Chapter 6: Dynamics-Inspired Garment Draping Prediction

6.1 Introduction

Other than accelerated cloth simulation proposed in Chapter 5, one can also use deep learning to directly predict the garment draping given the body as input. This approach offers real-time feedback to the user guaranteed by GPU-based network inference. Within the training distribution, learning-based methods can also produce draping results as good as simulated ones, accurately revealing information, such as fitting and material perception.

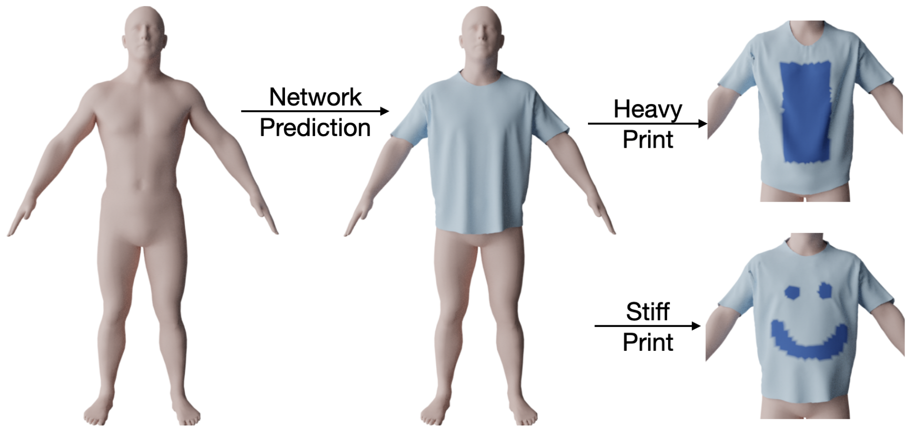

Figure 6.1: My model learns how to drape garments in two stages: first supervising the network with a physics-inspired loss, then optimizing the output according to the physics of specified garments, such as new print shapes or materials, in a self-correcting way.

However, there exists a number of major challenges for interactive learning-based virtual try-on. First, it is well-known that machine learning models tend to produce overly smoothed results when using per-vertex distances as their main loss [11, 180, 181]. Although previous works [11] addressed this issue to some extent, they are limited to a narrow set of shapes composed of nine distinct bodies. Next, while garments are often composed of different materials (*e.g.* the frontal graphics print in T-shirts), existing works typically model a single one because it is impractical to consider all combinations of different graphics prints shapes at training time. Finally, it is expected that the garments predicted by the network model is fit-accurate, although the most common losses in related work (per-vertex geometry errors) do not necessarily entail fit accuracy. This translates into changes in the overall shape of the garment and violations to its material properties.

In this chapter, I propose a novel semi-supervised framework to address all issues above. One key idea is that physical constraints can be reformulated as geometric loss functions that can be optimized during training. However, using the exact physical loss functions from scratch does not result in good draping due to their complexity. Therefore, I first train my model with supervised, physics-inspired loss functions, and then optimize the model output individually for each sample to conserve the actual physical energy and minimize/eliminate geometric intersections. Given their superior quality compared to the initial predictions, the resulting optimized samples can then be re-used to further improve the initial feed-forward network.

Overall, the key contributions of this work include:

- A semi-supervised framework that enables easy integration of constraints into the deep learning model.
- Introduction of novel loss functions that encode geometric, physical, design, and tailoring constraints.
- A novel encoder/decoder network that effectively captures global and local features from the input and dynamically aggregates neighborhood information.
- A new self-correcting method based on data augmentation that enables both more accurate predictions and reduced data preparation time.

6.2 Related Work

Drape prediction systems fall mainly in three categories: physics-based simulation, learning-based garment generation, and direct estimation of garments from real-world.

Physics-based Cloth Simulation. Physics-based garment simulation systems usually include spatial discretization [27, 91] and different forms of simulations [24, 182]. Although several techniques have been proposed to speed up cloth simulation, including GPU acceleration [162, 183], spatial and temporal parallelization [152, 170, 184, 185], and other techniques [186, 187, 188], real-time, physically accurate cloth dynamics for any given human shape remains illusive [74].

To reduce computation time, several works produce high frequency wrinkles on low-resolution meshes as a practical trade-off [173, 178, 186, 189, 190], but are typically limited to tight garments. In tight garments, cloth material and overall drape have a negligible effect in the wrinkles. In contrast, my feed-forward model can achieve real-time performance even when generating high-resolution garments. Inspired from optimization-based simulation [191], I use the physics-based metrics as some of the loss functions, resulting in realistic folds and wrinkles.

Learning-based Garment Draping. As a faster alternative to simulation, learning-based approaches have been developed for draping garments with better visual quality and realism, including normal map generation [124], KNN body-garment fusion [8], wrinkle-assisted design [9], displacement regression [192, 193], motion-conditioned auto-encoder [194], and least-square approximation [195]. However, these works are limited in different aspects: [124] do not provide geometric details; [8, 193, 195] generate relatively smoothed results; [193, 194, 195] do not generalize to a wide range of body shapes; [9] requires user knowledge of wrinkle formation; and [192] cannot deal with loose clothing and produces a single mesh containing both body and garment fused. In contrast, my method takes only a human body mesh as input and directly regresses the garment mesh as output with realistic geometric details.

Other works tackle the same task as my system does. Santesteban *et al.* [180] used RNN to capture the motion-dependent garment shapes on the body. Vidaurre *et al.* [181] extended the approach to cover more size variations of different design parts. Since [180, 181] supervise their methods with a pure per-vertex distance loss, their model suffers from smoothed folds especially when the garment is loose. To remedy the smoothing effect, Patel *et al.* [11] interpolated high frequency results from different anchors based on the inverse of per-vertex distances between garment predictions on the canonical pose. Despite its success, it is based on a limited discrete set of training shapes, which can pose generalization problems as can be observed in Sec. 6.5.6. In contrast to the three methods above, I design an exhaustive set of loss functions inspired by physics constraints, including the minimization of differences in the spectral domain.

