

# Gaussian Garments: Reconstructing Simulation-Ready Clothing with Photo-Realistic Appearance from Multi-View Video

## Supplementary Material

### Abstract

*This document supplements the main paper with details on i) the initial mesh reconstruction process ii) the implementation of the appearance model, iii) the registration process, iv) the behavior optimization, and v) the procedure for the automatic garment ordering. Additionally, we provide vi) additional evaluation of our registration, appearance modeling, and behavior reconstruction procedures. We compare our registration procedure with the SOTA registration method by Lin et al. [9] and our full method to SCARF [1]. Finally, we demonstrate the results of our method in the attached video, by reconstructing a diverse set of garments with various shapes, appearances, and materials. In the video, we also provide registration results in dynamic sequences, comparisons of our method to AnimatableGaussians [8], and evaluation of our behavior fine-tuning procedure.*

### 1. Initial Mesh Reconstruction

The process of reconstructing the garment mesh from single-frame multi-view imagery involves three key steps. First, we select a template frame where the garments are fully visible, from which we construct the dense oriented point cloud of the scene by running a multi-view stereo algorithm in COLMAP [12]. Next, we filter out background points and reconstruct the surface of the clothed human body using Poisson surface reconstruction [5]. Finally, we separate the individual garments from the body using semantic segmentation maps and apply a re-meshing algorithm by [7] to obtain well-defined triangle meshes of the desired resolution for each garment piece. We use 8000 vertices for each garment, which we observe works well with the pre-trained GNN simulator.

Extracting garment templates using semantic segmentation maps can often result in poor-quality mesh boundaries, particularly with low-resolution meshes. During training, 3D Gaussian kernels near the boundaries are pruned and densified to better align with multi-view observations, enabling successful registration. However, in cases where certain camera views are missing or flawed segmentation maps are provided, the expected open boundaries may be erroneously closed with faces, preventing the garment from deforming properly with the videos. Manual intervention may be required to remove these unwanted faces, ensuring the mesh is correctly shaped and capable of deforming accurately with the video content.

### 2. Appearance Details

To model the garment’s appearance, we use a Gaussian texture, i.e., a 2D image with multiple channels containing parameters for 3D Gaussians. To produce a 3D Gaussian Garment, we sample the Gaussians from the texture in a regular grid (e.g. one Gaussian per texture pixel). Note that each garment face may contain multiple Gaussians depending on the face’s texture location. Then, we position the Gaussians in 3D using the sampled parameters. Here we describe this process in detail.

Following the approach of Qian et al. [10], we define a local coordinate system for the Gaussians, which allows us to transform them along with the deforming mesh. The coordinate system for Gaussian  $i$  is defined by rotation matrix  $\mathbf{R}_j \in \text{SO}(3)$  specific to the face  $j$  and the Gaussian surface point  $\boldsymbol{\tau}_i \in \mathbb{R}^3$  as the origin. The coordinate system is unique for each Gaussian. Its basis comprises three unit vectors: the normal vector of the Gaussian’s corresponding triangular face, one of the triangle’s edges, and the cross-product of these two.

Inside the coordinate frame, we represent a Gaussian’s rotation as a quaternion  $\mathbf{r}_i \in \mathbb{H}$ , translational offset  $\boldsymbol{\mu}_i \in \mathbb{R}^3$ , and scale  $\mathbf{s}_i \in \mathbb{R}^3$ . This allows Gaussians to move within their corresponding mesh face, and to capture high-frequency texture details. Additionally, as the mesh deforms, the Gaussians attached to a face  $j$  are affected by the face’s scale,  $k_j \in \mathbb{R}_+$ . This scale is computed as  $\frac{B+H}{2}$ , where  $B$  and  $H$  are the base and height of the triangle.

Then, during rendering the local-frame Gaussians are transformed into world coordinates by the following equations:

$$\mathbf{r}'_i = \mathbf{R}_j \mathbf{r}_i, \quad (1)$$

$$\boldsymbol{\mu}'_i = k_j \mathbf{R}_j \boldsymbol{\mu}_i + \boldsymbol{\tau}_i, \quad (2)$$

$$\mathbf{s}'_i = k_j \mathbf{s}_i. \quad (3)$$

#### 2.1. Appearance Initialization

We initialize the appearance using zeros for all Gaussian parameters, create Gaussians on the mesh surface, and optimize them to match the template frame observations.

The primary optimization term here is the RGB error  $\mathcal{L}_{\text{RGB}}$ . It combines mean absolute error  $\mathcal{L}_1$  and structural similarity error  $\mathcal{L}_{\text{SSIM}}$  between the renders and ground-truth images.

$$\mathcal{L}_{\text{RGB}} = \lambda_{\text{RGB}} \mathcal{L}_1 + (1 - \lambda_{\text{RGB}}) \mathcal{L}_{\text{SSIM}}, \quad (4)$$

where  $\lambda_{RGB}$  is a balancing weight.

Additionally, we incorporate two regularization terms introduced by Qian et al. [10]. The first term,  $\mathcal{L}_{pos}$ , regularizes the Gaussians to stay close to their surface points, defined by their barycentric coordinates.

$$\mathcal{L}_{pos} = \|\max(\mu - \epsilon_{pos}, 0)\|_2, \quad (5)$$

where  $\mu$  are local translations and  $\epsilon_{pos}$  is a tolerance threshold.

