DrapeNet: Garment Generation and Self-Supervised Draping

Luca De Luigi[*,2]   Ren Li[*,1]   Benoît Guillard[1]   Mathieu Salzmann[1]   Pascal Fua[1]

[1]: CVLab, EPFL, {name.surname}@epfl.ch [2]: University of Bologna, luca.deluigi4@unibo.it
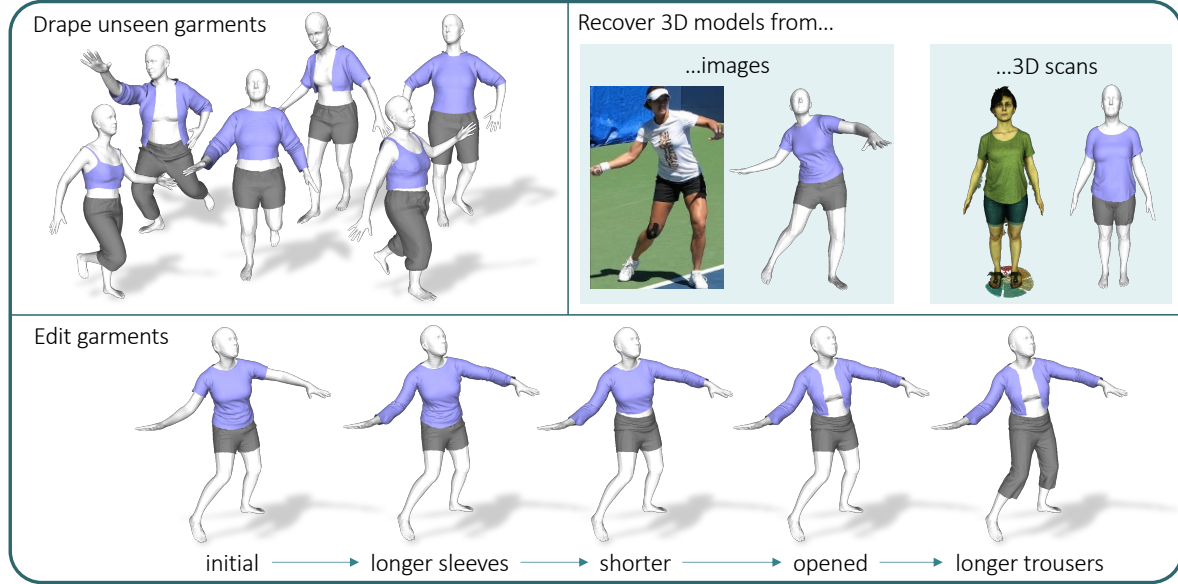
Figure 1. Our network can drape garments over bodies of different shapes in various poses. To minimize the required amount of supervision, our draping network is trained with physics-based self-supervision and generalizes to multiple garments by being conditioned on latent codes. These can be manipulated to edit specific features of the corresponding garments. Being fully differentiable, our pipeline makes it possible to recover 3D models of garments and bodies from observations such as images and 3D scans.

Abstract

Recent approaches to drape garments quickly over arbitrary human bodies leverage self-supervision to eliminate the need for large training sets. However, they are designed to train one network per clothing item, which severely limits their generalization abilities. In our work, we rely on self-supervision to train a single network to drape multiple garments. This is achieved by predicting a 3D deformation field conditioned on the latent codes of a generative network, which models garments as unsigned distance fields.

Our pipeline can generate and drape previously unseen garments of any topology, whose shape can be edited by manipulating their latent codes. Being fully differentiable, our formulation makes it possible to recover accurate 3D models of garments from partial observations – images or 3D scans – via gradient descent. Our code is publicly available at https://github.com/liren2515/DrapeNet.

1. Introduction

Draping digital garments over differently-shaped bodies in random poses has been extensively studied due to its many applications such as fashion design, moviemaking, video gaming, virtual try-on and, nowadays, virtual and augmented reality. Physics-based simulation (PBS) [3, 12, 26, 36, 37, 43, 44, 52–54, 57, 66] can produce outstanding results, but at a high computational cost.

Recent years have witnessed the emergence of deep neural networks aiming to achieve the quality of PBS draping while being much faster, easily differentiable, and offering new speed vs. accuracy tradeoffs [16, 17, 28, 40, 49, 51, 56, 59, 61]. These networks are often trained to produce garments that resemble ground-truth ones. While effective, this requires building training datasets, consisting of ground-truth meshes obtained either from computationally expensive simulations [35] or using complex 3D scanning setups [42]. Moreover, to generalize to unseen garments and poses, these supervised approaches require training databases encompassing a great variety of samples de-

---

[*] Equal contributions

1

picting many combinations of garments, bodies and poses.

The recent PBNS and SNUG approaches [5, 47] address this by casting the physical models adopted in PBS into constraints used for self-supervision of deep learning models. This makes it possible to train the network on a multitude of body shapes and poses without ground-truth draped garments. Instead, the predicted garments are constrained to obey physics-based rules. However, both PBNS and SNUG, require training a separate network for each garment. They rely on mesh templates for garment representation and feature one output per mesh vertex. Thus, they cannot handle meshes with different topologies, even for the same garment. This makes them very specialized and limits their applicability to large garment collections as a new network must be trained for each new clothing item.

In this work, we introduce `DrapeNet`, an approach that also relies on physics-based constraints to provide self-supervision but can handle generic garments by conditioning a *single* draping network with a latent code describing the garment to be draped. We achieve this by coupling the draping network with a garment *generative* network, composed of an encoder and a decoder. The encoder is trained to compress input garments into compact latent codes that are used as input condition for the draping network. The decoder, instead, reconstructs a 3D garment model from its latent code, thus allowing us to sample and edit new garments from the learned latent space.

Specifically, we model the output of the garment decoder as an unsigned distance function (UDF), which were demonstrated [15] to yield better accuracy and fewer interpenetrations than the inflated signed distance functions often used for this purpose [10, 22]. Moreover, UDFs can be triangulated in a differentiable way [15] to produce explicit surfaces that can easily be post-processed, making our pipeline fully differentiable. Hence, `DrapeNet` can not only drape garments over given body shapes but can also perform gradient-based optimization to fit garments, along with body shapes and poses, to partial observations of clothed people, such as images or 3D scans.

Our contributions are as follows:

- We introduce a *single* garment draping network conditioned on a latent code to handle generic garments from a large collection (e.g. *top* or *bottom* garments);
- By exploiting physics-based self-supervision, our pipeline only requires a few hundred garment meshes in a canonical pose for training;
- Our framework enables the fast draping of new garments with high fidelity, as well as the sampling and editing of new garments from the learned latent space;
- Being fully differentiable, our method can be used to recover accurate 3D models of clothed people from images and 3D scans.

2. Related Work

Implicit Neural Representations for 3D Surfaces. Implicit neural representations have emerged a few years ago as an effective tool to represent surfaces whose topology is not known a priori. They can be implemented using (clipped) *signed distance functions* (SDF) [39] or *occupancies* [31]. When an explicit representation is required, it can be obtained using Marching Cubes [21] and this can be done while preserving differentiability [1, 30, 46]. However, they can only represent watertight surfaces.

Thus, to represent open surfaces, such as clothes, it is possible to use inflated SDFs surrounding them. However, this entails a loss in accuracy and there has been a recent push to replace SDFs by *unsigned distance functions* (UDFs) [9, 58, 67]. One difficulty in so doing was that Marching Cubes was not designed with UDFs in mind, and obtaining explicit surfaces from these UDFs was therefore non-trivial. This has been addressed in [15] by modifying the Marching Cubes algorithm to operate with UDFs. We model garment with UDFs and use [15] to mesh them. Other works augment signed distance fields with covariant fields to encode open surface garments [8, 48].

Draping Garments over 3D Bodies. Two main classes of methods coexist, physics-based algorithms [3, 24, 25, 34, 35, 53] that produce high-quality drapings but at a high computational cost, and data-driven approaches that are faster but often at the cost of realism.

Among the latter, template-based approaches [5, 7, 19, 38, 40, 47, 50, 56] are dominant. Each garment is modeled by a specific triangulated mesh and a draping function is learned for each one. In other words, they do not generalize. There are however a number of exceptions. In [6, 16] the mesh is replaced by 3D point clouds that can represent generic garments. This enables deforming garments with arbitrary topology and geometric complexity, by estimating the deformation separately for each point. [65] goes further and allows differentiable changes in garment topology by sampling a fixed number of points from the body mesh. Unfortunately, this point cloud representation severely limits possible downstream applications.

In recent approaches [10, 22], a space of garments is learned with clothing items modeled as inflated SDFs and one single shared network to predict their deformations as a 3D displacement field. This makes deployment in real-world scenarios easier and allows the reconstruction of garments from images and 3D scans. However, the inflated SDF scheme reduces realism and precludes postprocessing using standard physics-based simulators or other cloth-specific downstream applications. Furthermore, both models are fully supervised and require a dataset of draped garments whose collection is extremely time-consuming.

Alleviating the need for costly ground-truth draped garments is tackled in [5, 47], by introducing physics-based
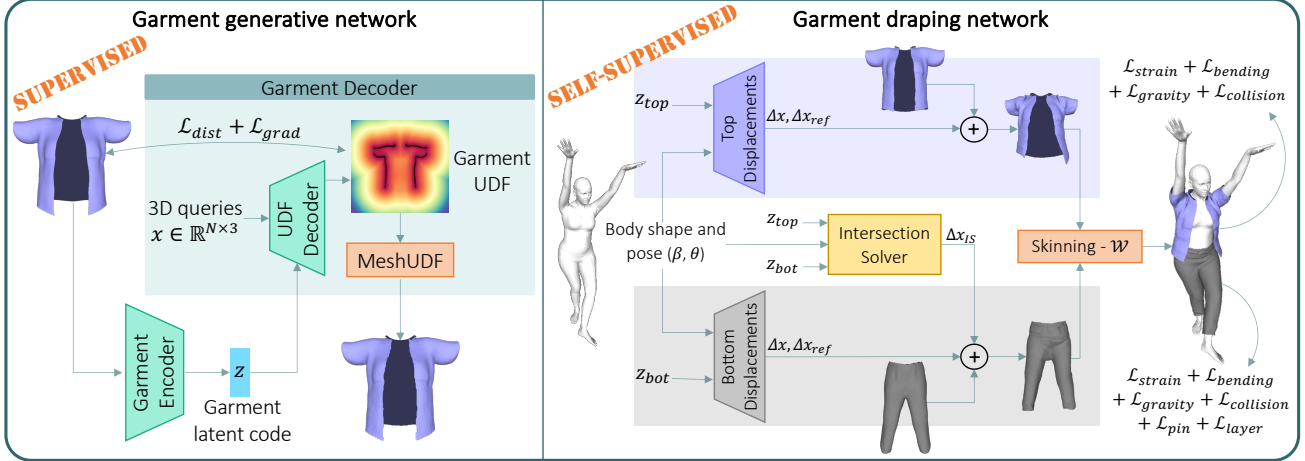
Figure 2. **Overview of our framework. Left:** Garment generative network, trained to embed garments into compact latent codes and predict their unsigned distance field (UDF) from such vectors. UDFs are then meshed using [15]. **Right:** Garment draping network, conditioned on the latent codes of the generative network. It is trained in a self-supervised way to predict the displacements Δx and Δx_{ref} to be applied to the vertices of given garments, before skinning them according to body shape and pose (β, θ) with the predicted blending weights $\mathcal{W}$. It includes an Intersection Solver module to prevent intersection between top and bottom garments.

losses to train draping networks in a self-supervised manner. The approach of [47] relies on a mass spring model to enforce the physical consistency of static garments deformed by different body poses. The method of [5] also accounts for variable body shapes and dynamic effects; furthermore, it incorporates a more realistic and expressive material model. Both methods, however, require training one network per garment, a limitation we remove.

3. Method

We aim to realistically deform and drape generic garments over human bodies of various shapes and poses. To this end, we introduce the `DrapeNet` framework, presented in Fig. 2. It comprises a generative network shown on the left and a draping network shown on the right. Only the first is trained in a supervised manner, but using only static unposed garments meshes. This is key to avoiding having to run physics-based simulations to generate ground-truth data. Furthermore, we condition the draping network on latent vectors representing the input garments, which allows us to use the same network for very different garments, something that competing methods [5, 47] cannot do.

The generative network is a decoder trained using an encoder that turns a garment into a latent code **z** that can then be decoded to an Unsigned Distance Function (UDF), from which a triangulated mesh can be extracted in a differentiable manner [15]. The UDF representation allows us to accurately represent open surfaces and the many openings that garments typically feature. Since the top and bottom garments – shirts and trousers – have different patterns, we train one generative model for each. Both networks have the same architecture but different weights.

The resulting *garment generative network* is only trained to output garments in a canonical shape, pose, and size that fit a neutral SMPL [27] body. Draping the resulting garments to bodies in non-canonical poses is then entrusted to a *draping network*, again one for the top and one for the bottom. As in [5, 22, 47], this network predicts vertex displacements w.r.t. the neutral position. The deformed garment is then skinned onto the articulated body model. To enable generalization to different tops and bottoms, we condition the draping process on the garment latent codes of the generative network, shown as $\mathbf{z}_{\mathrm{top}}$ and $\mathbf{z}_{\mathrm{bot}}$ in Fig. 2.

We use a small database of static unposed garments loosely aligned with bodies in the canonical position to train the two garment generating networks. This being done, we exploit physics-based constraints to train in a fully self-supervised manner the top and bottom draping networks for realism, without interpenetrations with the body and between the garments themselves.