Garment Capture and Estimation. Instead of simulating garments, some systems capture garments from real-world data. These systems focus on inferring the mesh geometry using visual features and retargeting it to the desired body. State-of-the-art methods have varying types of input, including images [12, 68, 110, 116, 120, 121, 196], mesh scans [22, 197], RGB videos [2, 198], or videos together with the depth channel [122, 199]. Despite the appeal of avoiding altogether physical simulation in both train and test stages, capturing methods still have a number of drawbacks. First, it is hard to collect a dataset large enough to train a model that can cover the large variation of human shapes. Moreover, simple retargeting often fails to account for non-rigid and non-linear transitions of the garment deformation between different bodies. Note that there is an intrinsic difference between my method and capture systems: while my method focuses on generating drapes *given just the underlying body*, capture methods extract the garment from existing media.

6.3 Method

In this work, I focus on a wide range of human body sizes rather than different poses. To this end, I used the SMPL model [30] to generate a set of 20,000 bodies of varied male shapes (see supp. mat. for the equivalent female model) following a uniform distribution of BMIs between 15 and 65, all with the same A-pose. In comparison, my analysis indicates that from the 9 discrete bodies used by TailorNet [11] only one has a BMI over 30. Given that 42.4% of the population in the US has obesity (BMI > 30) [200], TailorNet does not have enough coverage on large bodies.

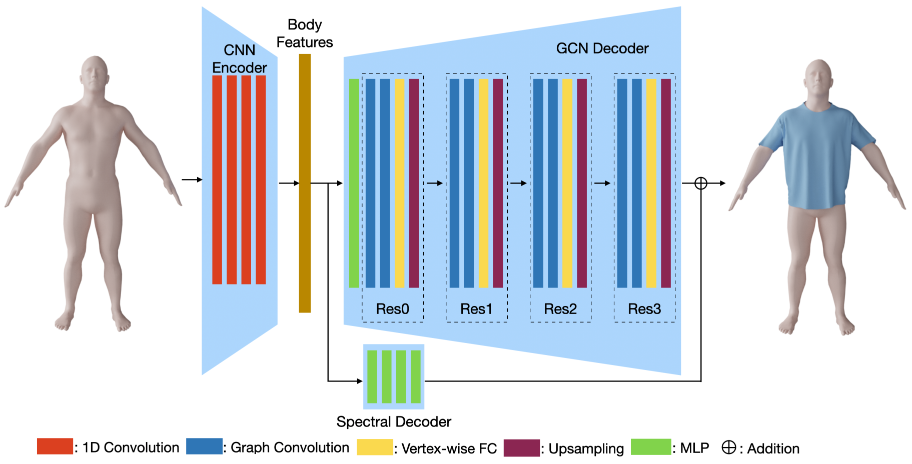

Figure 6.2: Overall structure of my network. I use 1D CNN as my encoder and Graph Convolutional Network (GCN) as the decoder with four different resolutions. I additionally employ an MLP-based spectral decoder for high frequency preservation.

The overall structure of my network is shown in Fig. 6.2. I first pass the ordered vertex coordinates of the body to a 1D CNN network to extract features (Sec. 6.3.1). Next, I transform the features corresponding to vertices in the body using 1-layer MLP, and distribute them to the vertices of the garment according to fixed correspondences between body and garment. I design a resolution hierarchy for the Graph Convolutional Network (GCN) decoder to capture both global and local information (Sec. 6.3.2). To improve the higher frequency wrinkles, I finally introduce an MLP branch that predicts the garment residuals in the spectral domain (Sec. 6.3.3). The losses used for training this network are described in Sec. 6.3.4.

6.3.1 Encoder

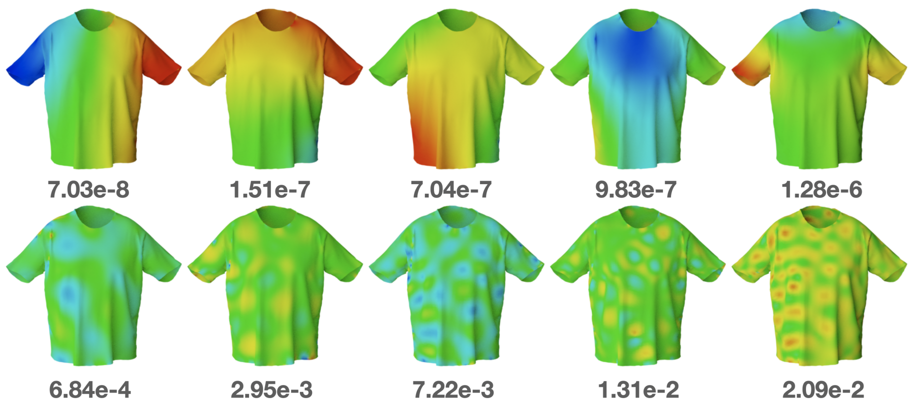

Figure 6.3: Visualization of the eigenvectors and their eigenvalues. Red denotes large weights, blue small ones.

Unlike PointNet [99], which obviates the vertices order by spatially pooling its features, I would like to exploit the fixed topology of the input SMPL vertices. Since the input mesh does not have a neighbor structure constant across vertices, a natural choice for this would be a GCN. However, in Sec. 6.5.2 I show that my simpler alternative, a 1D CNN, empirically outperforms the GCN. I hypothesize that this might be due to the large amount of model capacity devoted to body parts unrelated to the garment (*i.e.* head, hands, etc). In practice, using a 1D CNN on the original SMPL vertex order is more efficient and captures most SMPL neighborhoods, since

91.24% of the SMPL vertices have their adjacent indexed neighbors being adjacent in topology, and 94.01% are at most two-hops away.

