we use the same dataset as TailorNet for network training. We compare the accuracy of the predicted cloth meshes generated by our method and TailorNet. We compute the difference maps for each mesh by comparing the results with the ground truth mesh. We observe that our method predicts similar output meshes with richer details. In addition, our method produces fewer vertex errors compared to the ground truth than TailorNet. In benchmarks with many or detailed wrinkles, we observe that our network generates better results than TailorNet, as shown in Fig. 13. For example, our prediction of the cloth mesh has more wrinkles in the belly and shoulder areas, while TailorNet's prediction is flatter.

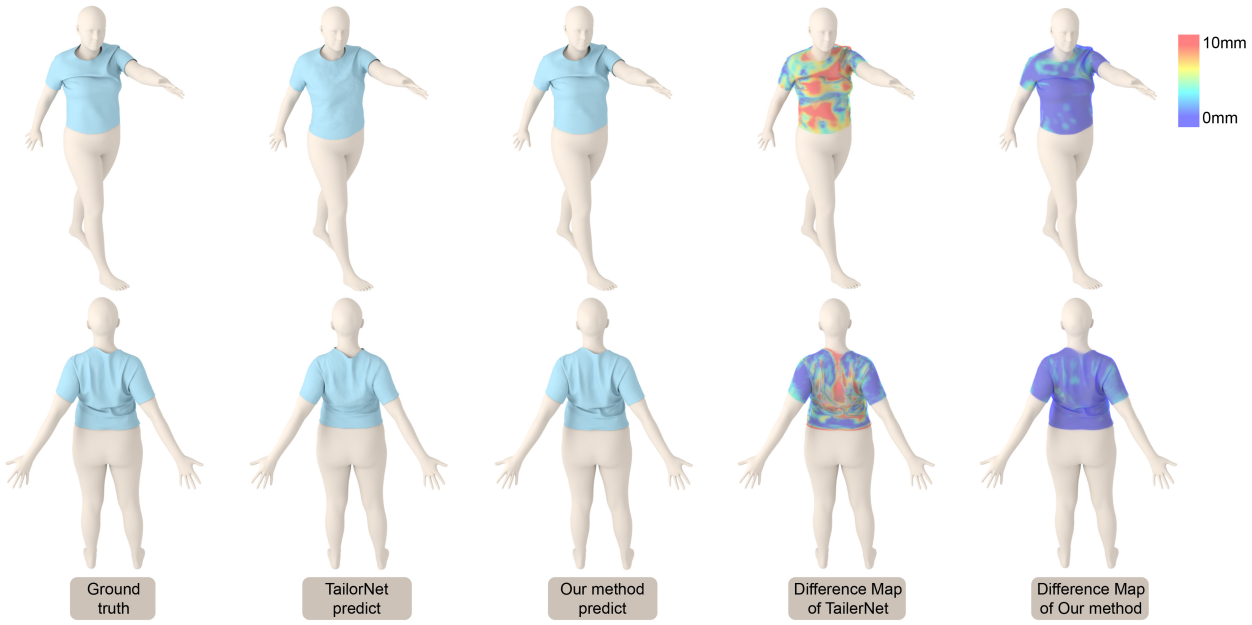

Figure 12: *We compare the performance of our approach with TailorNet [PLP20] on unseen poses. We use the same dataset, available as part of TailorNet for network training. We compare the accuracy of predicted meshes generated using TailorNet and those generated using our method. We also compare the accuracy with the ground truth. We highlight the vertex error for each learning-based method by computing the difference maps with the ground truth mesh. We get results similar to TailorNet's, but with richer details.*

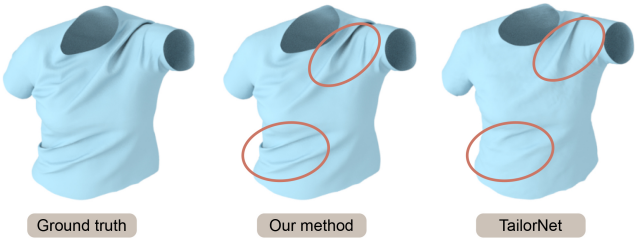

Figure 13: *Our method generates better results than TailorNet in terms of preserving wrinkles, as shown in the circled areas.*