3.1. Garment Generative Network

To encode garments into latent codes that can then be decoded into UDFs, we rely on a point cloud encoder that embeds points sampled from the unposed garment surface into a compact vector. This lets us obtain latent codes for previously unseen garments in a single inference pass from points sampled from its surface. This can be done given any arbitrary surface triangulation. Hence, it gives us the flexibility to operate on any given garment mesh.

We use DGCNN [62] as the encoder. It first propagates the features of points within the same local region at multiple scales and then aggregates them into a single global embedding by max pooling. We pair it with a decoder that takes as input a latent vector, along with a point in 3D space,

and returns its (unsigned) distance to the garment. The decoder is a multi-layer perceptron (MLP) that relies on Conditional Batch Normalization [60] for conditioning on the input latent vector.

We train the encoder and the decoder by encouraging them to jointly predict distances that are small near the training garments' surface and large elsewhere. Because the algorithm we use to compute triangulated meshes from the predicted distances [15] relies on the gradient vectors of the UDF field, we also want these gradients to be as accurate as possible [2,67]. We therefore minimize the loss

$$L_{garm} = L_{dist} + \lambda_g L_{grad} \,, \qquad (1)$$

where L_{dist} encodes our distance requirements, L_{grad} the gradient ones, and λ_g is a weight balancing their influence.

More formally, at training time and given a mini-batch comprising B garments, we sample a fixed number P of points from the surface of each one. For each resulting point cloud $\mathbf{p}_i$ ($1 \leq i \leq B$), we use the garment encoder E_G to compute the latent code

$$\mathbf{z}_i = E_G(\mathbf{p}_i) \qquad (2)$$

and use it as input to the decoder D_G. It predicts an UDF field supervised with Eq. (1), whose terms we define below.

Distance Loss. Having experimented with many different formulations of this loss, we found the following one both simple and effective. Given N points $\{\mathbf{x}_{ij}\}_{j \leq N}$ sampled from the space surrounding the i-th garment, we pick a distance threshold δ, clip all the ground-truth distance values $\{y_{ij}\}$ to it, and linearly normalize the clipped values to the range $[0, 1]$. This yields normalized ground-truth values $\bar{y}_{ij} = \min(y_{ij}, \delta)/\delta$. Similarly, we pass the output of the final layer of D_G through a sigmoid function $\sigma(\cdot)$ to produce a prediction in the same range for point $\mathbf{x}_{ij}$

$$\widetilde{y}_{ij} = \sigma(D_G(\mathbf{x}_{ij}, \mathbf{z}_i)) \,. \qquad (3)$$

Finally, we take the loss to be

$$\mathcal{L}_{dist} = \mathrm{BCE}\left[(\bar{y}_{ij})_{j \leq N}^{i \leq B} \,, \, (\widetilde{y}_{ij})_{j \leq N}^{i \leq B} \right], \qquad (4)$$

where $\mathrm{BCE}[\cdot, \cdot]$ stands for binary cross-entropy. As observed in [13], the sampling strategy used for points $\mathbf{x}_{ij}$ strongly impacts training effectiveness. We describe ours in the supplementary. In our experiments, we set $\delta = 0.1$, being the top and bottom garments normalized respectively into the upper and lower halves of the $[-1, 1]^3$ cube.

Gradient Loss. Given the same sample points as before, we take the gradient loss to be

$$\mathcal{L}_{grad} = \frac{1}{BN} \sum_{i,j} \|\mathbf{g}_{ij} - \widehat{\mathbf{g}_{ij}}\|_2^2 \,, \qquad (5)$$

where $\mathbf{g}_{ij} = \nabla_{\mathbf{x}} y_{ij} \in \mathbb{R}^3$ is the ground-truth gradient of the i-th garment's UDF at $\mathbf{x}_{ij}$ and $\widehat{\mathbf{g}_{ij}} = \nabla_{\mathbf{x}} D_G(\mathbf{x}_{ij}, \mathbf{z}_i)$ the one of the predicted UDF, computed by backpropagation.

3.2. Garment Draping Network

We describe our approach to draping generic garments as opposed to specific ones and our self-supervised scheme. We assume that all garments are made of a single common fabric material, and we drape them in a quasi-static manner.

3.2.1 Draping Generic Garments

We rely on SMPL [27] to parameterize the body in terms of shape (β) and pose (θ) parameters. It uses Linear Blend Skinning to deform a body template. Since garments generally follow the pose of the underlying body, we extend the SMPL skinning procedure to the 3D volume around the body for garment draping. Given a point $\mathbf{x} \in \mathbb{R}^3$ in the garment space, its position $D(\mathbf{x}, \beta, \theta, \mathbf{z})$ after draping becomes

$$D(\mathbf{x}, \beta, \theta, \mathbf{z}) = W(\mathbf{x}_{(\beta, \theta, \mathbf{z})}, \beta, \theta, \mathcal{W}(\mathbf{x})) \,, \qquad (6)$$
$$\mathbf{x}_{(\beta, \theta, \mathbf{z})} = \mathbf{x} + \Delta x(\mathbf{x}, \beta) + \Delta x_{\mathrm{ref}}(\mathbf{x}, \beta, \theta, \mathbf{z}) \,,$$
$$\Delta x_{\mathrm{ref}}(\mathbf{x}, \beta, \theta, \mathbf{z}) = \mathcal{B}(\beta, \theta) \cdot \mathcal{M}(x, \mathbf{z}) \,,$$

where $W(\cdot)$ is the SMPL skinning function, applied with blending weights $\mathcal{W}(\mathbf{x})$, over the point displaced by $\Delta x(\mathbf{x}, \beta)$ and $\Delta x_{\mathrm{ref}}(\mathbf{x}, \beta, \theta, \mathbf{z})$. $\mathcal{W}(\mathbf{x})$ and $\Delta x(\mathbf{x}, \beta)$ are computed as in [22, 50]. However, they only give an initial deformation for garments that roughly fits the underlying body. To refine it, we introduce a new term, $\Delta x_{\mathrm{ref}}(\mathbf{x}, \beta, \theta, \mathbf{z})$. It is a deformation field conditioned on body parameters β and θ, and on the garment latent code $\mathbf{z}$ from the generative network. Following the linear decomposition of displacements in SMPL, it is the composition of an embedding $\mathcal{B}(\beta, \theta) \in \mathbb{R}^{N_{\mathcal{B}}}$ of body parameters and a displacement matrix $\mathcal{M}(x, \mathbf{z}) \in \mathbb{R}^{N_{\mathcal{B}} \times 3}$ conditioned on $\mathbf{z}$. Being conditioned on the latent code $\mathbf{z}$, Δx_{ref} can deform different garments differently, unlike the methods of [5,47]. The number of vertices does not need to be fixed, since displacements are predicted separately for each vertex.

Since we have distinct encodings for the top and bottom garments, for each one we train two MLPs ($\mathcal{B}$, $\mathcal{M}$) to predict Δx_{ref}. The other MLPs for $\mathcal{W}(\cdot)$ and $\Delta x(\cdot)$ are shared.

3.2.2 Self-Supervised Training

We first learn the weights of $\mathcal{W}(\cdot)$ and $\Delta x(\cdot)$ as in [22, 50], which does not require any annotation or simulation data but only the blending weights and shape displacements of SMPL. We then train our deformation fields Δx_{ref} in a fully self-supervised fashion by minimizing the physics-based losses introduced below. In this way, we completely eliminate the huge cost that extensive simulations would entail.

Top Garments. For upper body garments – shirts, t-shirts, vests, tank tops, etc. – the deformation field is trained using the loss from [47], expressed as

$$\mathcal{L}_{top} = \mathcal{L}_{strain} + \mathcal{L}_{bend} + \mathcal{L}_{gravity} + \mathcal{L}_{col} \,, \qquad (7)$$

4

where $\mathcal{L}_{strain}$ is the membrane strain energy of the deformed garment, $\mathcal{L}_{bend}$ the bending energy caused by the folding of adjacent faces, $\mathcal{L}_{gravity}$ the gravitational potential energy, and $\mathcal{L}_{col}$ a penalty for collisions between body and garment. Unlike in [47], we only consider the quasi-static state after draping, that is, without acceleration.

Bottom Garments. Due to gravity, bottom garments, such as trousers, would drop onto the floors if we used only the loss terms of Eq. (7). We thus introduce an extra loss term to constrain the deformation of vertices around the waist and hips. The loss becomes

$$\mathcal{L}_{bottom} = \mathcal{L}_{strain} + \mathcal{L}_{bend} + \mathcal{L}_{gravity} + \mathcal{L}_{col} + \mathcal{L}_{pin},$$

$$\mathcal{L}_{pin} = \sum_{v \in V} |\Delta x_y|^2 + \lambda(|\Delta x_x|^2 + |\Delta x_z|^2) , \quad (8)$$

where V is the set of garment vertices whose closest body vertices are located in the region of the waist and hips. See supplementary material for details. The terms Δx_x, Δx_y and Δx_z are the deformations along the X, Y and Z axes, respectively. λ is a positive value smaller than 1 that penalizes deformations along the vertical direction (Y axis) and produces natural deformations along the other directions.

Top-Bottom Intersection. To ensure that the top and bottom garments do not intersect with each other when we drape them on the same body, we define a loss $\mathcal{L}_{IS}$ that ensures that when the top and the bottom garments overlap, the bottom garment vertices are closer to the body mesh than the top ones, which prevents them from intersecting – this is arbitrary, and the following could be formulated the other way around. To this end, we introduce an Intersection Solver (IS) network. It predicts a displacement correction Δx_{IS}, added only when draping bottom garments as

$$\tilde{\mathbf{x}}_{(\mathbf{z}_{top}, \mathbf{z}_{bot})} = \mathbf{x}_{(\mathbf{z}_{bot})} + \Delta x_{IS}(\mathbf{x}, \mathbf{z}_{top}, \mathbf{z}_{bot}) , \quad (9)$$

where we omit the dependency of $\tilde{\mathbf{x}}$, $\mathbf{x}$ and Δx_{IS} on the body parameters (β, θ) for simplicity. $\mathbf{z}_{top}$ and $\mathbf{z}_{bot}$ are the latent codes of the top and bottom garments, and $\mathbf{x}_{(\mathbf{z}_{bot})}$ is the input point displaced according to Eq. (6). The skinning function of Eq. (6) is then applied to $\tilde{\mathbf{x}}_{(\mathbf{z}_{top}, \mathbf{z}_{bot})}$ for draping. $\Delta x_{IS}(\cdot)$ is implemented as a simple MLP and trained with

$$\mathcal{L}_{IS} = \mathcal{L}_{bottom} + \mathcal{L}_{layer}, \quad (10)$$

where $\mathcal{L}_{layer}$ is a loss whose minimization requires the top and bottom garments to be separated from each other. We formulate it as

$$\mathcal{L}_{layer} = \sum_{v_B \in C} max(d_{bot}(v_B) - \gamma d_{top}(v_B), 0) , \quad (11)$$

where C is the set of body vertices covered by both the top and bottom garments, $d_{top}(\cdot)$ and $d_{bot}(\cdot)$ the distance to the top and the bottom garments respectively, and γ a positive value smaller than 1 (more details in the supplementary).
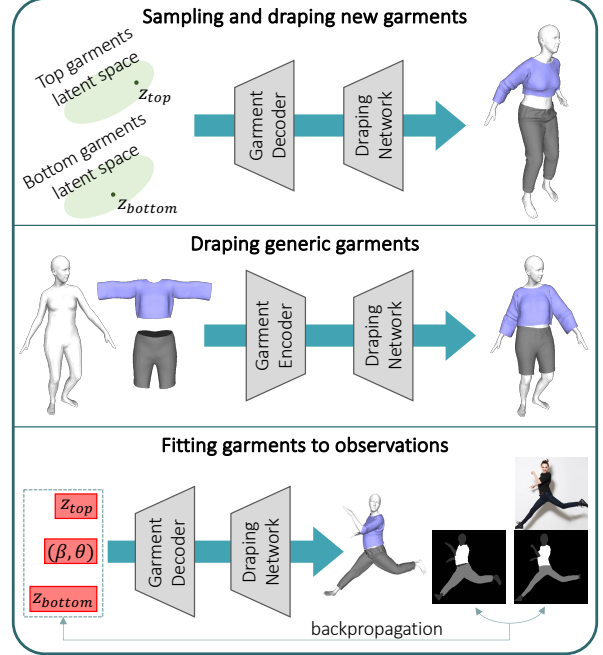

Figure 3. **Overview of** `DrapeNet` **applications. Top:** New garments can be sampled from the latent spaces of the generative networks, and deformed by the draping networks to fit to a given body. **Center:** The garment encoders and the draping networks form a general purpose framework to drape any garment with a single forward pass. **Bottom:** Being a differentiable parametric model, our framework can reconstruct 3D garments by fitting observations such as images or 3D scans. The red boxes indicate the parameters optimized in this process.

4. Experiments

We first describe our experimental setup and test `DrapeNet` for the different purposes depicted by Fig. 3. They include reconstructing different kinds of garments and editing them by manipulating their latent codes. We then gauge the draping network both qualitatively and quantitatively. Finally, we use `DrapeNet` to reconstruct garments from images and 3D scans.