6.3.2 GCN-Based Decoder

Following a similar reasoning as in the previous section, I would like to capture local relations between the garment vertex neighborhoods. Therefore, I use a graph convolutional network in the decoder. A common graph convolutional layer is defined as:

$$\mathbf{y} = \mathbf{f}_\theta(\mathbf{A}\mathbf{x}) \tag{6.1}$$

where $\mathbf{A}$ is the aggregation matrix which collects and processes the information in the neighborhood in an isotropic way, and $\mathbf{f}_\theta$ is the nonlinear activation function for feature extraction. The expressiveness of this network is inherently limited since the constant aggregation matrix cannot adapt its neighbor aggregation weights. Attention-based GCN [201] addresses this issue by proposing an MLP to estimate the aggregation parameters given the vertex features:

$$\mathbf{y} = \mathbf{f}_{\theta_1}(\mathbf{A}_{\theta_2}\mathbf{x}) \tag{6.2}$$

$$\mathbf{A}_{\theta_2}[i,j] = MLP(\mathbf{x}_i, \mathbf{x}_j) \tag{6.3}$$

In contrast, I propose to learn the aggregation parameters independently per vertex, without an explicit dependence on the features:

$$\mathbf{A}_{\theta_2}[i,j] = \theta_2[i,j] \tag{6.4}$$

Another particularity of my decoder is its hierarchical nature. Analogous to up-scaling in 2D image generation, feeding the encoded features to a coarsened mesh helps distributing global features to broader regions of the garment. To linearly upsample the features, I use the barycentric weights of the corresponding higher resolution vertices with respect to the lower resolution ones, all in UV space. I use four resolutions in the decoder for all experiments.

6.3.3 Spectral Domain Decomposition

Simulation systems are known to be input-sensitive; negligible differences in the input or initialization can result in substantial differences in the outputs, specially in the high frequency domain. Supervision on vertex positions tends to average those multiple possible outcomes, smoothing out its predictions. However, the high frequency content of the garment is critical for garment perception, since it is highly correlated to garment materials and tightness. This motivates the need to inspect the spectral components of the garment mesh [202]. Specifically, I apply the eigen decomposition on the Laplacian operator:

$$\mathbf{L} = \mathbf{U}\mathbf{D}\mathbf{U}^{-1} \tag{6.5}$$

where $\mathbf{U} \in \mathbb{R}^{n*n}$ and $\mathbf{D}$ are the eigenvectors and the diagonal matrix of the eigenvalues. I pick the subset of eigenvectors $\mathbf{V} \in \mathbb{R}^{n*k}$ corresponding to the smallest k eigenvalues. The spectral coefficients of a mesh $\mathbf{c} = \mathbf{V}^{\top}\mathbf{x}$ thereby represent the mesh components with lowest impact on Laplacian values. This method rejects the highest frequencies (typically noisy) since high frequency entails large local changes, which have a large impact in the Laplacian (and therefore large eigenvalues). The visualization of the eigenvectors is shown in Fig. 6.3.

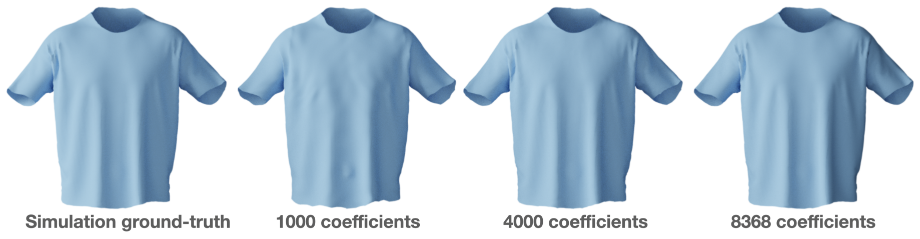

Figure 6.4: Reconstructions for different numbers of coefficients.

This spectral decomposition is used by introducing an MLP-based branch in the decoder network to account for residuals of the spectral components. I output the coefficients $\Delta\mathbf{c}$ of the 4,000 eigenvectors with the smallest eigenvalues, which are sufficient for reconstruction (see Fig. 6.4). These coefficients are then transferred back to the spatial domain $\Delta\mathbf{x} = \mathbf{V}\Delta\mathbf{c}$ and added to the final result.

I also introduce a spectral loss during training. This loss ensures that high frequency components, which typically result in small vertex displacements, deserve proper attention in the supervision of my model.

6.3.4 Loss Functions

I design a number of loss functions to supervise and guide the model towards realistic outputs. First of all, I split the output into a correspondence point set and a displacement map:

$$\mathbf{x} = \mathbf{c}_x + \mathbf{d}_x \quad \mathbf{y} = \mathbf{c}_y + \mathbf{d}_y \tag{6.6}$$

where $\mathbf{c}$ are the correspondence (closest) points on the body surface and $\mathbf{d}$ are the displacements w.r.t. the correspondence, and $*_x$ represents the prediction while $*_y$ represents the ground-truth. This partition of garment into body plus displacement is common in the literature [11, 196], but here it enables an important direction loss that prevents intersections and preserves normals:

$$\mathcal{L}_{dir} = R(-\frac{\mathbf{n}^\top(\mathbf{c}_x - \mathbf{c}_y)}{\|\mathbf{c}_x - \mathbf{c}_y\|}) + (1 - \frac{(\mathbf{x} - \mathbf{c}_y)^\top \mathbf{d}_y}{\|\mathbf{x} - \mathbf{c}_y\| \|\mathbf{d}_y\|}) \tag{6.7}$$

where R denotes relu, and $\mathbf{n}$ is the normal direction at $\mathbf{c}_y$. The first part of the direction loss constrains the correspondence to be outside of the body, while the second part constrains the direction of the prediction to be similar to the ground-truth. Since $\mathbf{c}_y$ is defined as the closest point on the body surface to the garment vertex $\mathbf{x}$, minimizing the direction loss can help generate results with fewer intersections and better normal consistency.