5.3. Quantitative Comparisons

We use the following error metrics for quantitative comparison between our mesh-based network and TailorNet:

$$\begin{aligned} \mathcal{E}_{dist} &= \frac{1}{N}\sum_{i=1}^{N}\left\|x_P^i - x_G^i\right\| \\ \mathcal{E}_{lap} &= \frac{1}{N}\sum_{i=1}^{N}\left\|\Delta(P) - \Delta(G)\right\| \\ \mathcal{E}_{norm} &= \frac{1}{N}\sum_{i=1}^{N}\arccos\left(\frac{(y_P^i)^T y_G^i}{\left\|y_P^i\right\|\left\|y_G^i\right\|}\right) \end{aligned} \tag{12}$$

where x_P^i is the position of vertex i of the predicted mesh P. x_G^i is the position of its corresponding vertex on the ground truth mesh G. y_P^i and y_G^i are the normal vector at x_P^i and x_G^i, respectively. N is the number of vertices of the cloth mesh. Δ is the Laplace operator.

Table 2 shows the error mean and variance of our method and the predictions of Tailornet with ground truth for all test frames. The statistical value of the prediction error of our method is significantly lower than that of TailorNet.

Table 2: *We compare the mean and standard deviations of mesh errors for our method and TailorNet on test frames based on the ground truth.*

Evaluation	TailorNet	Our Method
mean $\mathcal{E}_{dist}$(m)	7.90E-3	2.45E-3
std $\mathcal{E}_{dist}$(m)	2.74E-3	0.54E-3
mean $\mathcal{E}_{lap}$	1.94E-2	6.97E-3
std $\mathcal{E}_{lap}$	4.44E-3	1.20E-3
mean $\mathcal{E}_{norm}(^\circ)$	10.81	4.40
std $\mathcal{E}_{norm}(^\circ)$	2.45	1.02

5.4. Performance Analysis

We implement each layer of the cloth encoder and obstacle encoder with MLP as the baseline. The decoder in our network uses MLP, so we keep it invariable. Figure 15 shows the GPU memory usage when our network and baseline predict a single mesh. Our network occupies less GPU memory when making predictions. In experiments, it is revealed that more GPU memory is required for baseline training, which makes it impossible to train meshes with higher

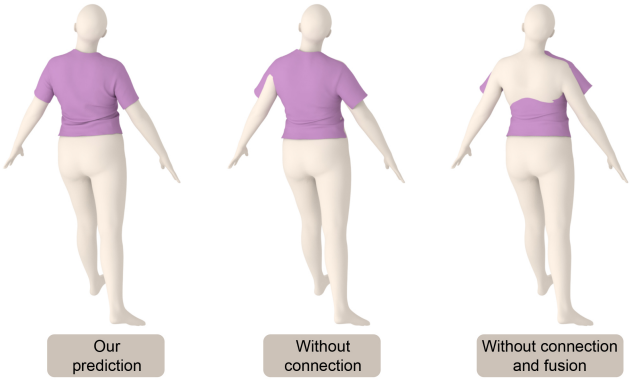

Figure 14: *We discard the connections in the decoder and concatenate the cloth vector and obstacle vector in latent space as variants. The results of these variants and our network are shown above.*

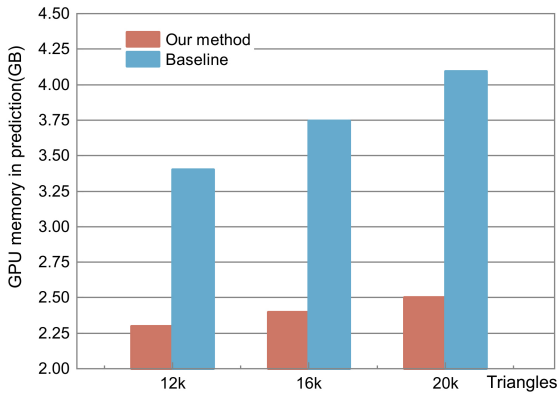

Figure 15: *We implement the cloth encoder and obstacle encoder with MLP as the baseline. We compare the GPU memory of our network and the baseline while predicting a single cloth mesh at runtime.*

resolutions. However, our method is able to train cloth meshes with more than 100k triangles.

