ULNeF: Untangled Layered Neural Fields for Mix-and-Match Virtual Try-On

Igor Santesteban
Universidad Rey Juan Carlos
Madrid, Spain
igor.santesteban@urjc.es

Miguel A. Otaduy
Universidad Rey Juan Carlos
Madrid, Spain
miguel.otaduy@urjc.es

Nils Thuerey
Technical University of Munich
Germany
nils.thuerey@tum.de

Dan Casas
Universidad Rey Juan Carlos
Madrid, Spain
dan.casas@urjc.es

This document presents all the implementation details to reproduce the pipeline shown in Figure 2 from the main paper.

A Per-garment preprocess

In total, we have tested the untangling using 5 garment models from the Berkeley Garment Library [9, 6]. For each garment, independently, we precompute the implicit surface model that estimates the signed distance field and covariant field conditioned to body shape, Table 1 shows the preprocessing cost per garment. Notice that this is a per-garment operation, which does not see multi-garment outfits or any tangled configuration at any time.

	Vertices	Triangles	Cloth simulation	Training (explicit)	Training (implicit)
T-shirt	4424	8710	1h 53min	54s	1h 2min
Top	4306	8454	43min	52s	1h 1min
Tank	3010	5825	1h 11min	50s	54min
Pants	3893	7696	50min	52s	56min
Dress	14297	28168	3h 42min	1min 29s	2h 36min

Table 1: Preprocessing per garment

All our models are implemented in PyTorch, using Adam [3] for training, and a linear learn rate scheduler that reduces the initial learning rate by a factor of 0.001 by the end of the training. Next, we define the aspects that are specific to each model.

A.1 Explicit garment model.

The explicit garment model receives $\beta \in \mathcal{R}^2$ as input and produces the garment deformation $\mathbf{X}(\beta) \in \mathcal{R}^{|V|\times 3}$ in canonical space, were $|V|$ is the number of vertices of the garment mesh. We use the two first coefficients of the SMPL model [4] since these are enough to capture the largest body deformations. To train the regressor $\mathbf{X}(\beta, \theta_{\text{explicit}})$, where θ_{explicit} are the trainable parameters, we minimize the difference w.r.t ground-truth data:

$$\theta_{\text{explicit}} = \arg\min \sum_{\beta} \|\mathbf{X}(\beta, \theta_{\text{explicit}}) - \mathbf{X}_{\text{GT}}(\beta)\|_2^2 \tag{1}$$

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Data. We compute ground-truth garment deformations using ArcSim [6, 5]. To this end, first we sample the space of body shapes by selecting 11 evenly-spaced values in $[-2.5, 2.5]$ and then we simulate the garments for each possible combination of those values. With two shape coefficients, this yields 121 body shapes. To simulate, we use the `gray-interlock` material included in the cloth simulator that models an interlock knit fabric made of 60% cotton and 40% polyester. Since the simulations require a collision-free initial state, we start all the simulations using the mean body shape (for which we have a manually created collision-free configuration) and perform 20 interpolation steps towards the target body shapes. Then we simulate for 200 additional steps or until the garment reaches equilibrium.

Preprocessing. The preprocessing consists on projecting the simulation data to the canonical space proposed by Santesteban *et al.* [7], which removes the influence of body shape in the garment deformation. For example, the height of the avatar introduces a translation in the vertical axis that results in completely different vertex positions, even if the overall deformation remains similar. Removing the influence of body shape is a way of removing this undesired variance in the training data. To make the learning task easier, we also normalize the data per-vertex by subtracting the mean and dividing by the standard deviation.

Network architecture and training. The function $\mathbf{X}(\beta, \theta_{\text{explicit}})$ is modeled as a Multilayer Perceptron (MLP) network with 3 hidden layers of size 256 and ReLU activations. The output layer has size $3|V|$ and after reshaping and denormalizing we obtain the deformed garment $\mathbf{X}(\beta, \theta_{\text{explicit}})$ in canonical space. We train the model for 1000 epochs using batches of size 8 and an initial learning rate of 1e-3.

A.2 Implicit garment model.

The implicit garment model receives the body shape coefficients $\beta \in \mathcal{R}^2$ and a point $x \in \mathcal{R}^3$ and returns the value of $f(x)$ and $h(x)$ (*i.e.*, the signed distance field and the covariant field evaluated at point x). Our goal is to train an implicit model that is consistent with the explicit model for all body shapes.