I also use per-vertex L1 difference of these two components separately to supervise the overall shape:

$$\mathcal{L}_{V2V} = \|\mathbf{c}_x - \mathbf{c}_y\|_1 + \|\mathbf{d}_x - \mathbf{d}_y\|_1 \tag{6.8}$$

Additionally, I applied physics-based losses to learn the correct deformation of the garment. The key idea here is to transfer the physics constraints applied in simulation to geometric differences. I choose two aspects that reflect the physics: edge lengths and deformation gradients. First, the edge loss measures the difference of the edge lengths relative to the ground-truth:

$$\mathcal{L}_e = \frac{1}{|E|} \sum_{(\mathbf{u},\mathbf{v})\in E} \frac{|\|\mathbf{u}_x - \mathbf{v}_x\| - \|\mathbf{u}_y - \mathbf{v}_y\||}{\|\mathbf{u}_y - \mathbf{v}_y\|} \tag{6.9}$$

where $*_x$ are the predictions and $*_y$ are the ground-truths, and E is the edge set of the mesh. This loss guides the model to generate more wrinkles because smoothed results often have smaller overall circumference (thereby larger $\mathcal{L}_{edge}$) than ground-truths. Additionally, I define a loss that supervises on the difference of the deformation gradient of each face in the mesh:

$$\mathcal{L}_d = \sum_{\mathbf{f}\in\mathcal{M}} \|\mathbf{F}_x(\mathbf{f}) - \mathbf{F}_y(\mathbf{f})\|_1 \tag{6.10}$$

$$\mathbf{F}_x(\mathbf{f}) = \mathbf{x}(\mathbf{f})\mathbf{X}^{-1}(\mathbf{f}) \tag{6.11}$$

where $\mathbf{F}(\mathbf{f})$ is the deformation gradient [203] of face $\mathbf{f}$ in the mesh $\mathcal{M}$, defined as the change of the world-space coordinates ($\mathbf{x}$) per unit change of the material-space coordinates ($\mathbf{X}$) within each triangle. This loss is less intuitive than the edge loss, but provides better alignment to the ground-truth regarding the potential energy and the internal forces it generates. For example, the deformation gradient can represent the shear stress and bulk stress separately, while the edge-based loss cannot.

To capture the curvature and higher frequency errors, I further define a Laplacian Difference loss, and a spectral loss as described in Sec. 6.3.3:

$$\mathcal{L}_l = \sum_{k=0}^{3} \|\mathbf{L}_k(\mathbf{x} - \mathbf{y})\|_1 \tag{6.12}$$

$$\mathcal{L}_s = \|\mathbf{V}^\top(\mathbf{x} - \mathbf{y})\|_1 \tag{6.13}$$

where $\mathbf{L}_k$ is the Laplacian operator on the mesh graph at the k-th resolution, and $\mathbf{V}$ are the eigenvectors of the Laplacian operator on the original mesh defined in Sec. 6.3.3. I apply the Laplacian difference loss in different resolutions to account for wrinkles and folds of different sizes.

The total loss is the sum of all losses defined above:

$$\mathcal{L} = \mathcal{L}_{V2V} + \mathcal{L}_{dir} + \mathcal{L}_e + \mathcal{L}_d + \mathcal{L}_l + \mathcal{L}_s \tag{6.14}$$

6.4 Physics-Enforced Optimization

During inference on unseen input bodies, it is likely that the drape prediction does not reach a stable dynamical state because the corresponding potential energy may not be fully minimized in all cases. More importantly, it is not uncommon in practical try-on applications that the garment is composed of materials different from the ones used in training. For example, a frontal graphic print of a T-shirt is usually stiffer and heavier than the rest of T-shirt. In theory such situations could be solved by training with the appropriate data, but this solution turns impractical when I consider not only a large variety of graphics print shapes for garments, but further more the infinite variety of them in services that allow you to create your own print.

To address the problems above, I propose to optimize the inferred models on specific samples at runtime. I finetune the network weights for each sample to minimize the potential loss of the garment defined below:

$$\mathcal{L}_p = \mathcal{L}_g + \mathcal{L}_{st} + \mathcal{L}_b \tag{6.15}$$

$$\mathcal{L}_g = \sum_{\mathbf{v} \in \mathcal{M}} m(\mathbf{v}) \mathbf{g}^\top \mathbf{x}(\mathbf{v}) \tag{6.16}$$

$$\mathcal{L}_{st} = \sum_{\mathbf{f} \in \mathcal{M}} \mathbf{S}(\mathbf{f}) \quad \mathcal{L}_b = \sum_{\mathbf{e} \in \mathcal{M}} \mathbf{B}(\mathbf{e}) \tag{6.17}$$

where $\mathcal{L}_p$, $\mathcal{L}_g$, $\mathcal{L}_{st}$, and $\mathcal{L}_b$ are the potential loss functions and its components: gravity, stretching, and bending energy, respectively. $\mathcal{M}$ is the predicted mesh, $m(\mathbf{v})$ and $\mathbf{x}(\mathbf{v})$ is the mass and coordinates of vertex $\mathbf{v}$, $\mathbf{S}(\mathbf{f})$ is the stretching energy of face $\mathbf{f}$, and $\mathbf{B}(\mathbf{e})$ is the bending energy of two adjacent faces with common edge $\mathbf{e}$. I follow the definitions of the stretching and bending energy from the simulator I used [74]. In short, material stiffness coefficients are multiplied to elements in the Green Strain of $\mathbf{f}$ and the curvature of $\mathbf{e}$, respectively. To make the optimization collision-aware, I introduce a penetration loss function [8].