4.1. Settings, Datasets and Metrics

Datasets. Both our generative and draping networks are trained with garments from CLOTH3D [4], a synthetic dataset that contains over 7K sequences of animated 3D humans parametrized used the SMPL model and wearing different garments. Each sequence comprises up to 300 frames and features garments coming from different templates. For training, we randomly selected 600 top garments (t-shirts, shirts, tank tops, etc.) and 300 bottom garments (both long and short trousers). Neither for the generative nor for the draping networks did we use the simulated deformations of the selected garments. Instead, we trained the networks using only garment meshes on average body shapes in T-pose.
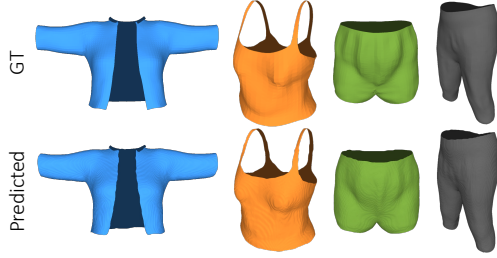
5

Figure 4. **Generative network: reconstruction of unseen garments in neutral pose/shape.** The latent codes are obtained with the garment encoder, then decoded into open surface meshes.

By contrast, for testing purposes, we selected random clothing items – 30 for top garments and 30 bottom ones – and considered *whole* simulated sequences.

Training. We train two different models for top and bottom garments, both for the generative and for the draping parts of our framework. First, the generative models are trained on the 600/300 neutral garments. Then, with the generative networks weights frozen, we train the draping networks by following [47]: body poses θ are sampled randomly from the AMASS [29] dataset, and shapes β uniformly from $[-3, 3]^{10}$ at each step. The other hyperparameters are given in the supplementary material.

Metrics. We report the Euclidean distance (ED), interpenetration ratio between body and garment (B2G), and intersection between top and bottom garments (G2G). ED is computed between corresponding vertices of the considered meshes. B2G is the area ratio between the garment faces inside the body and the whole surface as in [22]. Since CLOTH3D exclusively features pairs of top/bottom garments with the bottom one closer to the body, G2G is computed by detecting faces of the bottom garment that are outside of the top one, and taking the area ratio between those and the overall bottom garment surface.

4.2. Garment Paramerization

We first test the encoding-decoding scheme of Sec. 3.1.

Encoding-Decoding Previously Unseen Garments. The generative network of Fig. 2 is designed to project garments into a latent space and to reconstruct them from the resulting latent vectors. In Fig. 4, we visualize reconstructed previously-unseen garments from CLOTH3D. The reconstructions are faithful to the input garments, including fine-grained details such as the shirt collar on the left or the shoulder straps of the tank top.

Semantic Manipulation of Latent Codes. Our framework enables us to edit a garment by manipulating its latent code. For the resulting edits to have a semantic meaning, we assigned binary labels corresponding to features of interest to 100 training garments. For instance, we labeled garments as having "short sleeves" (label = 0) or "long sleeves" (label = 1). Then, we fit a linear logistic regressor to the garment latent codes. After training, the regressor weights indicate
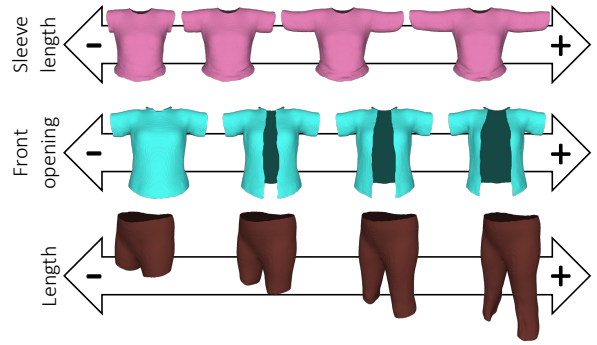

Figure 5. **Garment editing.** The latent codes produced by the garment encoder can be manipulated to edit specific features of the corresponding garments, without altering the overall geometry.

	DeePSD	DIG	Ours
ED-top (mm)	28.1	29.6	47.9
ED-bottom (mm)	18.3	20.0	27.3
B2G-top (%) ↓	7.2	1.8	**0.9**
B2G-bottom (%) ↓	3.4	0.8	**0.3**
G2G (%) ↓	2.0	4.0	**0.5**

Table 1. **Draping unseen garment meshes.** Comparison between DeePSD, DIG and our method, for top and bottom garments: Euclidean distance (ED), intersections with the body (B2G) and between garments (G2G) as ratio of intersection areas.

which dimensions of the latent space control the feature of interest. To this end, we first apply min-max normalization to the absolute weight values and then zero out the ones below a certain threshold, empirically set to 0.5. The remaining non-zero weights indicate which dimensions of the latent codes should be increased or decreased to edit the studied feature. To create Fig. 5, we applied this simple procedure to control the sleeve length and the front opening for top garments along with the length for bottom garments. As can be seen from the figure, our latent representations give us the ability to edit a specific garment feature while leaving other aspects of the garment geometry unchanged.

4.3. Garment Draping

We now turn to the evaluation of the draping network and compare its performance to those of DeePSD [6] or DIG [22], two *fully supervised* learning methods trained on CLOTH3D. DeePSD takes the point cloud of the garment mesh as input and predicts blending weights and pose displacements for each point; DIG drapes garments with a learned skinning field that can be applied to generic 3D points, but is similar for all garments. We chose those because, like `DrapeNet`, they both can deform garments of arbitrary geometry and topology.

Draping Unseen Meshes. We drape previously unseen garments on different bodies in random poses. We first encode the garments and use the resulting latent codes to condition the draping network, whose inference takes ∼5ms.
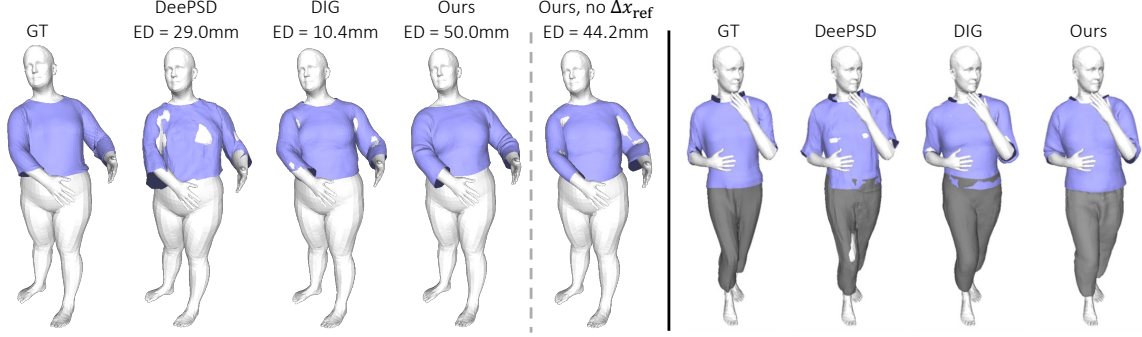
Figure 6. **Comparison between DeePSD, DIG and our method.** Ours is more realistic despite having the highest Euclidean distance (ED) error (**left**), and has less intersection between garments (**right**). **Left** also shows that Δx_{ref} is necessary for realistic deformations.
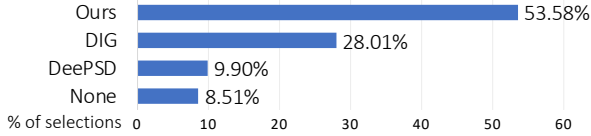

Figure 7. **Human evaluation of draping results.** When shown draping results of our method, DIG and DeePSD, evaluators selected ours as the most realistic one in more than half of the cases. *None* refers to the case when they had no clear preference.

Figure 8. **Switching input latent codes of the draping network.** Draping the same shirt by conditioning the draping network with **(a)** the corresponding latent code, **(b)** the code of an open vest, **(c)** of a t-shirt and **(d)** of a tank top. Gray meshes in dashed boxed are the garments corresponding to the input latent codes.

We provide qualitative results in Fig. 6 and report quantitative ones in Tab. 1. Despite being completely self-supervised, `DrapeNet` delivers the lowest ratio of body-garment interpenetrations (B2G) for both top and bottom garments and the least intersections between them (G2G).

However, `DrapeNet` also yields higher ED values, which makes sense because there is more than one way to satisfy the physical constraints and to achieve realism. Hence, in the absence of explicit supervision, there is no reason for the answer picked by `DrapeNet` to be exactly the same as the one picked by the simulator. In fact, as argued in [5] and illustrated by Fig. 6, which is representative in terms of ED, a low ED value does not necessarily correspond to a realistic draping. To confirm this, we conducted a human evaluation study by sharing a link to a website on friends groupchats. We gave no further instructions or details besides those given on the site and reproduced in the supplementary material. The website displays 3 drapings of the same garment over the same posed body, one computed using our method and the others using the other two. The users were asked to select which one of the three seemed more realistic and more pleasant, with a fourth potential response being "none of them". We obtained feedback from 187 different people. A total of 1258 individual examples were rated and we collected 3738 user opinions. In other words, each user expressed 20 opinions on average. The chart in Fig. 7 shows that our method was selected more than 50% of the times, with a large gap over the second best, DIG [22], selected less than 30% of the time. This result confirms that `DrapeNet` can drape garments with better perceptual quality than the competing methods.
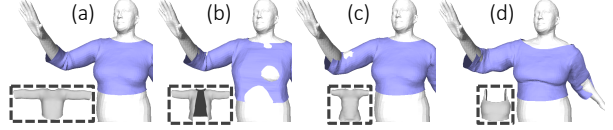
Ablation Study. In Fig. 8, we show what happens when the draping network is conditioned with a latent code of a garment that does not match the input one. This creates unnatural deformations on the front when using the code of a shirt with a front opening to deform a shirt without an opening. Similarly, the sleeves penetrate the arms when conditioning with the code of a short sleeves shirt. This demonstrates that the draping network truly exploits the latent codes to predict garment-dependent deformation fields.

In Fig. 6 **left** we show that removing our novel displacement term $\Delta x_{\mathrm{ref}}(\cdot)$ from Eq. (6) leads to unrealistic results.

We also ablate the influence of our Intersection Solver and observe that G2G increases from 0.5% to 1.1% without it. This demonstrates the effectiveness of this component at reducing collisions between top and bottom garments.

4.4. Fitting Observations

Since our method is end-to-end differentiable, it can be used to reconstruct 3D models of people and their garments from partial observations, such as 2D images and 3D scans.

Fitting Images. Given an image of a clothed person, we use the algorithm of [63, 64] to get initial estimates for the body parameters (β, θ) and a segmentation mask $\mathbf{S}$. Then, starting with the mean of the learned codes $\mathbf{z}$, we reconstruct a mesh for the body and its garments by minimizing

$$L(\beta, \theta, \mathbf{z}) = L_{\mathrm{IoU}}(R(D(\mathbf{G}, \beta, \theta, \mathbf{z}), \mathrm{SMPL}(\beta, \theta)), \mathbf{S}),$$
$$\mathbf{G} = \mathrm{MeshUDF}(D_G(\mathbf{z})), \quad (12)$$

w.r.t. $\mathbf{z}$, β and θ, where L_{IoU} is the IoU loss [23] in pixel space penalizing discrepancies between 2D masks, $R(\cdot)$ is

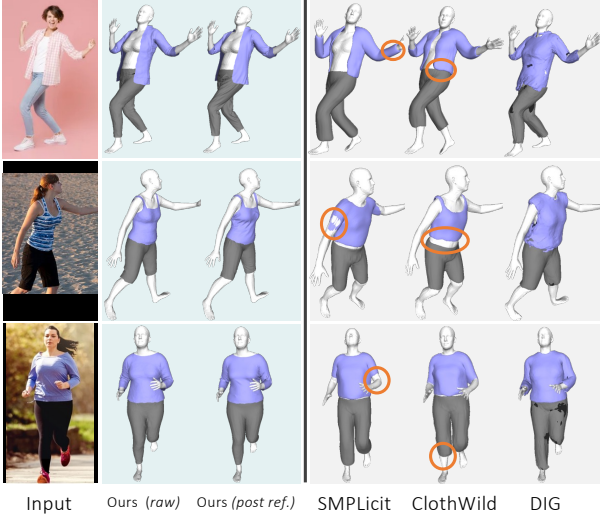| Input | Ours *(raw)* | Ours *(post ref.)* | SMPLicit | ClothWild | DIG |

Figure 9. **Recovering garments and bodies from images.** From left to right we show the input image and the 3D models recovered with our method (without and with post-refinement), and competitors methods: SMPLicit [10], ClothWild [33], DIG [22].
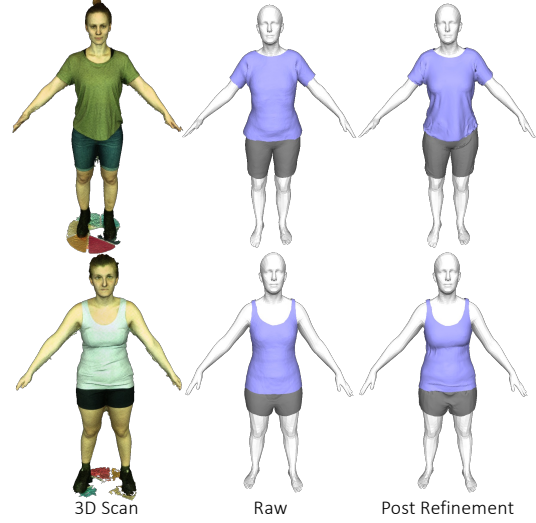

| 3D Scan | Raw | Post Refinement |

Figure 10. **Recovering garments and bodies from 3D scans.** We show 3D models recovered with our method from scans of the SIZER dataset [56]. *Raw* indicates the model recovered with Eq. (13) from the 3D scan. *Post Refinement* refers to the models further refined with the physics-based losses.

a differentiable mesh renderer [45], and $\mathbf{G}$ is the set of vertices of the garment mesh reconstructed with our garment decoder using $\mathbf{z}$. $D(\cdot)$ and $SMPL(\cdot)$ are the garment and body skinning functions defined in Eq. (6) and in [27], respectively. To ensure pose plausibility, θ is constrained by an adversarial pose prior [11].