The second term,  $\mathcal{L}_{scale}$ , penalizes the scale  $s$  of the Gaussians relative to the underlying mesh triangles.

$$\mathcal{L}_{scale} = \|\max(s - \epsilon_{scale}, 0)\|_2, \quad (6)$$

where  $\epsilon_{scale}$  is a tolerance threshold.

We use the resulting set of Gaussians, rigidly attached to the garment surface, to register the mesh to the multi-view videos.

## 2.2. Appearance Modelling

Many recent works model garment appearance on human avatars as a pose-dependent problem. They directly learn a bijective projection from a specific body pose to a certain garment appearance. However, garments' appearances change dynamically. Wrinkle patterns may vary under the same body pose, and different wrinkles may lead to different occlusions and shadows. Minor 3D structures, like fur, also introduce shifts relative to the garment surface. Therefore, we leverage the Style U-Net [8] architecture to predict appearance changes.

Given the deformed mesh in each frame, we first create ambient occlusion and normal maps using the Blender Python library. We used a texture size of  $512 \times 512$  px<sup>2</sup> in our experiments. These two maps provide the occlusion ratios and surface normal information, which helps to learn the shadows and specular effects on the garment surface. Then, we concatenate these maps along the color channel. We generate ambient occlusion maps separately for both outer and inner garment surfaces to better model their appearance.

The backbone of our model is Style U-Net, a conditional StyleGAN-based [4] generator. During training, the model takes as input the ambient occlusion and normal maps together with a view direction map and predicts offsets to the Gaussian texture. Before the forward process, we convert the normal map directions to the camera coordinate. We find it makes training converge faster with better reflective effects. The view direction map is a tensor with the same shape as the normal map. It contains the normalized directions from the camera position to the origin points, converted to the origin points' local coordinates. All invalid texture pixels, which do not correspond to any point on the surface, are set to zero. The view direction map is first passed through a small CNN with two convolutional layers and then added element-wise to the hidden layer within the Style U-Net.

The output has 51 channels with the same resolution as the input maps. The first 48 channels are offsets to the spherical harmonics, modeling the lighting effects, including shadows and highlights. The last 3 channels are translational offsets, compensating for registration inaccuracies and shifts of minor 3D structures.

## 3. Registration Details

Our registration pipeline uses multi-view images as visual guidance and optimizes Gaussian-bounded mesh positions to register the mesh to successive frames. We leverage the RGB and SSIM loss from 3D Gaussian Splatting [6] and add physical regularization terms to preserve realistic wrinkles caused by highly dynamic movement.

However, in cases where large occlusions are present, e.g., occlusions by adjacent body parts, the RGB and physical regularizations (bending and stretching) alone do not suffice for convergence, and the mesh tends to implode (see Fig. 1). To alleviate this problem, we include a body-garment collision term,  $\mathcal{L}_{body}$ , as a further regularizer that provides a displacement constraint when photometric support is lacking.

Still, in highly dynamic motions, the body-garment penetration at the beginning of the optimization procedure can hinder convergence. Therefore, for the first part of the optimization, we substitute  $\mathcal{L}_{body}$  by a term based on virtual edges,  $\mathcal{L}_{VE}$ , described below.

### 3.1. Virtual Edges Regularization

The garment geometry for each frame is initialized at the last frame's converged position. However, in highly dynamic sequences, the body may move greatly between the frames, resulting in large body-garment penetrations. In these cases, the  $\mathcal{L}_{body}$  regularizer fails to preserve the garment geometry, which tends to implode.

Therefore, we construct "virtual edges" between opposite faces of the garment mesh to prevent the mesh from collapsing onto itself. We identify such "opposite" faces by casting rays along the normal direction of each face and querying for the intersection face. We filter the identified face pairs by only keeping those whose normals are nearly parallel. We compute the following regularization term to prevent the face pairs from getting too close to each other:

$$\mathcal{L}_{VE} = \sum_i \max(L_{e_i} - l_{e_i}, 0)^2, \quad (7)$$

where  $L_{e_i}$  and  $l_{e_i}$  are lengths of the edge  $e_i$  in the template and the current geometries respectively.

We use  $\mathcal{L}_{VE}$  in the first half of the optimization and replace it with  $\mathcal{L}_{body}$  in the second half of the optimization. We observed that this scheduling allows  $\mathcal{L}_{VE}$  to maintain the mesh structure while  $\mathcal{L}_{RGB}$  optimizes the mesh node positions. Using  $\mathcal{L}_{body}$  for the second half of the optimization



allows for more accurate physical draping of the garment on the body, providing the best overall results (please see Table 1 of the main paper).

### 3.2. First Frame Matching

Our dataset consists of multiple multi-view videos of the same scene. For all videos, we start the registration from the same mesh geometry, reconstructed from the template frame. However, the first-frame pose of each video can be very different from the template frame pose in terms of mesh shape and overall position.

Therefore, we reconstruct a sparse full-body point cloud for the first frame of the new sequences. Then, we find the global rotation and translation that roughly align the template full-body geometry with each video’s first frame, by performing an iterative closest point (ICP) algorithm [11] between the point cloud reconstructed from the template frame and the target sequence.