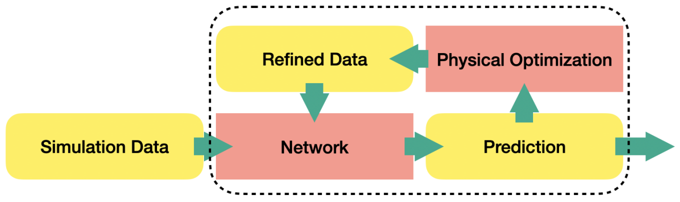

Figure 6.5: The semi-supervised self-correcting training pipeline.

The proposed optimization step serves two purposes in my pipeline. First, it can be used to generate better training data. I empirically show in Table 6.5 that my optimization generates garments with lower potential than the simulated data, even when both are initialized with my network. My hypothesis is that the lack of hard penetration constraints and kinematic limits helps my optimization find a more stable state. Re-training the network with this data can be regarded as self-correcting unsupervised learning since the training data has been produced by the previous network, turning the entire pipeline into a semi-supervised learning framework.

Second, it can be adapted to materials that are not covered by the original model, which covers homogeneous T-shirts. The optimization allows us to predict drapes where the T-shirts contain graphic prints with different shapes and materials (Fig. 6.1). To achieve this, I simply minimize the potential loss of the new system containing the new graphic print. Results in Sec. 6.5 show that the optimization is indeed effective in performing these two tasks.

6.5 Experiments

I conduct several experiments to evaluate my method, including ablation studies testing several different configuration of my system, accuracy tests of the physics optimization, comparisons with previous work, and generalization over multiple T-shirt sizes.

6.5.1 Data Generation

I use the SMPL [30] body model to generate my input dataset. Shape parameters are sampled following a uniform distribution in BMI space with range from 10 to 65 (see Fig. 6.6). In terms of pose, I restrict myself to a fixed A-pose to concentrate on a wide shape variation.

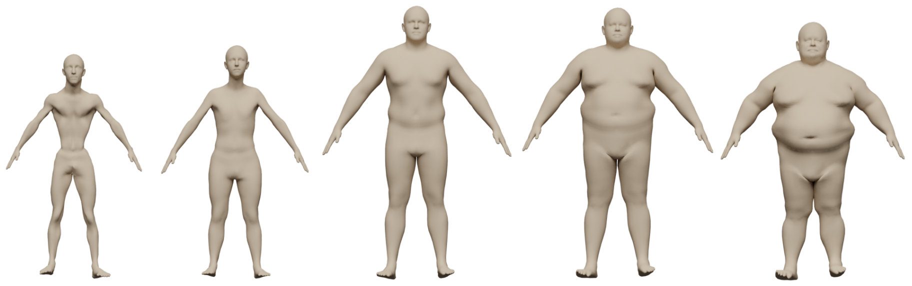

Figure 6.6: Bodies at BMI percentiles 10, 30, 50, 70 and 90%.

I use the T-shirt from TailorNet [11] as my garment template, after retopologizing it to generate a proper UV template for simulation. To generate the ground-truth garments, I initialize the garment to a precomputed configuration coherent with the T-pose average SMPL body. ArcSim [74] is used to simulate the garment movements while the body is linearly morphed into the target shape with A-pose.

The body morphing lasts for two simulation seconds in total, and ArcSim continues to simulate the garment dynamics on the body for two more seconds to obtain stable results. In total, I generate 20,000 such human bodies and garments, where the training/validation/test split is 8:1:1 (for this reason the tables are annotated as *16k* for the full dataset of 20000 samples).

6.5.2 Ablation Study

In this section I evaluate different components of my method in terms of the loss functions defined in Sec. 6.3.4 except from $\mathcal{L}_{V2V}$, which is replaced by the more intuitive mean euclidean distance (ME). I also track the ratio of all garment vertices which are intersected with the human body, p. In the tests in this section, all system components are held constant except for the one being tested.

Encoder. In this experiment my 1D CNN encoder is replaced with other approaches like PointNet [99], the original GCN [204] and my modified GCN (see Sec. 6.3.2), where I replaced upsampling for downsampling operations. Table 6.1 shows that my CNN encoder outperforms other approaches. GCN performs worst because a large part of its capacity is used in feature extraction on body regions of the body irrelevant for my task (*e.g.* hands and feet).

Encoder Type	ME (cm)	$\mathcal{L}_l$ (cm)	$\mathcal{L}_e$ (%)	$\mathcal{L}_s$	$\mathcal{L}_d$	p (%)
PointNet [99]	0.35	**0.16**	9.36	1.1e-3	4.6e-3	**0.05**
GCN [204]	0.45	0.18	10.32	1.2e-3	5.1e-3	0.1
GCN (mine)	0.36	**0.16**	9.54	1.1e-3	4.7e-3	**0.05**
CNN (mine)	**0.33**	**0.16**	**9.21**	**1.0e-3**	**4.5e-3**	**0.05**

Table 6.1: Encoders ablation. ME stands for Mean Euclidean.

Decoder. For comparison, I replaced my multi-scale GCN decoder (see Sec. 6.3.2) with a 2D CNN decoder that represents the garment in UV space [124], an MLP, and the original GCN [204]. My final model achieves the best results in most of the metrics, especially when adding the spectral residual connection. Although the 2D CNN decoder have slightly smaller intersection rates than mine, its large errors in terms of edge lengths and face deformations indicates that the model fails to learn the dynamics of the garment.

Decoder Type	ME (cm)	$\mathcal{L}_l$ (cm)	$\mathcal{L}_e$ (%)	$\mathcal{L}_s$	$\mathcal{L}_d$	p (%)
2D CNN	0.54	0.49	39.45	2.0e-3	22.2e-3	**0.04**
MLP	1.62	0.25	15.71	2.0e-3	7.5e-3	6.96
1D CNN	5.02	0.38	24.51	3.2e-3	11.6e-3	24.67
GCN (original)	1.92	0.41	37.01	3.5e-3	19.8e-3	2.56
GCN (mine)	0.34	**0.16**	9.36	1.1e-3	4.6e-3	0.06
GCN (mine) + spec	**0.33**	**0.16**	**9.21**	**1.0e-3**	**4.5e-3**	0.05

Table 6.2: Decoders ablation. ME stands for Mean Euclidean.