We have compared the running time of our method with a GPU-based physics-based simulator called I-Cloth [TWL*18], as shown in Figure 16. Compared to I-Cloth, our network achieves an order of magnitude performance improvement. Furthermore, the running time of our method does not change much with a higher resolution mesh. The overall accuracy and visual fidelity of the cloth mesh generated by our method are similar to those of I-Cloth.

We observe that our approach can obtain an interactive frame rate (about $30-45$ fps on an NVIDIA GeForce RTX 3090 GPU). Compared with TailorNet, our method has no obvious advantage in running time. This is because the input of the SMPL model is a small number of parameters, while our method inputs a mesh with all the vertex and edge information. The human mesh provided by TailorNet has 55k triangles, which will increase the calculation time of our network. In practice, we concentrate more on the deformation of the cloth mesh. Therefore, we can simplify the human body mesh, which will accelerate our network.

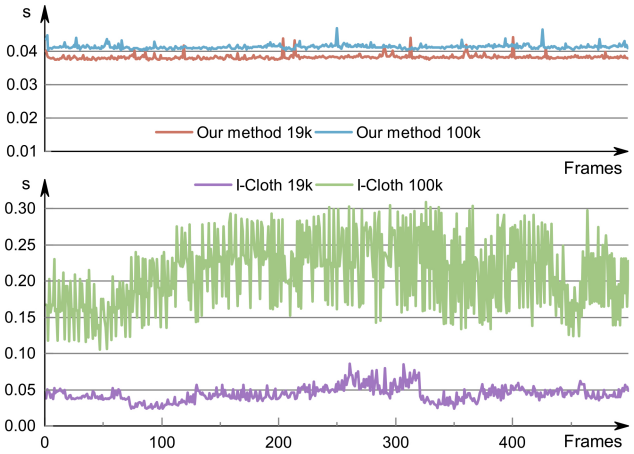

Figure 16: *Compared to a GPU-based physics-based simulator, I-Cloth [TWL*18] (bottom), our method (top) results in faster performances ($5-8X$ faster). We highlight the performance for cloth meshes with $19K$ and $100K$ triangles. The performance of our method is almost the same for a high-resolution mesh.*

5.5. Ablation Experiments

We implement a series of ablation experiments to verify the effectiveness of our network architecture. We remove the connections of each layer of the obstacle encoder in the decoder as a variant of our network. Furthermore, we discard the fusion network and use simple concatenation in the latent space as another variant. Figure 14 shows the predictions of our network and its two variants. Our network architecture plays an indispensable role in the convergence of results.

6. Conclusion, Limitations, and Future Work

We present a novel mesh-based network for interactive 3D cloth prediction. Our approach is general and does not make any assumption about the topology or connectivity of the cloth or the obstacles in the scene. Our approach can handle complex cloth simulation benchmarks and predict the deformed 3D mesh at about $30-45$ fps on a commodity mesh. To the best of our knowledge, ours is the first general learning-based method that can handle arbitrary obstacle meshes and many types of cloths.

Limitations: Our approach has some limitations. It requires considerable time to generate the training data, and it can take a few days to generate synthetic training datasets using a physics-based simulator. Furthermore, our approach assumes that the topology and connectivity of the cloth mesh is fixed. If the topology changes, we need to repeat the training step. This approach may work well for human models used for virtual try-on or dressing, as they have fixed topologies. Like prior learning-based methods, we cannot provide any rigorous guarantees in terms of absolute accuracy or collisions in our predicted mesh. The effectiveness of

our self-penetration is limited and may introduce undesirable new collisions. The computation of self-penetration depends on the discretization of the cloth mesh. We propose to different loss functions in the future to handle such self-penetrations. Or we can combine our approach with learning-based methods for collision handling [TPM21; TPS*21]. Moreover, compared with the SMPL model which only uses a few parameters, our network performs feature extraction and fusion on the complete mesh. This results in slower performance of our network, though we observe interactive performance of 30-45fps.

There are many avenues to improve the performance in the future. Our current approaches for synthetic data generation, training, and runtime prediction are not optimized, and it is therefore possible to improve the performance. We would like to incorporate better geometric learning-based methods that can account for highly dynamic obstacles as well as small changes in mesh topology or connectivity. It will be interesting to use visual knowledge [Pan21] for cloth deformation prediction. Finally, we would like to integrate our approach with different applications corresponding to virtual try-on or gaming and evaluate the performance.