## 4. Behavior Optimization Details

To mimic the real behavior of garments we fine-tune a pre-trained garment simulation GNN from ContourCraft [3]. As outlined in the main paper, the GNN autoregressively predicts accelerations  $\hat{\mathbf{a}}_{t+1}$  for the mesh nodes in each simulation step given their positions  $\mathbf{x}_t$ , velocities  $\mathbf{v}_t$ , material vectors  $\mathbf{m}$ , and canonical edge lengths  $\bar{E}$ :

$$\hat{\mathbf{a}}_{t+1} = g_\psi(\mathbf{x}_t, \mathbf{v}_t, \mathbf{m}, \bar{E}) \quad (8)$$

The geometry for each step is computed by integrating the predicted accelerations into the simulation:

$$\hat{\mathbf{x}}_{t+1} = \mathbf{x}_t + \mathbf{v}_t + \hat{\mathbf{a}}_{t+1} \quad (9)$$

For simplicity, we assume a time difference equal to 1 between successive frames.

Our goal here is to optimize  $\psi$ ,  $\mathbf{m}$ , and  $\bar{E}$  so that our simulations better match the behavior of the registered sequences.  $\psi$  are the parameters of the GNN and  $\mathbf{m}$  are material vectors. These are 4-value vectors attached to each node of the garment mesh and fed into the GNN.  $\bar{E}$  are canonical lengths of each edge in the mesh represented by scalar values. All these elements are parts of the original ContourCraft model that we optimize for our needs.

We tune these parameters using all the registered sequences in our training set. During fine-tuning we autoregressively simulate each training sequence with  $g_\psi$ . In each frame, we use the simulated geometry  $\hat{\mathbf{x}}_{t+1}$  to compute a loss value which comprises two terms:

$$\mathcal{L}_{behavior}(\hat{\mathbf{x}}_{t+1}, \mathbf{x}_{t+1}) = \mathcal{L}_{ccraft}(\hat{\mathbf{x}}_{t+1}, \mathbf{x}_{t+1}) \quad (10)$$

$$+ \lambda \mathcal{L}_2(\hat{\mathbf{x}}_{t+1}, \mathbf{x}_{t+1}), \quad (11)$$

where  $\mathbf{x}_{t+1}$  is the registered geometry for the frame  $t + 1$ , and  $\lambda$  is a balancing weight.  $\mathcal{L}_{ccraft}$  here is the original loss

function from ContourCraft, while  $\mathcal{L}_2$  is a simple mean squared error.

Finetuning the GNN using only registered sequences and  $\mathcal{L}_{behavior}$  enables it to mimic the behavior of the garments. The problem, however, is that our registered sequences only contain individual garments without multi-layer outfits. Because of this, the ContourCraft GNN, which originally could handle multi-layer outfits, starts forgetting how to properly handle multi-layer structures during fine-tuning. To alleviate this issue, we construct a set of multi-layer outfits from our reconstructed garments and use them in every other training iteration instead of the registered individual garments. Since we don’t have target geometries for these outfits, we only supervise these steps with  $\mathcal{L}_{ccraft}$ . This enables the model to both match the real garment behavior and properly handle inter-layer collisions.

## 5. Automatic Garment Ordering Procedure

We use the ContourCraft [3] GNN to devise a simple procedure to automatically untangle and order individual garments.

We start with all garments aligned with the canonical SMPL-X pose and shape. We order the garments by their position in the desired outfit—from the innermost to the outermost. Then we untangle each subsequent garment from the ones that should be beneath it, see Alg. 1.

To untangle a single garment we run two consecutive simulation stages. In the first one, we treat all the inner garments as solid bodies. This way, ContourCraft treats them as body geometry and pushes the outer garment outside them. Then, in the second stage, we simulate all garments, treating them as cloth. We repeat this procedure twice. See Alg. 2

The whole process takes around 1 minute for each garment on an NVIDIA GeForce 4090 GPU.

---

**ALGORITHM 1:** *UntangleAll*; we untangle a sequence of garments one by one from the innermost to the outermost.

---

**Input:** Garment geometries  $G_1 \dots G_N$  ordered from the innermost to the outermost

**for**  $i \in [2 \dots N]$  **do**

$G_{outer} \leftarrow G_i$

$G_{inner} \leftarrow [G_1 \dots G_{i-1}]$

Untangle( $G_{outer}, G_{inner}$ )

**return**  $G_1 \dots G_N$

---

## 6. Additional Evaluation

### 6.1. Registration

We present qualitative examples of our registration ablations in Fig. 1. When using only the RGB loss (“*Only-RGB*”) the garment geometry diverges within a few steps. This is because optimizing 3D geometry using solely the RGB