Loss functions. I verify the effectiveness of my loss functions by comparing the test metrics of the learned model when disabling the loss functions one at a time. The main baseline is the per-vertex L2 loss commonly used in previous works [11, 180, 181]. As shown in Table 6.3, all my designed loss functions contribute to smaller overall errors. It can be observed that the intersection ratio is drastically reduced simply by replacing the baseline L2 loss to $\mathcal{L}_{V2V}$ that trains the correspondence and displacement separately (Eqn. 6.8). Introducing the direction loss $\mathcal{L}_{dir}$ further reduces the intersection close to their minimum. The losses related to local shape ($\mathcal{L}_p$), deformations ($\mathcal{L}_e$ and $\mathcal{L}_d$), and spectral energy ($\mathcal{L}_s$) are reduced by introducing the rest of the losses, with negligible negative effects in other metrics.

Loss Functions	ME (cm)	$\mathcal{L}_l$ (cm)	$\mathcal{L}_e$ (%)	$\mathcal{L}_s$	$\mathcal{L}_d$	p (%)
$\mathcal{L}_{base} = L2$	0.4	0.21	12.35	1.2e-3	6.3e-3	2.05
$\mathcal{L}_0 = \mathcal{L}_{V2V}$	**0.33**	0.21	12.93	1.2e-3	6.7e-3	0.41
$\mathcal{L}_1 = \mathcal{L}_0 + \mathcal{L}_{dir}$	0.37	0.23	13.93	1.3e-3	7.2e-3	0.04
$\mathcal{L}_2 = \mathcal{L}_1 + \mathcal{L}_l$	0.36	0.18	11.39	1.2e-3	5.7e-3	**0.03**
$\mathcal{L}_3 = \mathcal{L}_2 + \mathcal{L}_e$	0.35	**0.16**	9.55	1.1e-3	4.7e-3	0.04
$\mathcal{L}_4 = \mathcal{L}_3 + \mathcal{L}_s$	0.34	**0.16**	9.36	1.1e-3	4.6e-3	0.04
$\mathcal{L}_5 = \mathcal{L}_4 + \mathcal{L}_d$ (mine)	**0.33**	**0.16**	**9.21**	**1.0e-3**	**4.5e-3**	0.05

Table 6.3: Losses ablation. ME stands for Mean Euclidean.

6.5.3 Optimization for Semi-Supervision

This section show how my optimization method described in Section 6.4 not only adapts to novel runtime conditions, but also allows us to train a system which achieves high accuracy and fast inference with less data. To validate this, I train my model with 25% amount of the original simulated training data (4,000 samples), and re-generate a full dataset (16,000 samples) where each sample is computed by a 100-iteration optimization from the model prediction. Keeping body input and train/validation/test split exactly the same, I train an additional model based on the new dataset and compare the performance to the model that is trained on the original one.

The test results are shown in Table 6.4. The initial models are denoted as *4k* and *16k* regarding different training data sizes, optimized results are marked with *-opt*, and *4k-retrain* is my retrained model on the optimized data. I evaluate the systems in terms of realism, training and inference time. Realism is evaluated in terms of percentage of intersections and the potential defined in Eqn. 6.15. I use this potential since the original physically simulated garments are not physically optimal and therefore the comparison in terms of per-vertex distance is misleading. The end-to-end training time denotes the total time needed to generate the first sample. For example, for *4k-retrain* it is the sum of the times from simulation (2,666 hours), *4k* training (3 hours), the optimization of all samples (31 hours), and final retraining (10 hours).

Training with more data results in higher accuracy but longer training times. After optimization, the total potential decreases and remaining intersections are resolved. Although optimization is faster than the simulator, it is prohibitively slow for real time applications. By using the optimized samples for the retrained network, I achieve real-time performance with a fairly competitive physical accuracy. Note that by retraining my system with only 4,000 samples (*4k-retrain*), my end-to-end training time is shorter than that of the original system trained on 16,000 samples.

Models	$\mathcal{L}_p$	p (%)	End-to-end training time (h)	Inference time
Simulation	1.0e-4	0.00	0	40min
4k	8.8e-5	0.1	2,669	17 ms
16k	7.5e-5	0.05	10,677	
4k-opt	4.3e-5	0.00	2,669	7 s
16k-opt	4.1e-5	0.00	10,677	
4k-retrain	5.6e-5	0.04	2,710	17 ms

Table 6.4: Self-correcting pipeline ablation study. Training with more data (*16k* vs. *4k*) results in higher accuracy but longer end-to-end training time due to data generation. My optimization (*-opt*) improves the performance but has long inference time. Re-training on the optimized data (*4k-retrain*) achieves small test errors, short training time and real-time inference.

6.5.4 Optimization for Graphic Print

The optimization method enables the draping adaptation to unseen material designs. In this experiment, I generate two datasets, one with a stiffer and denser graphic print on the T-shirt front and one without. I train two models from the datasets separately as the baselines, and also run the optimization process on the model trained with blank T-shirts to adapt to the new print. In this experiment I consider the potential $\mathcal{L}_p$ according to the materials of the T-shirt with a frontal graphic print for both optimization and evaluation. I run this process over all the bodies in my test set.

Table 6.5 shows that training with the correct offline data helps reduce minimally the potential, but the optimization achieves a much stronger error reduction. A qualitative example is also shown in Fig. 6.7, where it can be observed that the frontal print forms different wrinkles in the simulation, which are correctly captured in the optimized result. This process enables model generalization at runtime without new data generation, which is crutial if the amount of material configurations is very largely or simply unknown at training time. Note that simulation initialized from the model prediction is not only slower, but also less accurate than my optimized results.