Data. To generate the ground truth data for the implicit network, at the beginning of each epoch we randomly sample 20 body shapes from $\mathcal{U}(-2.5, 2.5)$, evaluate the explicit garment regressor to obtain the deformed garment surfaces and, for each surface, we compute ground-truth values of $f(x)$, $h(x)$ and their gradients for all the vertices of the surface as well as 3000 points sampled randomly in the volume (in our implementation, the volume is the bounding-box of the garment and the sampling is done uniformly along each axis). In total, the dataset has $20(|V| + 3000)$ samples. Since the cost of computing the dataset is similar to the cost of a training epoch, while the network trains on the GPU we regenerate the dataset on the CPU. This way we can sample the input space exhaustively and enforce consistency between the implicit and explicit models for any body shape, not just the 121 body shapes used to train the deformation regressor.

Network architecture and training. The implicit garment model is implemented as a MLP network with 4 hidden layers of size 256 and ELU activation functions. The input 3D position x is mapped to a higher dimensional space using Fourier features [8]. The mapping is computed as $\gamma(x) = [cos(2\pi\mathbf{B}x), sin(2\pi\mathbf{B}x)]$ where $\mathbf{B}$ is a random Gaussian matrix of size 64×3 whose values are sampled from $\mathcal{N}(0, 2)$. The model is trained for 1000 epochs using batches of size 516 and an initial learning rate of 1e-3. The weight λ of the gradient loss terms is set to 0.1.

B Untangling operator.

The untangling operator is trained once and reused for any combination of up to N garments.

Data. We sample random values of $f^* \in \mathcal{R}^N$ from $\mathcal{U}(-0.2, 1.5)$ and $h^* \in \mathcal{R}^N$ from $\mathcal{U}(-1.0, 1.0)$. For each pair of f^* and h^* we compute ground-truth values of the untangled surfaces f using the method by Buffet *et al.* [1]. In total, the training set of the untangling operator has 1 million samples.

Network architecture. The untangling operator is implemented as a MLP network with 4 hidden layers of size 256 and ELU activation functions. Since the MLP requires a fixed input size, we set N=7 so that it can handle up to 7 garment layers, which is more than enough for common outfits. To untangle outfits with less than N layers we simply set $h = 1$ for all the unused slots. The model is trained for 3000 epochs using batches of size 516 and an initial learning rate of 1e-3.

C Optimization.

We solve the optimization of the untangled garments using Pytorch's implementation of L-BFGS, with the step size set to 1.0, history size to 100, and line search activated for additional robustness (strong Wolfe method). Empirically, we set $\omega = 1e-5$ so that the optimization avoids large triangle distortions without interfering with our main goal of moving the vertices to the untangled surfaces. For the results shown in the paper, we run the optimization until convergence. For the interactive demo, we found that running the optimization for just 4 steps is enough to resolve most collisions and achieve interactive frame rates. We address residual collisions using the rendering solution from [2], which applies a small offset to the depth buffer of the outer layer. This solution only works for very small collisions as large depth offsets introduce very noticeable artifacts.

References

[1] Thomas Buffet, Damien Rohmer, Loic Barthe, Laurence Boissieux, and Marie-Paule Cani. Implicit Untangling: A Robust Solution for Modeling Layered Clothing. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 38(4), 2019. doi:10.1145/3306346.3323010.

[2] Edilson De Aguiar, Leonid Sigal, Adrien Treuille, and Jessica K Hodgins. Stable spaces for real-time clothing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 29(4), 2010. doi:10.1145/1778765.1778843.

[3] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[4] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 34(6):1–16, 2015. doi:10.1145/2816795.2818013.

[5] Rahul Narain, Tobias Pfaff, and James F. O'Brien. Folding and crumpling adaptive sheets. *ACM Trans. Graph.*, 32(4):51:1–51:8, July 2013. doi:10.1145/2461912.2462010.

[6] Rahul Narain, Armin Samii, and James F O'brien. Adaptive Anisotropic Remeshing for Cloth Simulation. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 31(6):1–10, 2012. doi:10.1145/2366145.2366171.

[7] Igor Santesteban, Nils Thuerey, Miguel A Otaduy, and Dan Casas. Self-Supervised Collision Handling via Generative 3D Garment Models for Virtual Try-On. *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2021.

[8] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[9] Huamin Wang, James F O'Brien, and Ravi Ramamoorthi. Data-Driven Elastic Models for Cloth: Modeling and Measurement. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 30(4):1–12, 2011. doi:10.1145/2010324.1964966.