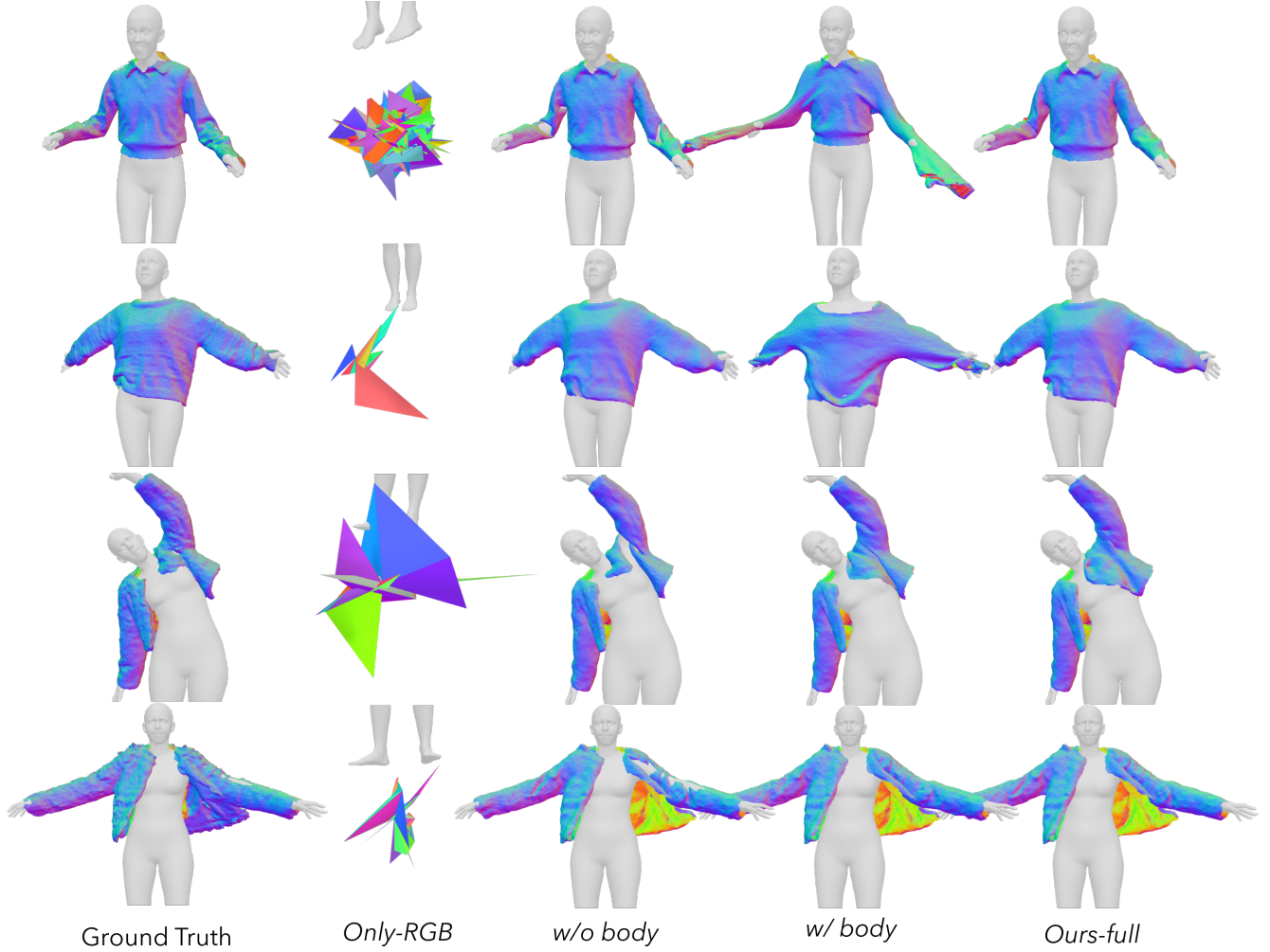


Figure 1. Qualitative comparison of our full registration algorithm and the ablations. When only optimizing the RGB loss (*only-RGB*), the optimization diverges completely. With physical losses (*w/o body*) the garment preserves its structure but does not always conform to the body. When using the body penetration term (*w/ body*), the optimization is prone to artifacts caused by the incorrect initialization. With our full pipeline (*Ours-full*) we first pull the garment geometry closer to the body pose and then enable the body penetration term.

---

**ALGORITHM 2:** *Untangle*; to untangle a single garment, we first simulate it over inner ones treating the latter as solid bodies. Then we re-simulate all the garments together as cloth. We set  $N_{epochs}$  to 2

---

**Input:** Outer garment  $G_{outer}$ ; set of inner garments

```

 $G_{inner}$ 
for  $i \in [1..N_{epochs}]$  do
   $G_{cloth} \leftarrow G_{outer}$ 
   $G_{solid} \leftarrow G_{inner}$ 
   $\text{Simulate}(G_{cloth}, G_{solid})$ 
   $G_{cloth} \leftarrow G_{inner} + \{G_{outer}\}$ 
   $\text{Simulate}(G_{cloth}, \emptyset)$ 
return  $G_{outer}, G_{inner}$ 

```

---

loss is an ill-posed problem, especially with monochromatic objects, like many garments in our dataset. Moreover, the

renders that use the base Gaussian texture may not exactly match GT frames, resulting in noisy signals that accumulate over several frames and lead to diverging results. Introducing the physics losses without an underlying body geometry (“*w/o body*”) has a regularizing effect preventing physically implausible results. However, large body–garment penetrations occur. Naïvely penalizing body–garment collisions (“*w/ body*”) does not allow for robust optimization because the collision computation cannot handle fast movements due to bad initialization. For instance, if a hand goes through the sleeve between time frames, the body collision term will push the sleeve outside the body instead of pulling it back on. Therefore, we demonstrate that our full model (“*Ours-full*”) works best for all pose sequences.