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

References

[BEB12] BROCHU, TYSON, EDWARDS, ESSEX, and BRIDSON, ROBERT. "Efficient Geometrically Exact Continuous Collision Detection". *ACM Trans. Graph.* 31.4 (2012), 96:1–96:7 2.

[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.

[BME20a] BERTICHE, HUGO, MADADI, MEYSAM, and ESCALERA, SERGIO. "DeePSD: Automatic Deep Skinning And Pose Space Deformation For 3D Garment Animation". *arXiv preprint arXiv:2009.02715* (2020) 2, 3, 8.

[BME20b] BERTICHE, HUGO, MADADI, MEYSAM, and ESCALERA, SERGIO. "Physically Based Neural Simulator for Garment Animation". *arXiv preprint arXiv:2012.11310* (2020) 2, 3, 8.

[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.

[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.

[CGY*21] CHEN, LAN, GAO, LIN, YANG, JIE, et al. "Deep Deformation Detail Synthesis for Thin Shell Models". *arXiv preprint arXiv:2102.11541* (2021) 5.

[CMM*20] CHENTANEZ, NUTTAPONG, MACKLIN, MILES, MÜLLER, MATTHIAS, et al. "Cloth and skin deformation with a triangle mesh based convolutional neural network". *Computer Graphics Forum*. Vol. 39. 8. Wiley Online Library. 2020, 123–134 2, 5.

[CPA*21] CORONA, ENRIC, PUMAROLA, ALBERT, ALENYA, GUILLEM, et al. "SMPLicit: Topology-aware generative model for clothed people". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, 11875–11885 3.

[CZY20] CHEN, LAN, ZHANG, XIAOPENG, and YE, JUNTAO. "Multi-feature super-resolution network for cloth wrinkle synthesis". *arXiv preprint arXiv:2004.04351* (2020) 5.

[dASTH10] DE AGUIAR, EDILSON, SIGAL, LEONID, TREUILLE, ADRIEN, and HODGINS, JESSICA K. "Stable spaces for real-time clothing". *ACM Trans. Graph. (SIGGRAPH)* 29 (4 July 2010), 106:1–106:9. ISSN: 0730-0301 2.

[FYK10] FENG, WEI-WEN, YU, YIZHOU, and KIM, BYUNG-UCK. "A deformation transformer for real-time cloth animation". *ACM Trans. Graph. (SIGGRAPH)* 29.4 (July 2010), 108:1–108:9. ISSN: 0730-0301 2.

[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) 2, 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 2, 3, 8.

[GJ19] GAO, HONGYANG and JI, SHUIWANG. "Graph U-Nets". *international conference on machine learning*. PMLR. 2019, 2083–2092 3, 4.

[GLM05] GOVINDARAJU, NAGA K, LIN, MING C, and MANOCHA, DINESH. "Quick-cullide: Fast inter-and intra-object collision culling using graphics hardware". *IEEE Proceedings. VR 2005. Virtual Reality, 2005.* IEEE. 2005, 59–66 2.

[GRH*12] GUAN, PENG, REISS, LORETTA, HIRSHBERG, DAVID A, et al. "Drape: Dressing any person". *ACM Transactions on Graphics (TOG)* 31.4 (2012), 1–10 8.

[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, 8.

[HKS17] HOLDEN, DANIEL, KOMURA, TAKU, and SAITO, JUN. "Phase-functioned neural networks for character control". *ACM Transactions on Graphics (TOG)* 36.4 (2017), 1–13 4.

[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.

[KB14] KINGMA, DIEDERIK P and BA, JIMMY. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980* (2014) 6.

[KCZO07] KAVAN, LADISLAV, COLLINS, STEVEN, ZARA, JIRI, and O'SULLIVAN, CAROL. "Skinning with Dual Quaternions". *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*. I3D '07. New York, NY, USA, 2007, 39–46 3.

[KKN*13] KIM, DOYUB, KOH, WOOJONG, NARAIN, RAHUL, et al. "Near-exhaustive Precomputation of Secondary Cloth Effects". *ACM Trans. Graph (SIGGRAPH).* 32.4 (July 2013), 1–8. ISSN: 0730-0301 2.