For the sake of simplicity, Eq. (12) formulates the reconstruction of a single garment $\mathbf{G}$. In practice, we extend this formulation to both the top and the bottom garments shown in the target image. Fig. 9 depicts the results of minimizing this loss. It outperforms the state-of-the-art methods SMPLicit [10], ClothWild [33] and DIG [22]. The garments we recover follow the ones in the input image with higher fidelity and visual quality, without interpenetration between the body and the garments or between the two garments.

After this optimization, we can further refine the result by minimizing the physics-based objectives of Eq. (7) w.r.t. the per-vertex displacements of the reconstructed garments, as opposed to w.r.t. the latent vectors. We describe this procedure in the supplementary material. As shown in the third column of Fig. 9, this further boosts the realism of the reconstructed garments. Note that this refinement is feasible thanks to the open surface representation allowed by our UDF model. Applying these physically inspired losses to an inflated garment, as produced by SMPLicit, ClothWild and DIG, yields poor results with many self-intersections, as shown in the supplementary material.

Fitting 3D scans. Given a 3D scan of a clothed person and segmentation information, we apply a strategy similar to the one presented above and minimize

$$L(\beta, \theta, \mathbf{z}) = d(D(\mathbf{G}, \beta, \theta, \mathbf{z}), \mathbf{S_G}) + \vec{d}(SMPL(\beta, \theta), \mathbf{S_B}), \quad (13)$$

w.r.t. $\mathbf{z}$, β and θ, where $\mathbf{S_G}$ and $\mathbf{S_B}$ denote the segmented garment and body scan points, and $d(a, b)$ and $\vec{d}(a, b)$ are the bidirectional and the one-directional Chamfer distance from b to a. Similarly to Eq. (12), we apply Eq. (13) to recover both the top and bottom garments. Fig. 10 shows our fitting results for some scans of the SIZER dataset [56]. The recovered 3D models closely match the input scans. Moreover, we can also apply a post-refinement procedure similar to the one described above, by minimizing both the physics-based losses from Eq. (7) and the Chamfer distance to the input scan w.r.t. the 3D coordinates of the vertices of the reconstructed models. This leads to even more realistic results, with fine wrinkles aligning to the input scans.

5. Conclusion

We have shown that physics-based self-supervision can be leveraged to learn a single parameterization for many different garments to be draped on human bodies in arbitrary poses. Our approach relies on UDFs to represent garment surfaces and on a displacement field to drape them, which enables us to handle a continuous manifold of garments without restrictions on their topology. Our whole pipeline is differentiable, which makes it suitable for solving inverse problems and for modeling clothed people from image data.

Future work will focus on modeling dynamic poses instead of only static ones. This is of particular relevance for loose clothes, where our reliance on the SMPL skinning prior should be relaxed. Moreover, we will investigate replacing our current global latent code by a set of local ones to yield finer-grained control both for garment editing and draping.

References

[1] M. Atzmon, N. Haim, L. Yariv, O. Israelov, H. Maron, and Y. Lipman. Controlling Neural Level Sets. In *Advances in Neural Information Processing Systems*, 2019. 2

[2] M. Atzmon and Y. Lipman. SALD: Sign Agnostic Learning with Derivatives. In *International Conference on Learning Representations*, 2020. 4

[3] D. Baraff and A. Witkin. Large Steps in Cloth Simulation. In *ACM SIGGRAPH*, pages 43–54, 1998. 1, 2

[4] H. Bertiche, M. Madadi, and S. Escalera. CLOTH3D: Clothed 3D Humans. In *European Conference on Computer Vision*, pages 344–359, 2020. 5

[5] H. Bertiche, M. Madadi, and S. Escalera. PBNS: Physically Based Neural Simulation for Unsupervised Garment Pose Space Deformation. *ACM Transactions on Graphics*, 2021. 2, 3, 4, 7

[6] H. Bertiche, M. Madadi, E. Tylson, and S. Escalera. DeePSD: Automatic Deep Skinning and Pose Space Deformation for 3D Garment Animation. In *International Conference on Computer Vision*, 2021. 2, 6, 16

[7] B. L. Bhatnagar, G. Tiwari, C. Theobalt, and G. Pons-Moll. Multi-Garment Net: Learning to Dress 3D People from Images. In *International Conference on Computer Vision*, 2019. 2

[8] T. Buffet, D. Rohmer, L. Barthe, L. Boissieux, and M-P. Cani. Implicit untangling: A robust solution for modeling layered clothing. *ACM Transactions on Graphics*, 38(4):1–12, 2019. 2, 15

[9] J. Chibane, A. Mir, and G. Pons-Moll. Neural Unsigned Distance Fields for Implicit Function Learning. In *Advances in Neural Information Processing Systems*, 2020. 2

[10] E. Corona, A. Pumarola, G. Alenya, G. Pons-Moll, and F. Moreno-Noguer. Smplicit: Topology-Aware Generative Model for Clothed People. In *Conference on Computer Vision and Pattern Recognition*, 2021. 2, 8, 15, 20

[11] A. Davydov, A. Remizova, V. Constantin, S. Honari, M. Salzmann, and P. Fua. Adversarial Parametric Pose Prior. In *Conference on Computer Vision and Pattern Recognition*, 2022. 8

[12] M. Designer, 2018. https : / / www . marvelousdesigner.com. 1

[13] Y. Duan, H. Zhu, H. Wang, L. Yi, R. Nevatia, and L. J. Guibas. Curriculum DeepSDF. In *European Conference on Computer Vision*, 2020. 4

[14] H. Fan, H. Su, and L. Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *Conference on Computer Vision and Pattern Recognition*, 2017. 16

[15] B. Guillard, F. Stella, and P. Fua. Meshudf: Fast and Differentiable Meshing of Unsigned Distance Field Networks. In *European Conference on Computer Vision*, 2022. 2, 3, 4, 20

[16] E. Gundogdu, V. Constantin, S. Parashar, A. Seifoddini, M. Dang, M. Salzmann, and P. Fua. Garnet++: Improving Fast and Accurate Static 3D Cloth Draping by Curvature Loss. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):181–195, 2022. 1, 2, 15

[17] E. Gundogdu, V. Constantin, A. Seifoddini, M. Dang, M. Salzmann, and P. Fua. Garnet: A Two-Stream Network for Fast and Accurate 3D Cloth Draping. In *International Conference on Computer Vision*, 2019. 1

[18] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, 2015. 12

[19] B. Jiang, J. Zhang, Y. Hong, J. Luo, L. Liu, and H. Bao. Bcnet: Learning body and cloth shape from a single image. In *European Conference on Computer Vision*, 2020. 2

[20] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimisation. In *International Conference on Learning Representations*, 2015. 14, 16

[21] T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares. Efficient Implementation of Marching Cubes' Cases with Topological Guarantees. In *Journal of Graphics Tools*, 2003. 2

[22] R. Li, B. Guillard, E. Remelli, and P. Fua. DIG: Draping Implicit Garment over the Human Body. In *Asian Conference on Computer Vision*, 2022. 2, 3, 4, 6, 7, 8, 15, 16, 17

[23] R. Li, M. Zheng, S. Karanam, T. Chen, and Z. Wu. Everybody Is Unique: Towards Unbiased Human Mesh Recovery. In *British Machine Vision Conference*, 2021. 7, 16

[24] Y. Li, M. Habermann, B. Thomaszewski, S. Coros, T. Beeler, and C. Theobalt. Deep physics-aware inference of cloth deformation for monocular human performance capture. In *International Conference on 3D Vision*, 2021. 2

[25] J. Liang, M. Lin, and V. Koltun. Differentiable Cloth Simulation for Inverse Problems. In *Advances in Neural Information Processing Systems*, 2019. 2

[26] T. Liu, S. Bouaziz, and L. Kavan. Quasi-newton methods for real-time simulation of hyperelastic materials. *ACM Transactions on Graphics*, 2017. 1

[27] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M.J. Black. SMPL: A Skinned Multi-Person Linear Model. *ACM SIGGRAPH Asia*, 34(6), 2015. 3, 4, 8

[28] Q. Ma, J. Yang, A. Ranjan, S. Pujades, G. Pons-Moll, S. Tang, and M. J. Black. Learning to Dress 3D People in Generative Clothing. In *Conference on Computer Vision and Pattern Recognition*, 2020. 1

[29] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black. AMASS: Archive of Motion Capture as Surface Shapes. In *International Conference on Computer Vision*, pages 5442–5451, 2019. 6

[30] I. Mehta, M. Chandraker, and R. Ramamoorthi. A Level Set Theory for Neural Implicit Evolution under Explicit Flows. In *European Conference on Computer Vision*, 2022. 2

[31] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 2, 12

9

[32] Ben Mildenhall, S. P. P., M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *European Conference on Computer Vision*, 2020. 12

[33] G. Moon, H. Nam, T. Shiratori, and K.M. Lee. 3d clothed human reconstruction in the wild. In *European Conference on Computer Vision*, 2022. 8, 15

[34] R. Narain, T. Pfaff, and J.F. O'Brien. Folding and crumpling adaptive sheets. *ACM Transactions on Graphics*, 2013. 2, 15

[35] R. Narain, A. Samii, and J.F. O'brien. Adaptive anisotropic remeshing for cloth simulation. *ACM Transactions on Graphics*, 2012. 1, 2, 15

[36] Nvidia. Nvcloth, 2018. 1

[37] Nvidia. NVIDIA Flex, 2018. https://developer.nvidia.com/flex. 1

[38] X. Pan, J. Mai, X. Jiang, D. Tang, J. Li, T. Shao, K. Zhou, X. Jin, and D. Manocha. Predicting loose-fitting garment deformations using bone-driven motion networks. In *ACM SIGGRAPH*, 2022. 2

[39] J. J. Park, P. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove. Deepsdf: Learning Continuous Signed Distance Functions for Shape Representation. In *Conference on Computer Vision and Pattern Recognition*, 2019. 2

[40] C. Patel, Z. Liao, and G. Pons-Moll. Tailornet: Predicting clothing in 3d as a function of human pose, shape and garment style. In *Conference on Computer Vision and Pattern Recognition*, 2020. 1, 2

[41] T. Pfaff, R. Narain, J.M. De Joya, and J.F. O'Brien. Adaptive tearing and cracking of thin sheets. *ACM Transactions on Graphics*, 33(4):1–9, 2014. 15

[42] G. Pons-Moll, S. Pujades, S. Hu, and M.J. Black. Clothcap: Seamless 4D Clothing Capture and Retargeting. *ACM SIGGRAPH*, 36(4):731–7315, July 2017. 1

[43] X. Provot. Collision and self-collision handling in cloth model dedicated to design garments. In *Computer Animation and Simulation*. 1997. 1

[44] Xavier Provot et al. Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In *Graphics interface*, 1995. 1

[45] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. PyTorch3D. https://github.com/facebookresearch/pytorch3d, 2020. 8, 16

[46] E. Remelli, A. Lukoianov, S. Richter, B. Guillard, T. Bagautdinov, P. Baque, and P. Fua. Meshsdf: Differentiable Iso-Surface Extraction. In *Advances in Neural Information Processing Systems*, 2020. 2

[47] I. Santesteban, M.A. Otaduy, and D. Casas. SNUG: Self-Supervised Neural Dynamic Garments. In *Conference on Computer Vision and Pattern Recognition*, 2022. 2, 3, 4, 5, 6, 19

[48] I. Santesteban, M.A. Otaduy, N. Thuerey, and D. Casas. Ulnef: Untangled layered neural fields for mix-and-match virtual try-on. In *Advances in Neural Information Processing Systems*, 2022. 2, 15

[49] I. Santesteban, M. A. Otaduy, and D. Casas. Learning-Based Animation of Clothing for Virtual Try-On. *Computer Graphics Forum (Proc. of Eurographics)*, 33(2), 2019. 1

[50] I. Santesteban, N. Thuerey, M. A. Otaduy, and D. Casas. Self-Supervised Collision Handling via Generative 3D Garment Models for Virtual Try-On. In *Conference on Computer Vision and Pattern Recognition*, 2021. 2, 4

[51] Y. Shen, J. Liang, and M.C. Lin. Gan-based garment generation using sewing pattern images. In *European Conference on Computer Vision*, 2020. 1

[52] Optitext Fashion Design Software, 2018. https://optitex.com/. 1

[53] Tongkui Su, Yan Zhang, Yu Zhou, Yao Yu, and Sidan Du. GPU-based Real-time Cloth Simulation for Virtual Try-on. In *Pacific Conference on Computer Graphics and Applications*, 2018. 1, 2

[54] M. Tang, R. Tong, R. Narain, C. Meng, and D. Manocha. A GPU-based streaming algorithm for high-resolution cloth simulation. In *Computer Graphics Forum*, 2013. 1

[55] M. Tang, T. Wang, Z. Liu, R. Tong, and D. Manocha. I-Cloth: Incremental Collision Handling for Gpu-Based Interactive Cloth Simulation. In *ACM Transactions on Graphics*, 2018. 15