We also compare to a state-of-the-art method for garment registration by Lin et al. [9] (“*Lin2023*”), using 13 garments

Table 1. Comparison between our registration stage and Lin et al. [9]. Our method only uses multi-view RGB images as supervision, whereas [9] directly optimizes a template mesh to fit GT scans.

	F-score, % $\uparrow$	CD, cm $\downarrow$	p2m, cm $\downarrow$	$\mathcal{L}_{body}$ $\downarrow$
[9]	<b>98.7</b>	<b>0.399</b>	<b>0.134</b>	4.60e-5
Ours	90.4	1.001	0.486	<b>1.24e-5</b>

from the 4D-Dress dataset. While our method relies only on multi-view observations from RGB cameras, [9] fit the garment template to the same GT scans as used for evaluation. Given this, our registration procedure performs only slightly worse than [9] (see Table 1). Our method performs only 0.6 cm worse in terms of Chamfer Distance (CD) and 0.2 cm worse in point-to-mesh distance. Meanwhile, the scan data in 4D-Dress dataset usually contains outlier faces, e.g., closed dress bottoms or duplicate layers on two sides of an open jacket. Given the large data volume, removing all erroneous structures from the ground-truth data is difficult. As a result, [9] overfits these artifacts leading to faulty geometries (Fig. 7).

## 6.2. Appearance

In Fig. 3 we show a visual comparison of our final appearance model to the set of ablations described in the main paper. On top of this, we also compare our method to SCARF [2]. SCARF is a NeRF-based method that reconstructs an articulated garment radiance field from a *monocular* video. While SCARF is not a direct baseline to our method due to it using only monocular data, it is the closest method to ours which has publically available code. For this comparison, we use four outfits created from individual Gaussian garments that match the outfit of each subject (see Fig. 2). To optimize the SCARF model we concatenate training frames from different videos and different cameras and treat them as monocular videos. We find that if optimized over frames from all videos and all cameras, SCARF produces extremely blurry results due to data stochasticity. We call the models optimized over all frames “*SCARF-all-frames*”. We also optimize SCARF models over only 500 frames from our videos, making sure they cover the whole body surface in diverse poses. We call such models “*SCARF-500-frames*”. The models optimized over 500 frames produce much crisper results but still do not match the ground truth as well as those of Gaussian Garments. Please see Fig. 2 for visual comparison and Table 2 for quantitative evaluation.

## 6.3. Behavior

Here we evaluate the efficiency of our behavior reconstruction procedure. To do that, we fine-tune the garment modeling GNN from [3] and optimize per-vertex material vectors together with rest geometries over the training sequences. We then simulate the garments for the held-out sequences and compare them to garment registrations obtained by our

Table 2. Quantitative comparison of our method against SCARF. Gaussian Garments’ appearance model and fine-tuned garment simulation GNN allow it to produce high-quality visuals that align with ground-truth observations.

	LPIS $\downarrow$	SSIM $\uparrow$	PSNR $\uparrow$
<i>SCARF-all-frames</i>	7.65e-2	0.884	37.3
<i>SCARF-500-frames</i>	6.78e-2	0.928	39.2
Ours	<b>4.80e-2</b>	<b>0.951</b>	<b>41.5</b>

Table 3. Quantitative evaluation of our behavior-tuning procedure. We compare sequences simulated by the GNN to the registered sequences using the L2 loss term. By fine-tuning the garment simulation GNN, our method can match the behavior of the registered and ground-truth meshes more closely.

	L2 $\downarrow$
<i>default</i>	5.43e-2
<i>tuned-leave-one-out</i>	4.38e-2
<i>tuned-together</i>	<b>4.34e-2</b>

approach using the mean L2 distance between the simulated and registered vertex positions.

In Table 3 we compare the “*default*” untuned GNN to two tuned variants. In both variants, we optimize the GNN parameters  $\psi$  together with the material vectors  $\mathbf{m}$  and rest edge lengths  $\bar{E}$  for the garments. We call the latter two “garment parameters”. In “*tuned-together*” we optimize the network parameters and the garment parameters for all 15 garments together and then run an evaluation on the validation sequences. Then, we use the “*tuned-leave-one-out*” variant to demonstrate how a fine-tuned GNN can generalize to garments that are not in the original fine-tuning set. Here we finetune a separate model for each garment in two stages. In the first stage, we optimize the model parameters and garment parameters for all garments except one. In the second stage, we freeze the model parameters and only optimize the garment parameters for the remaining unseen garments. This results in 15 models—one for each garment. We evaluate each model using the validation sequence for the remaining left-out garment. As seen from Table 3, models from “*tuned-leave-one-out*” perform only slightly worse than the one from “*tuned-together*”. Hence, we can expect reasonable results for novel garments without fine-tuning the GNN parameters again.

## 6.4. Applications

We demonstrate results in the following applications: simulating the garments in novel and dynamic poses, mixing and matching, and dynamic resizing.

In Fig. 5 we show a qualitative comparison of our method to AnimatableGaussians [8] for novel and dynamic pose sequences. Our method manages to realistically capture garment motions in dynamic scenes.