Models	$\mathcal{L}_{st}$	$\mathcal{L}_b$	$\mathcal{L}_g$	$\mathcal{L}_p$	Runtime	$p(\%)$
w/o Print	7.6e-5	4.5e-7	-9.2e-7	7.5e-5	17ms	0.05
w/ Print	7.5e-5	4.4e-7	-9.2e-7	7.4e-5	17ms	0.05
Optimized	4.2e-5	5.1e-7	-7.8e-7	4.1e-5	7s	0.00
Simulated	1.7e-4	6.2e-7	-9.6e-7	1.7e-4	40min	0.00

Table 6.5: Adaptation to new materials. The optimized results is better than the network estimations and the simulation initialized with the model predictions with print, while still being 342x faster.

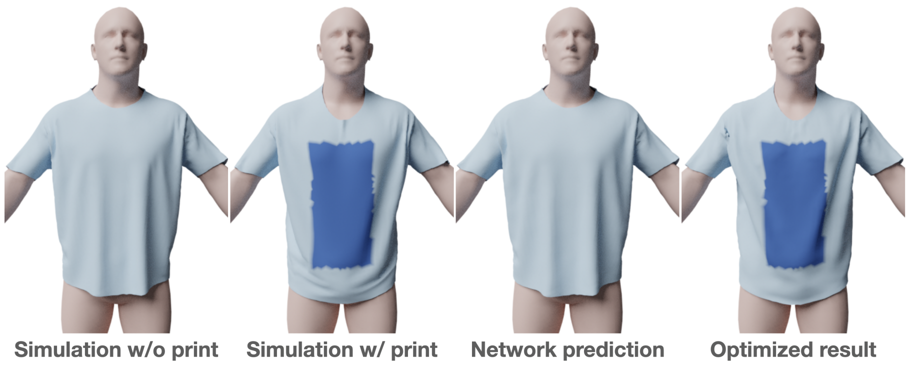

Figure 6.7: Qualitative examples of self-correcting optimization. My method can adapt the model to unseen materials.

6.5.5 Quantitative Comparisons

I compare quantitatively my system (*16k* in Table 6.4) with the system most closely related to mine, TailorNet [11]. Unfortunately I cannot compare to other relevant systems like [180, 181] since they do not provide publicly available code or data; however, a qualitative comparison is provided in Sec. 6.5.6 for completeness. As discussed in Sec. 6.2, remaining works do not fall into the same category as mine, so I do not include them for comparison.

I compare against Tailornet retrained on my own data. Note that for my new dataset, each of the anchor shapes contains a single pose, so learned weighted sum of anchor-based high frequency prediction becomes a learned weighted sum of fixed garments (Table 6.6 *w/ anchor*). To provide more capacity to the high frequency network, I propose an alternative which trains a single high frequency network without anchors (Table 6.6 *w/o anchor*). The original Tailornet could not be used for quantitative comparison since I had to retopologize their template to create a UV map, resulting in a different vertex count. As shown in Table 6.6, my method outperforms TailorNet versions in all test metrics, reducing the errors by

44% ($\mathcal{L}_l$) to 98% (p). I believe the decoder and the loss from TailorNet are the causes of this gap. First, my GCN decoder with learned aggregation outperforms their MLP decoder. Second, I have shown in Sec. 6.5.2 that the per-vertex distance loss from TailorNet is outperformed by my physics-inspired loss.

Methods	ME (cm)	$\mathcal{L}_l$ (cm)	$\mathcal{L}_e$ (%)	$\mathcal{L}_s$	$\mathcal{L}_d$	p (%)
[11] w/ anchor	1.4	0.4	26.59	2.5e-3	14.1e-3	4.89
[11]-post	1.4	0.4	26.4	2.5e-3	14.0e-3	0.03
[11] w/o anchor	1.36	0.29	17.65	2.1e-3	8.8e-3	3.1
Mine	**0.33**	**0.16**	**9.21**	**1.0e-3**	**4.5e-3**	**0.05**
Improvement	75%	44%	47%	52%	48%	98%

Table 6.6: Comparison with TailorNet retrained with my dataset with high frequency discrete anchors (*w/ anchor*), after applying its post-processing (*-post*), and without them (*w/o anchor*). My method reduces the error metrics by 44% to 98%.

6.5.6 Qualitative Results

I compare my model (*16k* in Table 6.4) qualitatively with previous works [11, 180, 181]. A comparison against Tailornet trained on my dataset is shown in Fig. 6.8. In a few examples where the garment roughly fits the body (Column 1, 2), TailorNet successfully predicts good results without too many intersections. However, my prediction results are still visually better due to realization of steeper wrinkles. In other cases, TailorNet fails to provide collision-free results, and the garments are unrealistically crumpled. In contrast, my method provides realistic folds and wrinkles of the garment that follow the dynamics with the body shape.

I also tried my best to compare my results with other two works where code is not available [180, 181]. I took their original figures and generated my drapes for similar body shape. Based on the results in Fig 6.9, I hypothesize that their