[56] G. Tiwari, B. L. Bhatnagar, T. Tung, and G. Pons-Moll. Sizer: A Dataset and Model for Parsing 3D Clothing and Learning Size Sensitive 3D Clothing. In *European Conference on Computer Vision*, 2020. 1, 2, 8, 20

[57] T. Vassilev, B. Spanlang, and Y. Chrysanthou. Fast cloth animation on walking avatars. In *Computer Graphics Forum*, 2001. 1

[58] R. Venkatesh, T. Karmali, S. Sharma, A. Ghosh, R. V. Babu, L. A. Jeni, and M. Singh. Deep Implicit Surface Point Prediction Networks. In *International Conference on Computer Vision*, 2021. 2

[59] R. Vidaurre, I. Santesteban, E. Garces, and D. Casas. Fully Convolutional Graph Neural Networks for Parametric Virtual Try-On. In *Computer Graphics Forum*, 2020. 1

[60] H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A.C. Courville. Modulating Early Visual Processing by Language. In *Advances in Neural Information Processing Systems*, 2017. 4, 12, 13

[61] T. Y. Wang, D. Ceylan, J. Popovic, and N. J. Mitra. Learning a Shared Shape Space for Multimodal Garment Design. In *ACM SIGGRAPH Asia*, 2018. 1

[62] Y. Wang, Y. Sun, Z. Liu, S. Sarma, M. Bronstein, and J.M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. In *ACM Transactions on Graphics*, 2019. 12, 13

[63] Yu Y. Rong, T. Shiratori, and H. Joo. Frankmocap: Fast monocular 3d hand and body motion capture by regression and integration. In *International Conference on Computer Vision Workshops*, 2021. 7

[64] L. Yang, Q. Song, Z. Wang, M. Hu, C. Liu, X. Xin, W. Jia, and S. Xu. Renovating parsing R-CNN for accurate multiple human parsing. In *European Conference on Computer Vision*, 2020. 7

[65] I. Zakharkin, K. Mazur, A. Grigorev, and V. Lempitsky. Point-based modeling of human clothing. In *International Conference on Computer Vision*, 2021. 2

[66] C. Zeller. Cloth simulation on the gpu. In *ACM SIGGRAPH*. 2005. 1

[67] F. Zhao, W. Wang, S. Liao, and L. Shao. Learning Anchored Unsigned Distance Functions with Gradient Direction Alignment for Single-View Garment Reconstruction. In *International Conference on Computer Vision*, 2021. 2, 4

Supplementary Material

In this appendix we first provide more details about our networks and their architectures in Sec. 6.

In Sec. 7 we expand on the choice and formulations of some loss terms we use. Importantly, in Sec. 7.4 we explain the physics-based refinement procedure used in the main paper, and show that *modelling garments as open surfaces is necessary for it.*

Then in Sec. 8 we report additional quantitative and qualitative results of our pipeline and the runtime of its components. Finally, in Sec. 9 we describe how human ratings were collected.

6. Network Architectures and Training

6.1. Garment Generative Network

6.1.1 Garment Encoder

To encode a given garment into a compact latent code, we first sample P points from its surface and then we feed them to a DGCNN [62] encoder, detailed in Fig. 11. The input point cloud is processed by four *edge convolution* layers, which project the input 3D points into features with increasing dimensionality – *i.e.*, 64, 64, 128 and finally 256.

Each edge convolution layer works as follows. For each input point, the features from its K neighbours are collected and used to prepare a matrix with K rows. Each row is the concatenation of two vectors: $\mathbf{f}_i - \mathbf{f}_0$ and $\mathbf{f}_0$, $\mathbf{f}_i$ and $\mathbf{f}_0$ being respectively the feature vector of the i-th neighbour and the feature vector of the considered point. Each row of the resulting matrix is then transformed independently to the desired output dimension. The output feature vector for the considered point is finally obtained by applying max pooling along the rows of the produced matrix.

The original DGCNN implementation recomputes the neighborhoods in each edge convolution layer, using the distance between the feature vectors as metric. This can be explained by the original purposes of DGCNN, *i.e.*, point cloud classification and part segmentation. Since we are interested in encoding the geometric details of the input point cloud, we compute neighborhoods only once based on the euclidean distance of the points in the 3D space and reuse this information in every edge convolution layer. We set $K = 16$ in our experiments.

The feature vectors from the four edge convolutions are then concatenated to form a single vector with 512 elements, that is fed to a final linear layer paired with batch normalization and leaky ReLU. Such layer projects the 512 sized vectors into the final desired dimension, which is 32 in our case. The final latent code is obtained by compressing the feature matrix with shape $P \times 32$ along the first dimension with max pooling.

6.1.2 Garment Decoder

The garment generative network features an implicit decoder that can predict the unsigned distance field of a garment starting from its latent code. More specifically, the decoder is a coordinate-based MLP that takes as inputs the garment latent code and a 3D query. Using the latent code as condition, the decoder predicts the unsigned distance from the query to the garment surface.

Our UDF decoder, shown in Fig. 12, is inspired by [31]. The input 3D query is first mapped to a higher dimensional space ($\mathbb{R}^{63}$) with the positional encoding proposed in [32], which is known to improve the capability of the network to approximate high frequency functions. The encoded query is then mapped with a linear layer to $\mathbb{R}^{512}$ and then goes through 5 residual blocks. The output of each block is computed as $\mathbf{f}_{out} = \mathbf{f}_{in} + \Delta\mathbf{f}$, where $\mathbf{f}_{in}$ is the input vector and $\Delta\mathbf{f}$ is a residual term predicted by two consecutive linear layers starting from $\mathbf{f}_{in}$. The size of the feature vector is 512 across the whole sequence of residual blocks. The output of the last block is mapped to the scalar output $out \in \mathbb{R}$ with a final linear layer.

All the linear layers but the output one are paired with Conditional Batch Normalization (CBN) [60] and ReLU activation function. CBN is used to condition the MLP with the input latent code $\mathbf{z}$. In more details, each CBN module applies standard batch normalization [18] to the input vectors, with the difference that the parameters of the final affine transformation are not learned during the training but are instead predicted at each inference step by dedicated linear layers starting from $\mathbf{z}$.

Finally, recall that our generative network is trained with the binary cross-entropy loss. Thus, the output of the decoder must be converted to the corresponding UDF value by first applying the sigmoid function and then scaling the result with the UDF clipping distance δ, which we set to 0.1 in our experiments. Such procedure is indeed the dual of the one applied on the UDF ground-truth labels during training to normalize them in the range $[0, 1]$.

6.1.3 Surface Sampling

We sample supervision points with a probability inversely proportional to the distance to the surface: 30% of the points are sampled directly on the input surface, 30% are sampled by adding gaussian noise with ϵ variance to surface points, 30% are obtained with gaussian noise with 3ϵ variance, and the remaining ones are gathered by sampling uniformly the bounding box in which the garment is contained. Since in our experiments, the top and bottom garments are normalized respectively into the upper and lower halves of the $[-1, 1]^3$ cube, we set $\epsilon = 0.003$.
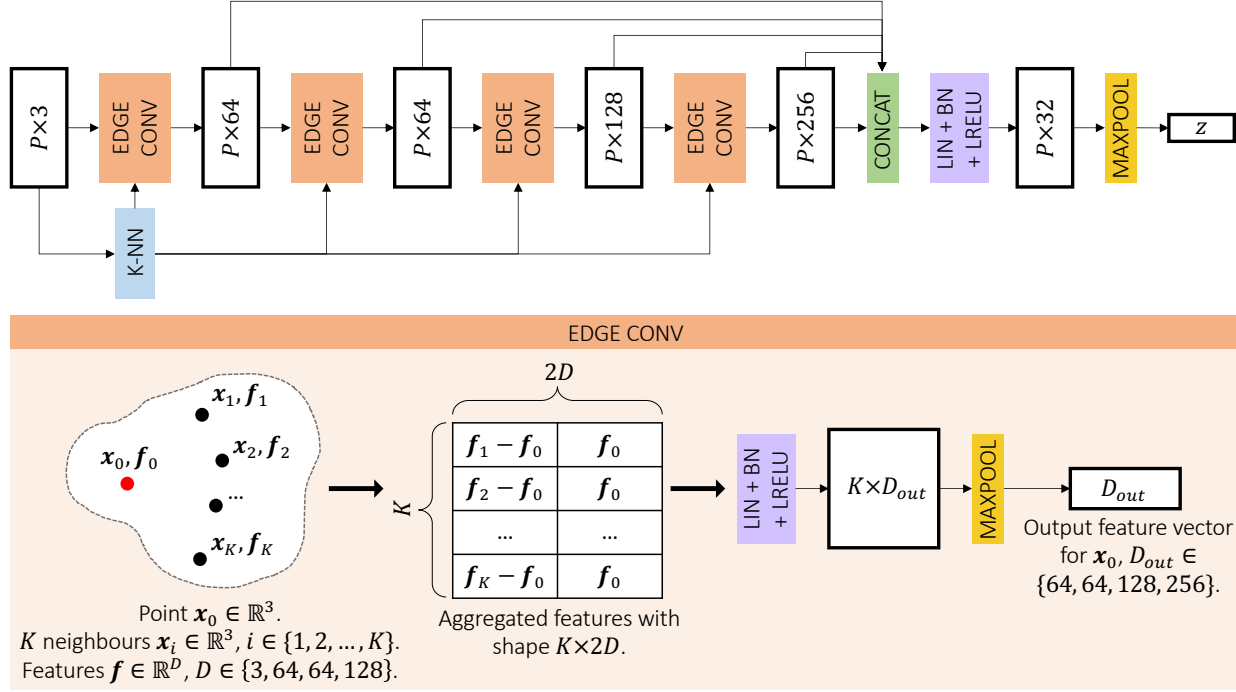
Figure 11. **DGCNN point cloud encoder.** We adopt DGCNN [62] as the point cloud encoder of our garment generative network. The input cloud is passed through four *edge convolutions*, which gather features of local neighborhoods of points to project them into higher dimensional spaces. The features from all the layers are then concatenated and projected to the final desired dimension. Max pooling is finally used to obtain the latent code **z** for the input cloud. *CONCAT* stands for features concatenation, while *LIN + BN + LRELU* represents a linear layer followed by batch normalization and leaky ReLU.
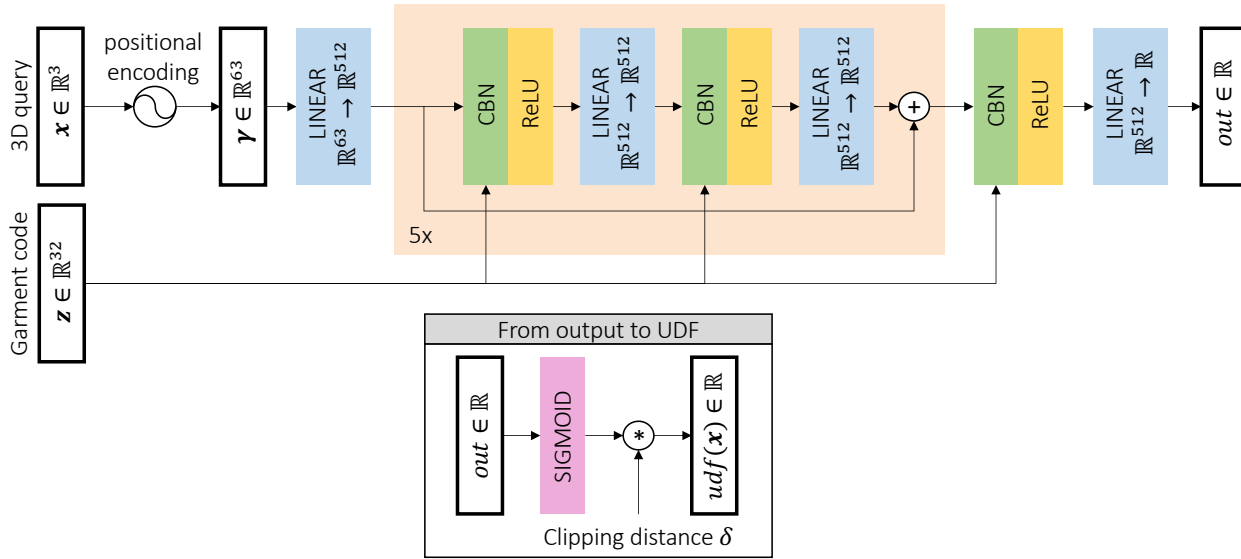
Figure 12. **UDF decoder.** Given a 3D query and a garment latent code, the decoder of our garment generative network is trained to predict the UDF of the input query w.r.t. the surface of the garment. The latent code is used to condition the prediction by the means of Conditional Batch Normalization (CBN) [60]. Since we trained the decoder with the binary cross-entropy loss, its outputs need to be converted to UDF values by applying the sigmoid function and then scaling the result with the UDF clipping distance δ.

6.2. Draping Network

The networks $\mathcal{W}(\mathbf{x}) \in \mathbb{R}^{24}$ and $\Delta x(\mathbf{x}, \beta) \in \mathbb{R}^3$ that predict blending weights and coarse displacements are implemented by a 9-layer multilayer perceptron (MLP) with a skip connection from the input layer to the middle. Each layer has 256 nodes except the middle and the last ones. ReLU is used as the activation function. The body-parameter-embedding module $\mathcal{B}(\beta, \theta) \in \mathbb{R}^{128}$ and the displacement-matrix module $\mathcal{M}(x, \mathbf{z}) \in \mathbb{R}^{128 \times 3}$ for Δx_{ref} are implemented by a 5-layer MLP with LeakyReLU activation in-between. Each layer has 512 nodes except the last one. Δx_{IS} uses the same architecture as Δx_{ref}.