Figure 2. Visual comparison of our method to SCARF [1]. The appearance model and a fine-tuned garment simulation GNN enable Gaussian Garments to produce visually appealing results and better model garment dynamics. The body meshes shown above are included for visualization only and were not used in the quantitative evaluation in Table 2. SCARF also optimizes offsets to the body geometry resulting in slightly different body models compared to the original SMPL-X used by Gaussian Garments.

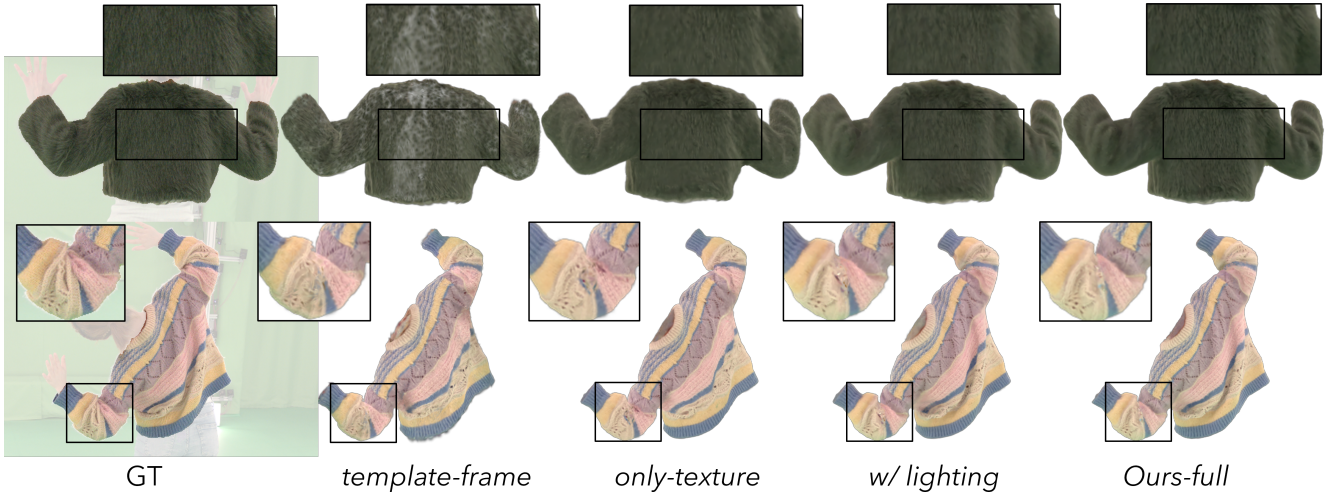


Figure 3. Qualitative comparison of our full appearance model to a sequence of ablations. Note how our full model preserves more high-frequency details and does not contain lighting artifacts.

We further demonstrate garment mix-and-match in Fig. 6, where we combine garments from two (top) and three (bottom) different subjects, and automatically resize them to fit diverse body shapes.

Additional results and animated sequences are provided in the supplementary video.

## 6.5. Reconstruction time

We reconstruct each garment separately. In our experiments, we use 1050 multi-view frames with 44 camera views to reconstruct each garment. Our registration and appearance optimization procedures take roughly 36 hours on an NVIDIA GeForce RTX 2080 Ti. Specifically, it takes 3.5 hours to register each sequence (on average 150 frames) and 24.5 hours for all sequences. Afterward, it takes 1.5 hours to create ambient occlusion and normal maps in Blender, and 10 hours to