[KW16] KIPF, THOMAS N and WELLING, MAX. "Semi-supervised classification with graph convolutional networks". *arXiv preprint arXiv:1609.02907* (2016) 4.

[LBK17] LIU, TIANTIAN, BOUAZIZ, SOFIEN, and KAVAN, LADISLAV. "Quasi-newton methods for real-time simulation of hyperelastic materials". *Acm Transactions on Graphics (TOG)* 36.3 (2017), 1–16 2.

[LBOK13] LIU, TIANTIAN, BARGTEIL, ADAM W, O'BRIEN, JAMES F, and KAVAN, LADISLAV. "Fast simulation of mass-spring systems". *ACM Transactions on Graphics (TOG)* 32.6 (2013), 1–7 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 2.

[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 2, 6.

[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 2, 6.

[Pan21] PAN, YUNHE. "Miniaturized five fundamental issues about visual knowledge". *Frontiers Inf. Technol. Electron. Eng.* 22.5 (2021), 615–618 11.

[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, 6, 8, 9.

[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 6.

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

[Pro97] PROVOT, XAVIER. "Collision and Self-collision Handling in Cloth Model Dedicated to Design Garments". *Graphics Interface*. 1997, 177–189 2.

[RAS17] ROCCO, IGNACIO, ARANDJELOVIC, RELJA, and SIVIC, JOSEF. "Convolutional neural network architecture for geometric matching". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, 6148–6157 4.

[RHBL07] RANZATO, MARC'AURELIO, HUANG, FU JIE, BOUREAU, Y-LAN, and LECUN, YANN. "Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition". *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, 1–8 2, 3.

[SOC19] SANTESTEBAN, IGOR, OTADUY, MIGUEL A, and CASAS, DAN. "Learning-based animation of clothing for virtual try-on". *Computer Graphics Forum*. Vol. 38. 2. Wiley Online Library. 2019, 355–366 2, 3, 8.

[STOC21] SANTESTEBAN, IGOR, THUEREY, NILS, OTADUY, MIGUEL A, and CASAS, DAN. "Self-Supervised Collision Handling via Generative 3D Garment Models for Virtual Try-On". *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021) 2, 3.

[TPM21] TAN, QINGYANG, PAN, ZHERONG, and MANOCHA, DINESH. *LCollision: Fast Generation of Collision-Free Human Poses using Learned Non-Penetration Constraints*. 2021. arXiv: 2011.03632 [cs.GR] 11.

[TPS*21] TAN, QINGYANG, PAN, ZHERONG, SMITH, BREANNAN, et al. *Active Learning of Neural Collision Handler for Complex 3D Mesh Deformations*. 2021. arXiv: 2110.07727 [cs.CV] 11.

[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, 10.

[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*. Vol. 35. 2. Wiley Online Library. 2016, 511–521 2.

[WCC*21] WU, NANNAN, CHAO, QIANWEN, CHEN, YANZHEN, et al. "Example-based Real-time Clothing Synthesis for Virtual Agents". *arXiv preprint arXiv:2101.03088* (2021) 2, 3.

[WHRO10] WANG, HUAMIN, HECHT, FLORIAN, RAMAMOORTHI, RAVI, and O'BRIEN, JAMES. "Example-based wrinkle synthesis for clothing animation". *ACM Trans. Graph. (SIGGRAPH)* 29.4 (July 2010), 107:1–107:8. ISSN: 0730-0301 2.

[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, 6, 8.

[ZBO13] ZURDO, JAVIER S., BRITO, JUAN P., and OTADUY, MIGUEL A. "Animating Wrinkles by Example on Non-Skinned Cloth". *IEEE Trans. Vis. Comp. Graph.* 19.1 (2013), 149–158. ISSN: 1077-2626 2.

[ZCWM21] ZHANG, MENG, CEYLAN, DUYGU, WANG, TUANFENG, and MITRA, NILOY J. "Dynamic Neural Garments". *arXiv preprint arXiv:2102.11811* (2021) 3.

[ZWCM21] ZHANG, MENG, WANG, TUANFENG, CEYLAN, DUYGU, and MITRA, NILOY J. "Deep detail enhancement for any garment". *Computer Graphics Forum*. Vol. 40. 2. Wiley Online Library. 2021, 399–411 2.