6.3. Training Hyperparameters

The generative models (top/bottom ones) are trained on the 600/300 neutral garments for 4000 epochs, using mini-batches of size $B = 4$. Each item of the mini-batch contains an input point cloud with $P = 10,000$ points and $N = 20,000$ random UDF 3D queries. The dimension of the latent codes is set to 32 for both top and bottom garments, and we set $\lambda_g = 0.1$ in

$$\mathcal{L}_{garm} = \mathcal{L}_{dist} + \lambda_g \mathcal{L}_{grad} . \tag{14}$$

The draping networks are trained for 250K iterations with mini-batches of size 18, where each item is composed of the vertices of one garment paired with one body shape and pose. We set $\lambda = 0.1$ for $\mathcal{L}_{pin}$ and $\gamma = 0.5$ for $\mathcal{L}_{layer}$.

Both the generative and the draping networks are trained with Adam optimizer [20] and learning rates set to 0.0001 and 0.001 respectively.

7. Loss Terms and Ablation Studies

7.1. $\mathcal{L}_{garm}$ for Garment Reconstruction

We report here an ablation study that we conducted to determine the best formulation for $\mathcal{L}_{garm}$, the loss function presented in Eq. (1) of the main paper, that we use to train our garment generative network.

In particular, we consider three variants for $\mathcal{L}_{dist}$, the term of the supervision signal that guides the network to predict accurate values for the garments UDF. In addition to the binary cross-entropy loss (BCE) presented in Eq. (4) of the main paper, we study the possibility of using more traditional regression losses, such as L1 and L2 losses. Adopting the notation introduced in Sec. 3.1 of the main paper, the L1 loss is defined as $\frac{1}{BN} \sum_{i,j} |min(y_{ij}, \delta) - \widetilde{y}_{ij}|$, while the L2 loss is computed as $\frac{1}{BN} \sum_{i,j} (min(y_{ij}, \delta) - \widetilde{y}_{ij})^2$.

On top of the three variants for $\mathcal{L}_{dist}$, we also consider for each one the possibility of removing the gradients supervision from $\mathcal{L}_{garm}$, $i.e.$, setting $\lambda_g = 0$.

We trained our generative network for 48 hours with the resulting six loss function variants and then compared the

quality of the garments reconstructed with the garment decoder. Fig. 13 presents a significant example of what we observed on the test set. Without gradients supervision (top row of the figure), none of the considered loss functions (BCE, L1 or L2) can guide the network to predict smooth surfaces without artifacts or holes. Adding the gradients supervision (bottom row) induces a strong regularization on the predicted distance fields, helping the network to predict surfaces without holes in most of the cases. However, using the L1 loss leads to rough surfaces, as one can observe in the center column of the bottom row of the figure. The BCE and the L2 losses (first and third columns of the bottom row), instead, produce smooth surfaces that are pleasant to see. We finally opted for the BCE loss over the L2 loss, since the network trained with the latter occasionally predicts surfaces with small holes, as in the example shown in the figure.

7.2. $\mathcal{L}_{pin}$ for Bottom Garments

To determine V, the set of bottom garment vertices that need to be constrained by $\mathcal{L}_{pin}$, we first find the closest body vertex v_B for each bottom garment vertex v. If v_B locates in the body trunk (cyan region as shown in Fig. 14), v is added to V.

In Fig. 15, we show the draping results of bottom garments by using different values for λ in $\mathcal{L}_{pin}$. When λ equals 0 or 1, the deformations along the X and Z axes are not natural because no constraints or too strong constraints are applied, while it is not the case when $\lambda = 0.1$, which is our setting.

7.3. $\mathcal{L}_{layer}$ for Top-bottom Intersection

To determine C, the set of body vertices covered by both the top and bottom garments, we first subdivide the SMPL body mesh for a higher resolution, and then we compute C_{top} the set of closest body vertices for the given top garment, and C_{bottom} the set of closest body vertices for the bottom. C is derived as the intersection of C_{top} and C_{bottom}.

In Fig. 16 we compare the results of models trained without and with $\mathcal{L}_{layer}$. We can observe that without $\mathcal{L}_{layer}$, the top tank can intersect with the bottom trousers, while it is not the case when using $\mathcal{L}_{layer}$. This indicates the efficacy of $\mathcal{L}_{layer}$ to avoid intersections between garments.

7.4. Physics-based Refinement

After recovering the draped garment $\mathbf{G}_D$ from images by the optimization of Eq. (12) of the main paper, we can apply the physics-based objectives of Eq. (7) (main paper) to increase its level of realism

$$
\begin{aligned}
L(\Delta_{\mathbf{G}}) = & \mathcal{L}_{strain}(\mathbf{G}_D + \Delta_{\mathbf{G}}) + \mathcal{L}_{bend}(\mathbf{G}_D + \Delta_{\mathbf{G}}) \\
& + \mathcal{L}_{gravity}(\mathbf{G}_D + \Delta_{\mathbf{G}}) + \mathcal{L}_{col}(\mathbf{G}_D + \Delta_{\mathbf{G}}) ,
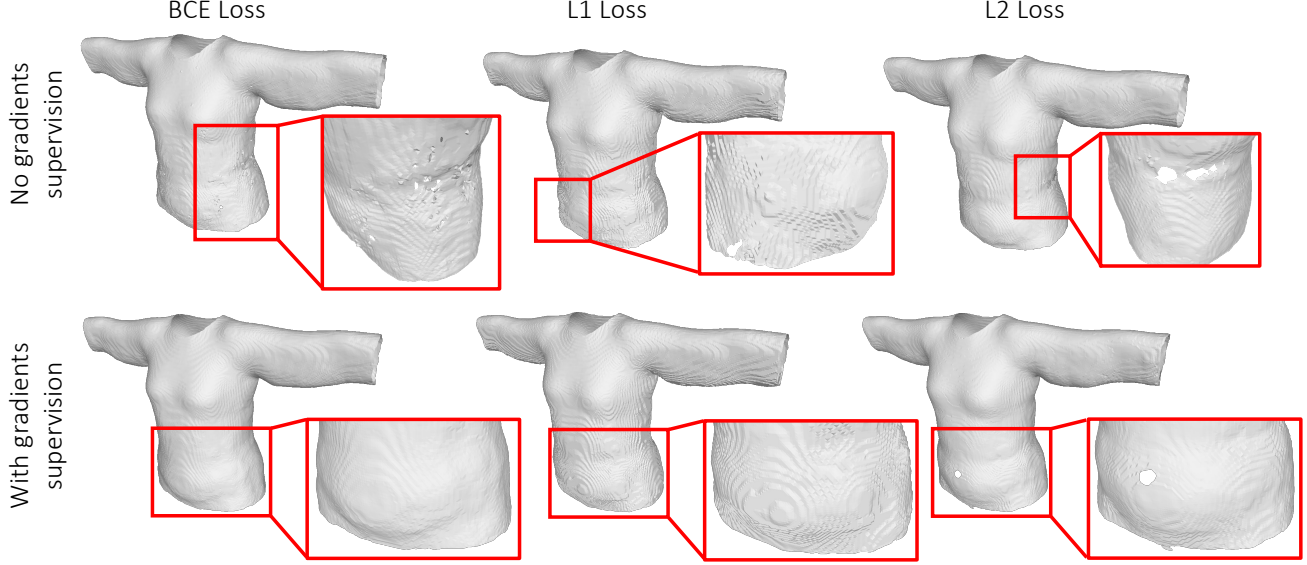\end{aligned}
\tag{15}
$$

Figure 13. **Comparison between different loss functions for the garment generative network.** We present the same garment reconstructed by our generative network after being trained for 48 hours with six different alternatives of loss functions.
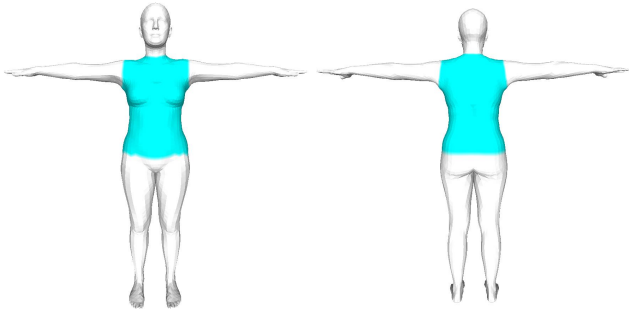

Figure 14. Body region (marked in cyan) used to compute $\mathcal{L}_{pin}$.

where $\Delta_\mathbf{G}$ is the per-vertex-displacement initialized to zero. For the recovery from 3D scans, we apply the following optimization which minimizes both the above physics-based objectives and the Chamfer Distance $d(\cdot)$ to the input scan $\mathbf{S_G}$

$$
\begin{aligned}
L(\Delta_\mathbf{G}) =& \mathcal{L}_{strain}(\mathbf{G}_D + \Delta_\mathbf{G}) + \mathcal{L}_{bend}(\mathbf{G}_D + \Delta_\mathbf{G}) \\
&+ \mathcal{L}_{gravity}(\mathbf{G}_D + \Delta_\mathbf{G}) + \mathcal{L}_{col}(\mathbf{G}_D + \Delta_\mathbf{G}) \\
&+ d(\mathbf{G}_D + \Delta_\mathbf{G}, \mathbf{S_G}) \,.
\end{aligned} \tag{16}
$$

This refinement procedure is only applicable to open surface meshes, and our UDF model is thus key to enabling it. Applying Eq. (15) or Eq. (16) to an inflated garment (as recovered by SMPLicit [10], ClothWild [33] and DIG [22]) indeed yields poor results with many self-intersections as illustrated in Fig. 17. This is because inflated garments modelled as SDFs have a non-zero thickness, with distinct inner and outer surfaces whose interactions are not taken into account in this fabric model. The physics model we apply
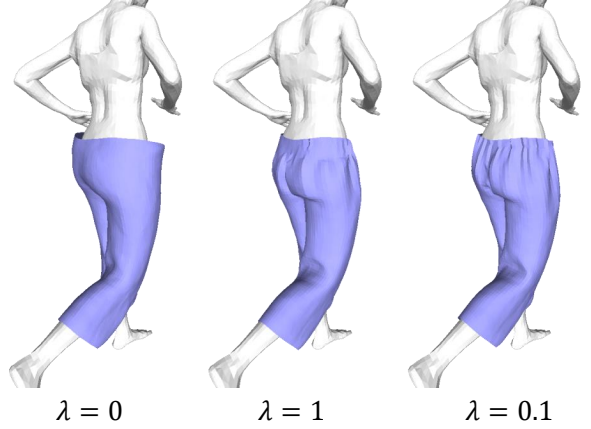

Figure 15. **Comparison between different values for λ of $\mathcal{L}_{pin}$.** To restrict the deformation mainly along the vertical direction (Y axis) and produce natural deformations along other directions, λ has to be a positive value smaller than 1. We use $\lambda = 0.1$ for our training.

on garment meshes indeed considers collisions of the garment with the body, but not with itself, which is what happens with the inner and outer surfaces in Fig. 17. Adding a physics term to prevent self intersections would not be trivial, and is related to the complex task of untangling layered garments [8, 48]

Note that this is also the case for most garment draping softwares [16, 34, 35, 41, 55] to expect single layer garments. Modeling garment with UDFs is thus a key feature of our pipeline for its integration in downstream tasks.
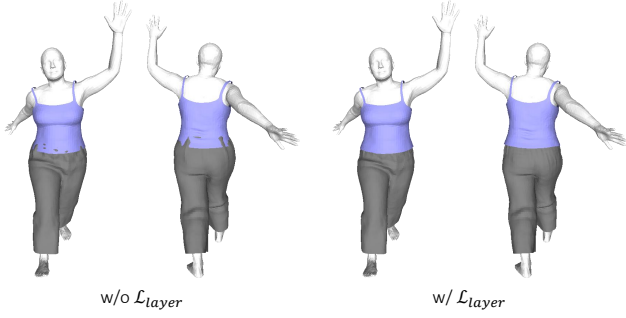
w/o $\mathcal{L}_{layer}$ w/ $\mathcal{L}_{layer}$

Figure 16. **Comparison: draping without and with $\mathcal{L}_{layer}$.** Without it, the top and bottom garments intersect with each other.

Both the optimizations of Eqs. (12) and (13) of the main paper and Eqs. (15) and (16) are done with Adam [14] but with different learning rates set to 0.01 and 0.001 respectively.
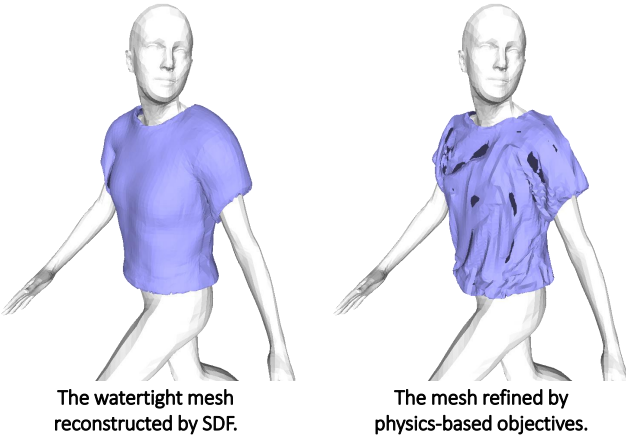

The watertight mesh reconstructed by SDF.   The mesh refined by physics-based objectives.