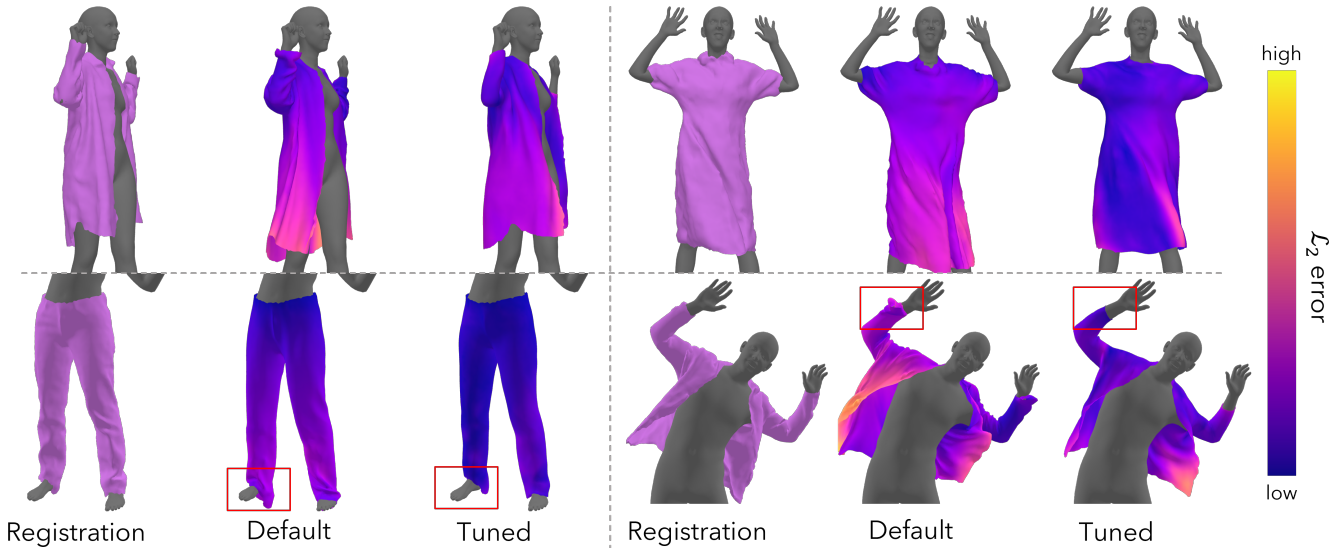


Figure 4. Visual comparisons of the simulations produced by a default pre-trained ContourCraft [3] model and our fine-tuned version. A brighter color denotes a higher L2 error between the simulation and the registered mesh. The fine-tuned model achieves behavior that better matches the registered sequences. By optimizing the rest geometries of the garments we also better match the original size of the garments (bottom left) and avoid simulation artifacts (bottom right).

train appearance models with 5 epochs on 44 camera view data (46200 images in total). It takes an additional 20 hours for the behavior finetuning stage.

## References

- [1] Yao Feng, Jinlong Yang, Marc Pollefeys, Michael J Black, and Timo Bolkart. Capturing and animation of body and clothing from monocular video. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022. 1, 6
- [2] Yutao Feng, Xiang Feng, Yintong Shang, Ying Jiang, Chang Yu, Zeshun Zong, Tianjia Shao, Hongzhi Wu, Kun Zhou, Chenfanfu Jiang, et al. Gaussian splashing: Dynamic fluid synthesis with gaussian splatting. *arXiv preprint arXiv:2401.15318*, 2024. 5
- [3] Artur Grigorev, Giorgio Becherini, Michael Black, Otmar Hilliges, and Bernhard Thomaszewski. Contourcraft: Learning to resolve intersections in neural multi-garment simulations. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–10, 2024. 3, 5, 7
- [4] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405, 2018. 2
- [5] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. *Proceedings of the fourth Eurographics symposium on Geometry processing*, 7:61–70, 2006. 1
- [6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 2
- [7] Bruno Lévy and Nicolas Bonneel. Variational anisotropic surface meshing with voronoi parallel linear enumeration. In *Proceedings of the 21st international meshing roundtable*, pages 349–366. Springer, 2013. 1
- [8] Zhe Li, Zerong Zheng, Lizhen Wang, and Yebin Liu. Animatable gaussians: Learning pose-dependent gaussian maps for high-fidelity human avatar modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19711–19722, 2024. 1, 2, 5, 8
- [9] Siyou Lin, Boyao Zhou, Zerong Zheng, Hongwen Zhang, and Yebin Liu. Leveraging intrinsic properties for non-rigid garment alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14485–14496, 2023. 1, 4, 5, 9
- [10] Shenhan Qian, Tobias Kirschstein, Liam Schoneveld, Davide Davoli, Simon Giebenhain, and Matthias Nießner. Gaussiana-vatars: Photorealistic head avatars with rigged 3d gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20299–20309, 2024. 1, 2
- [11] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001. 3
- [12] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

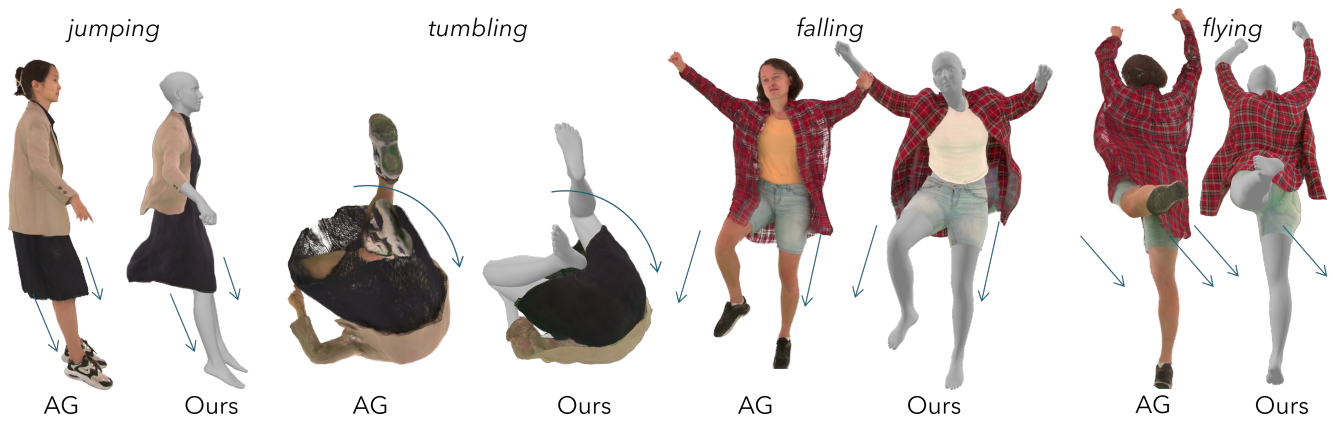


Figure 5. Qualitative comparison of our method to Animatable Gaussians [8] (“AG”). Combining 3DGS with a mesh-based representation, Gaussian Garments are much more robust in simulating challenging poses. With learned cloth simulation, we can also more faithfully model dynamic motions.