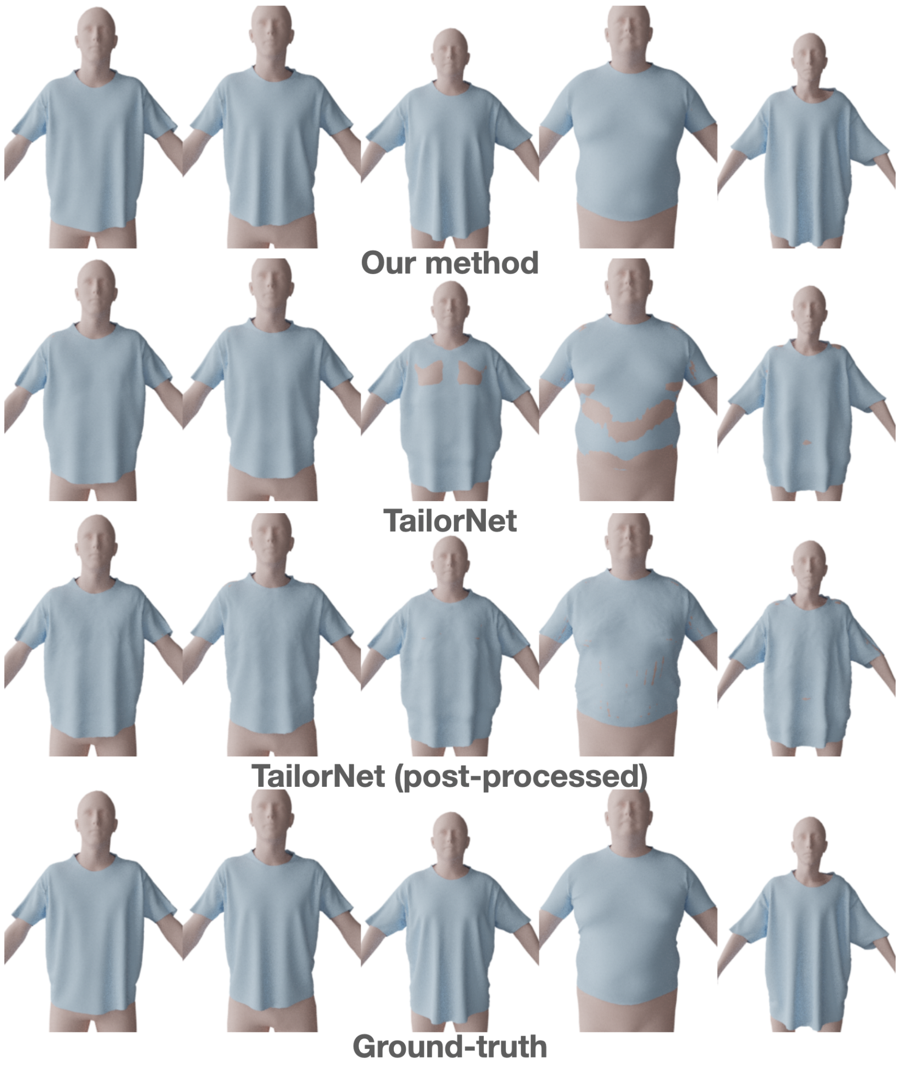

Figure 6.8: Qualitative comparison with TailorNet. My model captures better garment shapes regarding global fold structures, normal consistency, fine wrinkle details, and collision avoidance.

per-vertex loss generates smoother garments with less rich wrinkles. However, I should note that there are substantial differences in terms of T-shirt size, topology and materials, so this comparison is far from definitive.

6.5.7 Generalization to Different Garment Sizes

This section shows the potential of my model to generalize to different sizes of the garment. I simulate a smaller T-shirt of the same topology on the original

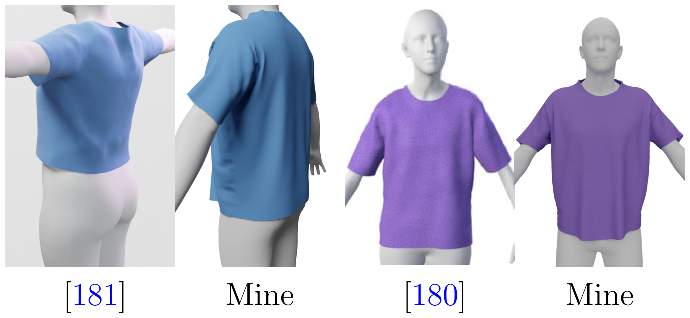

Figure 6.9: Qualitative comparison with [180, 181]. My model generates finer wrinkles around the waist and armpits.

20,000 body samples. I train three different models based on 16,000 small T-shirts, 16,000 original T-shirts, and 32,000 T-shirt of both sizes, and tested them either against the small or original size set, both of them with 2,000 samples. When the network is trained on all the data, the T-shirt size is indicated to the network by simply introducing a flag to the encoded features without any further modifications. As shown in Table 6.7, the model trained on both sizes preserves similar accuracy to the ones trained separately, only slightly worse due to the higher complexity of the task. To improve the results, one possible solution is to train an auto-encoder for a set of garment templates and inject the latent code as the feature. I leave this exploration as future work.

Train-Test Set	ME (cm)	$\mathcal{L}_l$ (cm)	$\mathcal{L}_e$ (%)	$\mathcal{L}_s$	$\mathcal{L}_d$	p (%)
Small-Small	0.34	0.16	9.26	1.1e-3	4.5e-3	0.09
Both-Small	0.56	0.19	11.28	1.3e-3	5.5e-3	0.41
Original-Original	0.33	0.16	9.21	1.0e-3	4.5e-3	0.05
Both-Original	0.34	0.16	9.43	1.1e-3	4.6e-3	0.05

Table 6.7: Comparison for models trained on different sizes of the garments; *A*-*B* means train on *A* and tested on *B*. The model trained on both sizes achieves comparable results.

6.6 Conclusion

I propose a novel network structure and semi-supervised framework to learn realistic fit-accurate garment draping on a wide range of human body sizes. I utilize physics-enforced loss functions and a more expressive GCN decoder during the initial training. Next, a self-correcting optimization method that minimizes the potential energy is proposed to fix the remaining violation of physical laws. Retraining my original network with the optimized samples makes its predictions even more physically accurate. Experimental results show that my method outperforms the state of the art regarding physical realism, dynamics-dependent wrinkle formation, and collision avoidance.

One limitation of this work is that my model currently only supports a canonical body pose. In future work I am planning to support multiple poses at test time by simulating my data under multiple poses and using the unposing technique from [11].