Figure 17. **Applying post-refinement procedure to watertight mesh**. Left: the watertight mesh reconstructed by DIG [22]. Right: the same mesh after being refined with physics-based objectives (Eq. (15)). Physics-based refinement is not compatible with inflated garment meshes, and leads to many self-intersections.

8. Additional Results

8.1. Garment Encoder/Decoder

8.1.1 Additional Qualitative Results

Fig. 18 and Fig. 19 show the encoding-decoding capabilities of our garment generative network for top and bottom *test* garments, respectively. The ground-truth garments are passed through the garment encoder, which produces a compact latent code for each clothing item. Then, our garment decoder reconstructs the input garments surface from the latent codes. It is possible to notice how the output gar-

ments closely match the input ones, both in terms of geometry and topology.

8.1.2 Latent Space Optimization (LSO).

After training the garment generative network, we obtain a latent space that allows us to sample a garment latent code and to feed it to the implicit decoder to reconstruct the explicit surface. We study here the possibility of exploring the garment latent space by the means of LSO. To do that, given a target 2D silhouette or a sparse 3D point cloud of a garment, we optimize with gradient descent a latent code – initialized to the training codes average – so that the frozen decoder conditioned on it can produce a garment which fits the target image or point cloud.

Given the silhouette $\mathcal{S}$ of a target garment, we can retrieve its latent code $\mathbf{z}$ by minimizing

$$
\begin{aligned}
L(\mathbf{z}) &= L_{\text{IoU}}(R(\mathbf{G}), \mathcal{S}) , \\
\mathbf{G} &= \text{MeshUDF}(D_G(\cdot, \mathbf{z})) ,
\end{aligned}
\tag{17}
$$

where L_{IoU} is the IoU loss [23] in pixel space measuring the difference between 2D silhouettes , $R(\cdot)$ is a differentiable silhouette renderer for meshes [45], and $\mathbf{G}$ is the garment mesh reconstructed with our garment decoder using $\mathbf{z}$.

In the case of a target garment provided as a point cloud $\mathcal{P}$, the garment latent code $\mathbf{z}$ can be obtained by minimizing

$$
\begin{aligned}
L(\mathbf{z}) &= d(ps(\mathbf{G}), \mathcal{P}) , \\
\mathbf{G} &= \text{MeshUDF}(D_G(\cdot, \mathbf{z})) ,
\end{aligned}
\tag{18}
$$

where $d(a, b)$ is the Chamfer distance [14] between point clouds a and b, and $ps(\cdot)$ represents a differentiable procedure to sample points from a given mesh [45].

In both cases, we run the optimization for 1000 steps, with Adam optimizer [20] and learning rate set to 0.01.

In Fig. 20 and Fig. 21 we present some results of the LSO procedures here described, showing that the latent space learned by the garment generative network can be explored effectively with gradient descent to recover the codes associated with the target garments.

8.2. Draping Network

8.2.1 Additional Qualitative Results

In Fig. 22 we show additional qualitative results of garment draping produced by our method, where the garment meshes are generated by our UDF model. It can be seen that our method can realistically drape garments with different topologies over bodies of various shapes and poses.

8.2.2 Euclidean Distance is not a Good Metric

In Fig. 23, we show an example of bottom garment where our result is more realistic than the competitors DeePSD [6]
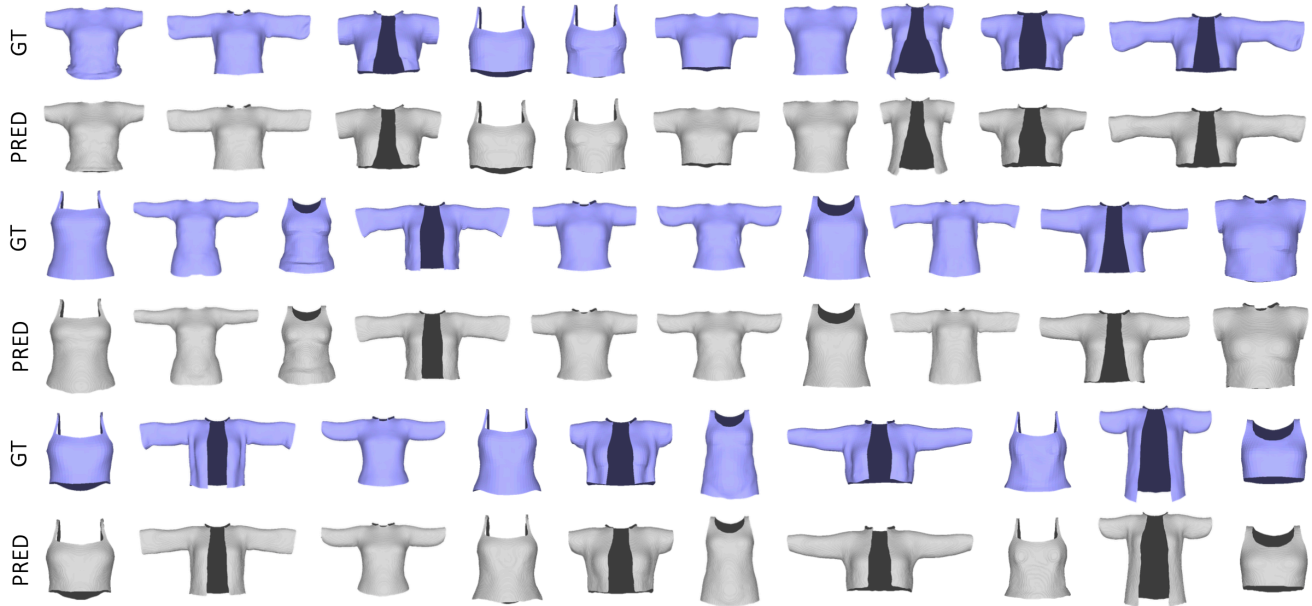
Figure 18. **Generative network: reconstruction of unseen garments in neutral pose/shape (top garments).** Latent codes for unseen garments can be obtained with our garment encoder. These codes are then used by the garment decoder to reconstruct open surface meshes. Input garments are colored in purple, while the reconstructed meshes are colored in gray.
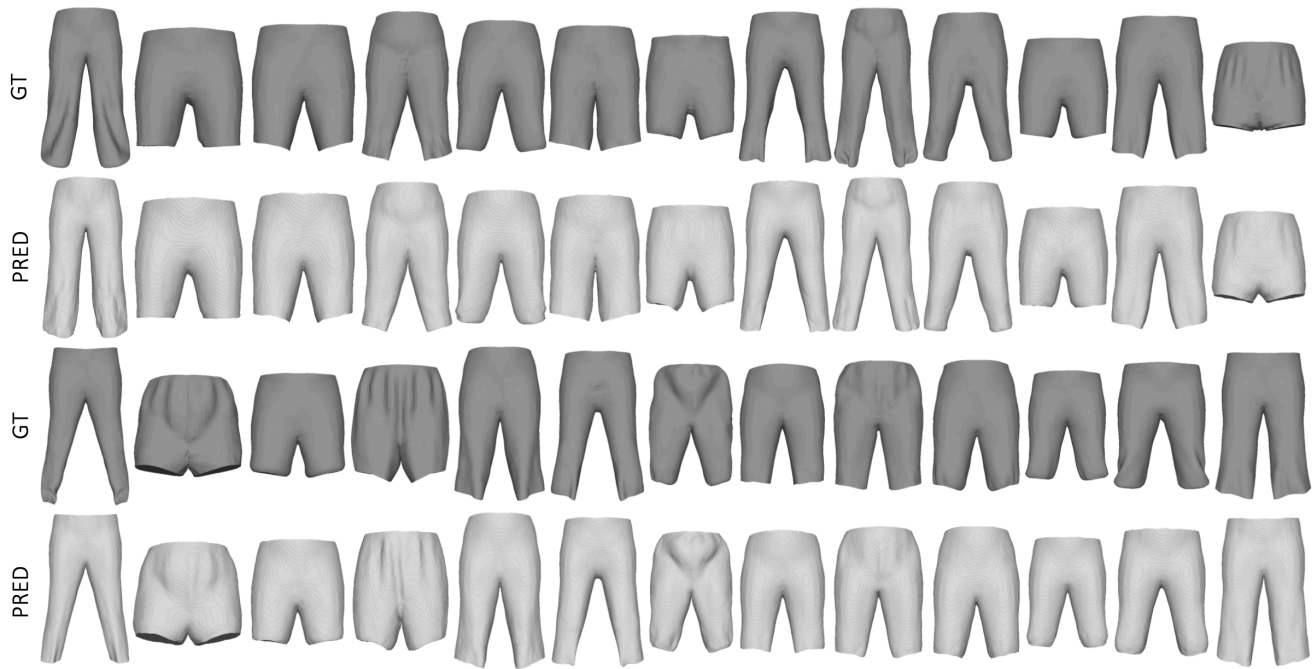

Figure 19. **Generative network: reconstruction of unseen garments in neutral pose/shape (bottom garments).** Latent codes for unseen garments can be obtained with our garment encoder. These codes are then used by the garment decoder to reconstruct open surface meshes. Input garments are colored in dark gray, while the reconstructed meshes are colored in light gray.