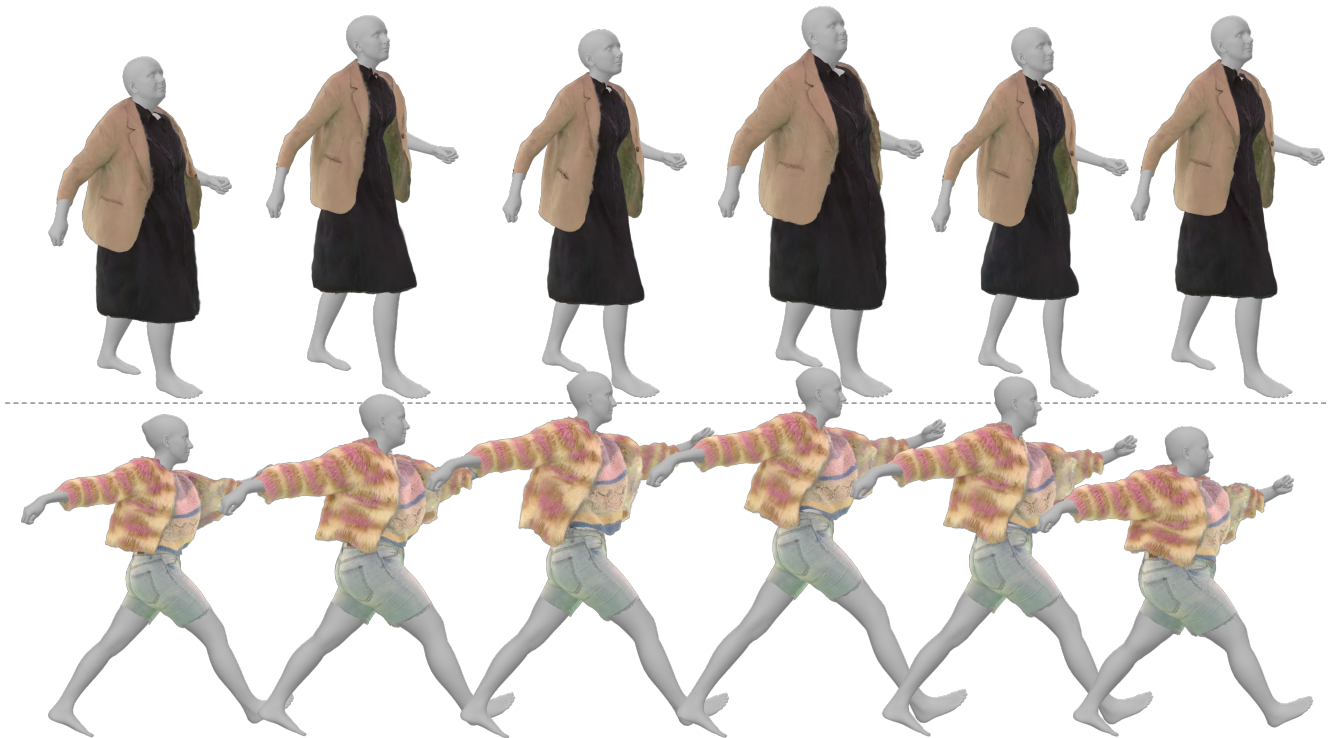


Figure 6. We can automatically resize the Gaussian Garments to fit the desired body shape. Here we randomly sample shape parameters for the parametric body model and render the same outfit for different shapes.

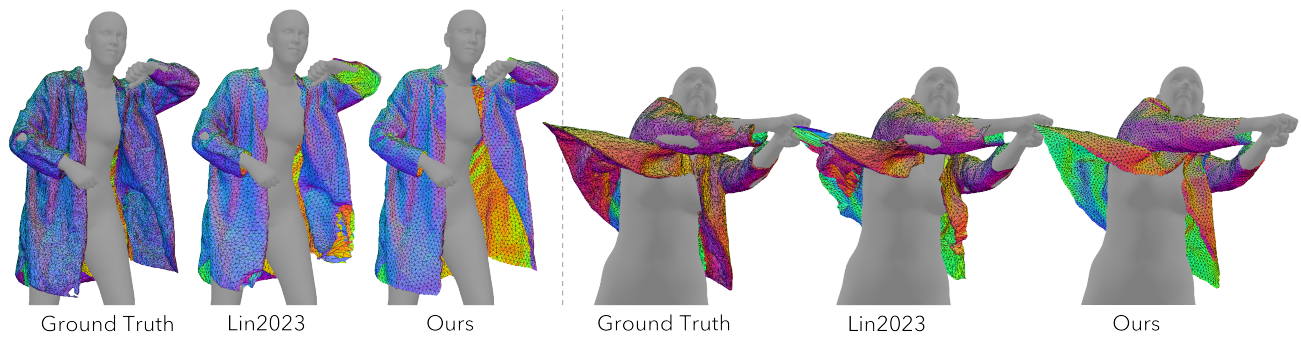


Figure 7. Qualitative comparison of our method to Lin et al. [9] (“*Lin2023*”). Since Lin et al. register garments to ground-truth scans, it may overfit the artifacts present in these scans. In contrast, our registration procedure only uses multiview RGB videos and produces physically realistic geometries.