and DIG [22] despite having the highest Euclidean distance. This demonstrates again that Euclidean distance is not able to measure the draping quality, as discussed in the main paper.
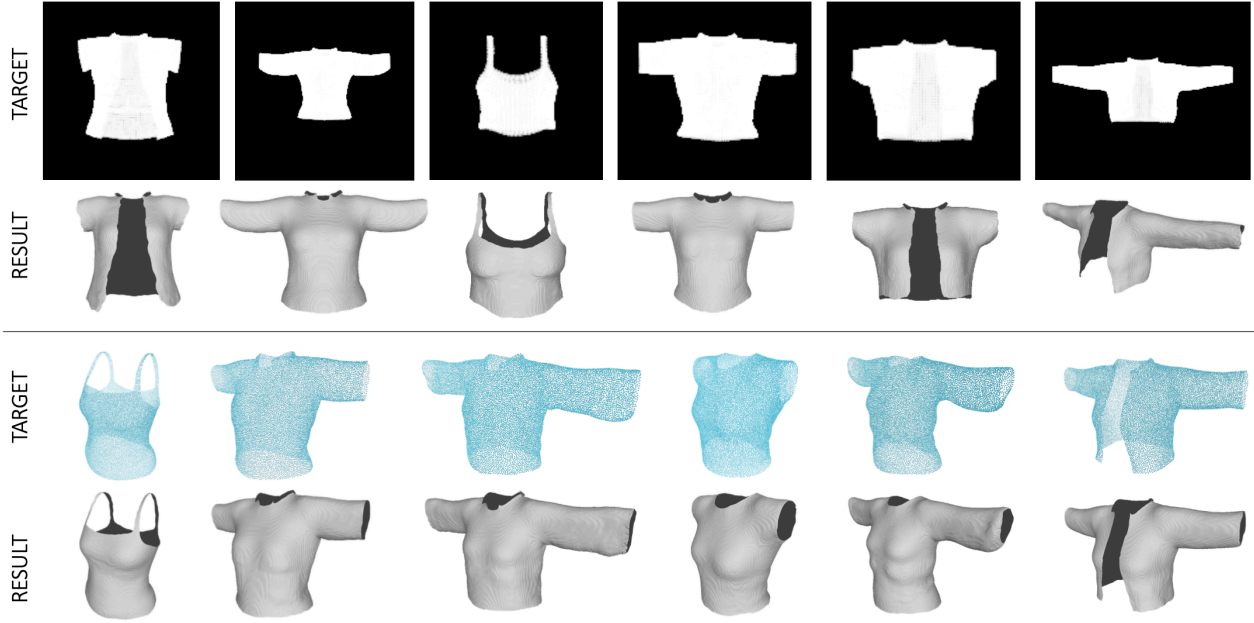
Figure 20. **Generative network: latent space optimization (top garments).** After training, we can explore the latent space learned by the garment generative network with gradient descent, to recover target garments from 2D silhouettes (top) or 3D point clouds (bottom).
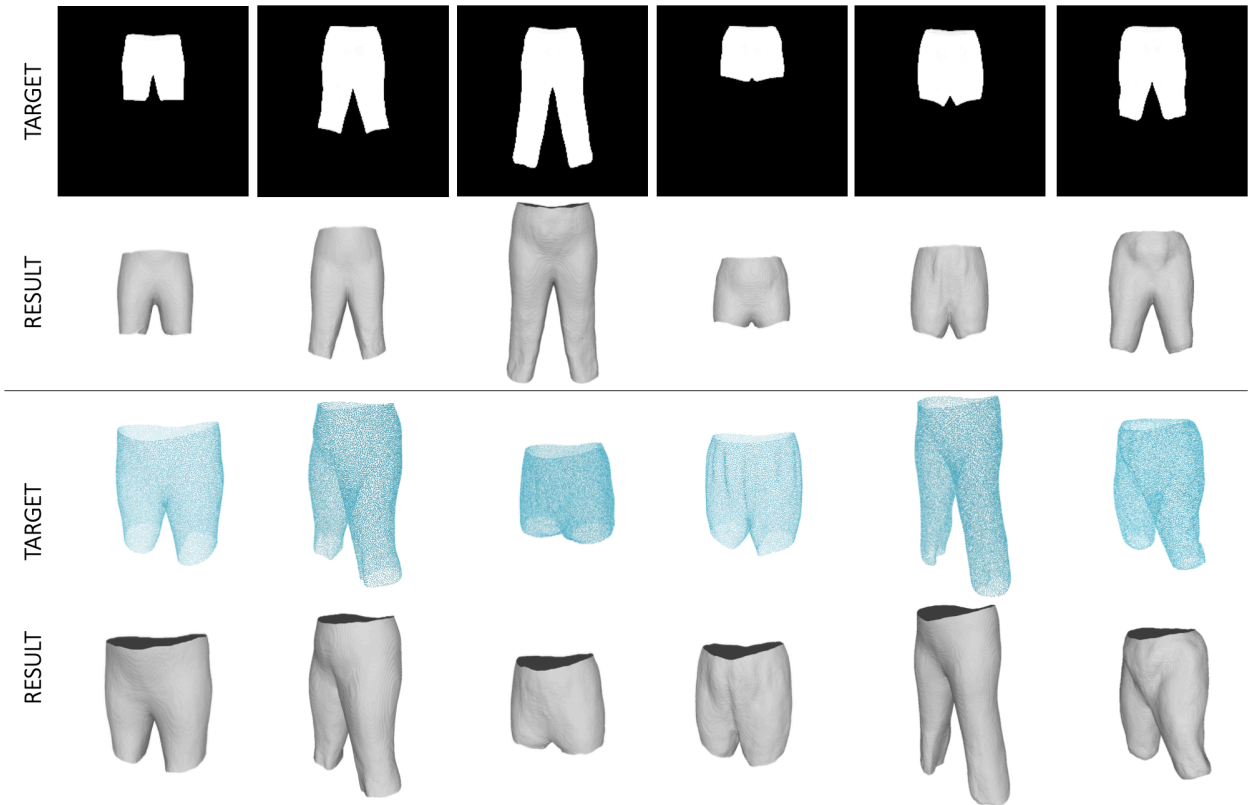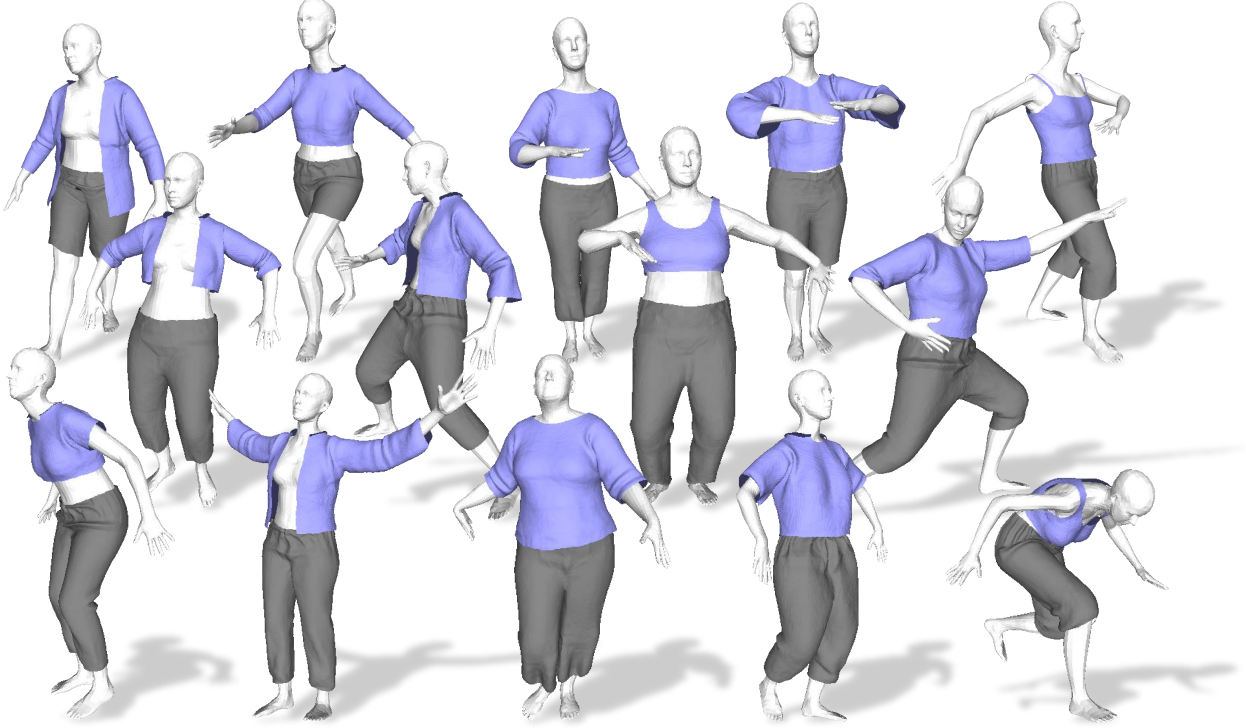

Figure 21. **Generative network: latent space optimization (bottom garments).** After training, we can explore the latent space learned by the garment generative network with gradient descent, to recover garments from 2D silhouettes (top) or 3D point clouds (bottom).

Figure 22. **Additional results:** draping garments of different topologies over bodies in various shapes and poses with our method.

Top	Strain ↓	Bending ↓	Gravity ↓	Total ↓	Bottom	Strain ↓	Bending ↓	Gravity ↓	Total ↓
DeePSD	7.22	**0.01**	**0.98**	8.21	DeePSD	8.46	0.02	0.90	9.38
DIG	6.32	**0.01**	1.05	7.38	DIG	7.48	**0.01**	0.90	8.39
Ours	**0.43**	**0.01**	1.05	**1.81**	Ours	**0.41**	**0.01**	**0.86**	**1.28**

Table 2. **Draping unseen garment meshes.** Quantitative comparison in physics-based energy between DeePSD, DIG and our method. "Strain", "Bending" and "Gravity" denote the membrane strain energy, the bending energy and the gravitational potential energy, respectively.
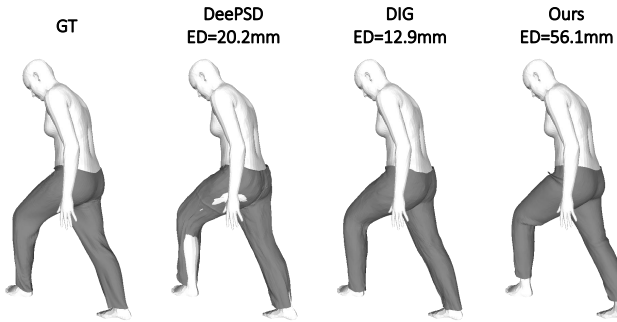
Figure 23. **Comparison between DeePSD, DIG and our method.** Our result is more realistic than the others despite having the highest Euclidean distance (ED) error.

8.2.3 Quantitative Evaluation in Physics-based Energy

In Tab. 2, we report the physics-based energy of *Strain*, *Bending* and *Gravity* as proposed by [47] on test garment meshes when draped by DeePSD, DIG and our method. These energy terms are used as training losses for our garment network (Eqs. (7) and (8) of the main paper). For the gravitational potential energy, we choose the lowest body vertex as the 0 level. Generally, our results have the lowest energies, especially for the *Strain* component. Since DeePSD and DIG do not apply constraints on mesh faces, their results exhibit much higher *Strain* energy. This indicates that our method can produce results that have more realistic physical properties.

19

8.3. Inference Times

We report inference times for the components of our framework, computed on an NVIDIA Tesla V100 GPU. The garment encoder, which needs to be run only once for each garment, takes ∼25 milliseconds. The decoder takes ∼2 seconds to reconstruct an explicit garment mesh from a given latent code, including the modified Marching Cubes from [15] at resolution 256^3.

The draping network takes ∼5 ms to deform a garment mesh composed of 5K vertices. Since it is formulated in an implicit manner and is queried at each vertex, its inference time increases to ∼8 ms for a mesh with 8K vertices, or ∼53 ms with 100K vertices.

8.4. Fitting SMPLicit [10] to 3D Scans
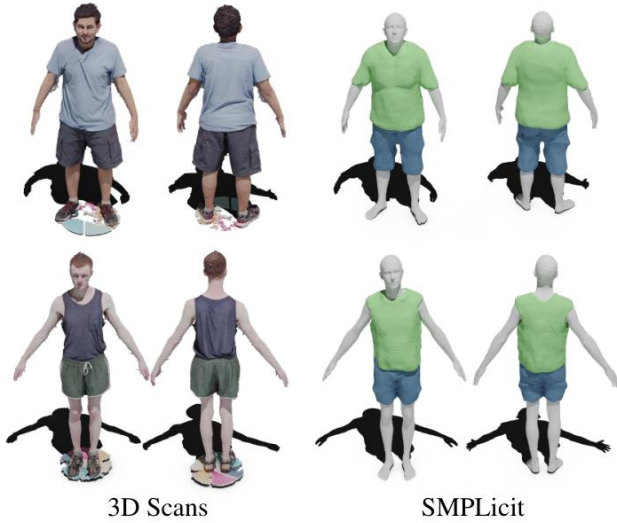
3D Scans   SMPLicit

Figure 24. **Recovered garments of SMPLicit from 3D scans**. Figures are extracted from [10].

In Fig. 24 we show results of fitting the concurrent approach SMPLicit [10] to 3D scans of the SIZER dataset [56]. We can observe that they are not as realistic as ours shown in Fig. 10 of the main paper. Since we have no access to their code and not enough information for a re-implementation, we directly extract this figure from [10].

9. Human Evaluation

In Fig. 25 we show the interface and instructions that were presented to the 187 respondents of our survey. These evaluators were volunteers with various backgrounds from the authors respective social circles, which were purposely not given any further detail or instruction. We collected collected 3738 user opinions in total, each user expressing 20 opinions on average.
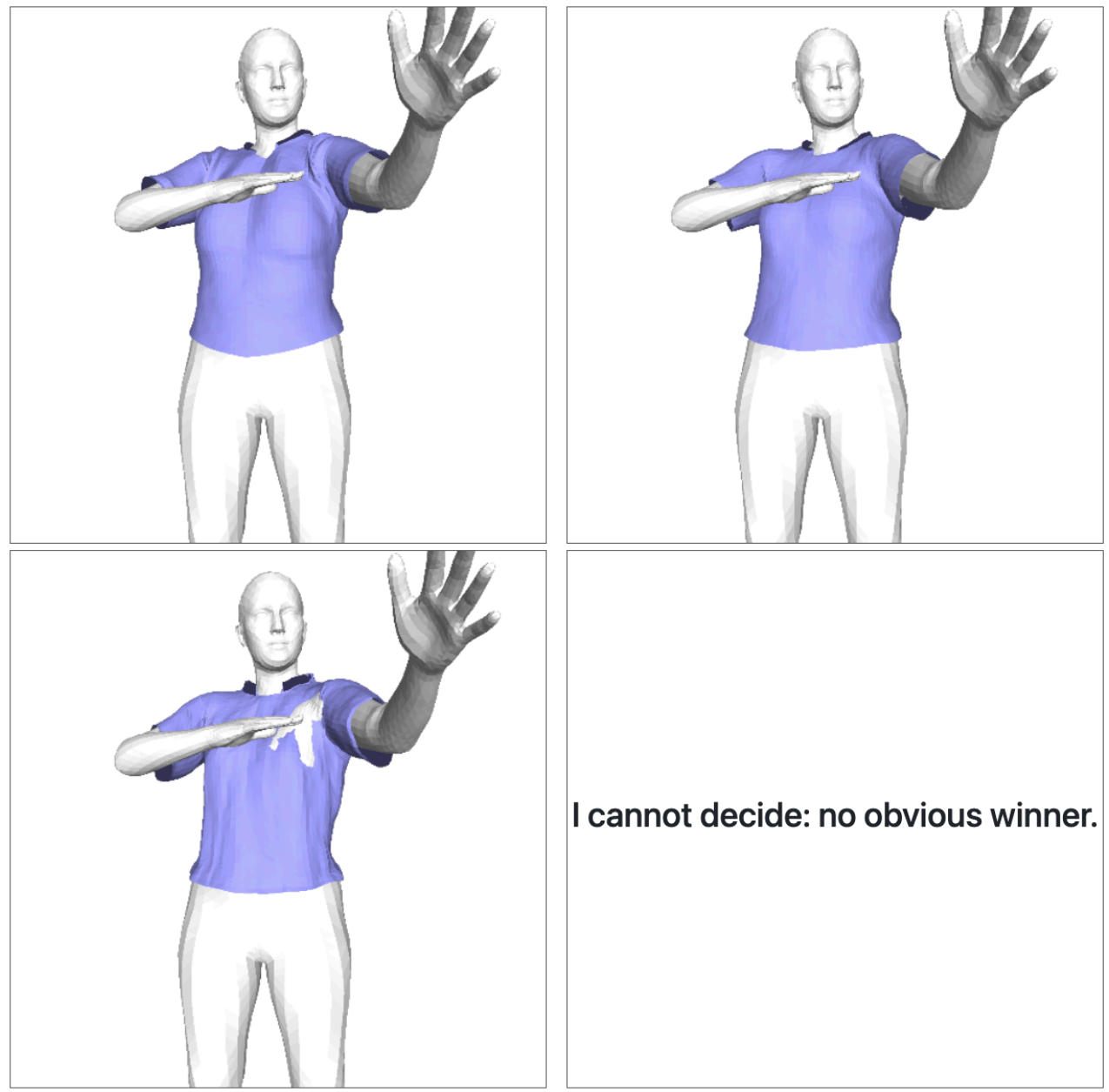
Figure 25. **Interface of our qualitative survey.** The garment is draped with our method, DIG, and DeePSD, in a random order.