A Generative Multi-Resolution Pyramid and Normal-Conditioning 3D Cloth Draping
Supplementary Material

Hunor Laczkó[*,†] Meysam Madadi[†,‡] Sergio Escalera[†,‡] Jordi Gonzalez[*,†]
hunor.laczko@uab.cat, mmadadi@cvc.uab.es, sescalera@ub.edu, jordi.gonzalez@uab.cat
[*]Universitat Autònoma de Barcelona, [†]Computer Vision Center, [‡] Universitat de Barcelona
Barcelona, Spain

In this supplementary material, we provide further details on the architecture (Sec. 1), training (Sec. 2), building template UV coordinates (Sec. 3), and additional qualitative results (Sec. 4).

1. Architecture[1]

1.1. Baseline

As we showed in Fig. 1 in the main paper, VAE_{drape} is a conditional VAE based on ConvNext [1]. ConvNext is made up of stages, where after each stage the image or its representation is downscaled. Each stage can have a different depth, meaning the number of blocks it contains. We empirically update the number of blocks and features at each level. In this case, the depths are 3, 3, 3, 9, and 3 with 32, 64, 128, 256, and 128 $(= F)$ features, respectively. The baseline receives an input of resolution 512×256 and the encoder ConvNext has five stages, this way the output features will have a size $8 \times 4 \times F$. The condition encoders work similarly. The dimensionality F is set to 128, 16, 32, and 80 for the VAE_{drape} latent vector, template garment, body, and normals condition encoders, respectively. The combined dimensionality of condition encoders is 128 to balance between conditioning features and VAE_{drape} latent vector. We also use a tanh activation on the conditioning latent codes to ensure we have the same scale in the feature space. Finally, all the latent codes are concatenated before feeding the decoder. For the decoder, we mirror the encoder architecture and replace the downsampling step with upsampling using transposed convolutions. The output size is the original 512×256 resolution. We use tanh activation on the last decoder layer.

1.2. Conditioning autoencoders

We explained conditioning encoders in the baseline architecture. However, these encoders are part of their own autoencoders, pretrained and frozen. That means one autoencoder is used to train each of the three conditioning inputs: template garment, posed body, and normal map. The encoder parts of these networks will act as the condition encoders. When incorporating them in the main pipeline, their weights are frozen to ensure that the quality of the features remains intact. We use tanh activation on the last decoder layer.

The conditioning network for the normal map is the VAE_{norm}. The reason for this design is that normals are not available at inference time. Therefore, we create a generative pipeline for them such that we can sample from it at inference time. VAE_{norm} itself is conditioned on the template garment and posed body UV images. This is because we want the normal features to be disentangled from the other conditioning variables.

1.3. Pyramid

The pyramid model is built using several VAE_{drape} models of varying image resolutions. The main difference between them is the number of stages, which have to be decreased for lower levels of the pyramid. As such we use 5, 4, 3, and 2 stages for the resolutions 512×256, 256×128, 128×64, and 64×32, respectively. Similarly, we decrease the kernel sizes of the convolutions and use kernel sizes of 7, 5, 5, and 3 for the mentioned resolutions. Note that we keep the same feature dimension of $8 \times 4 \times 128$ for all the levels' latent codes.

The first level provides the base for the prediction which is a low-resolution UV image and all subsequent layers are added on top of this as offsets. To achieve this, the first level receives the low-resolution ground truth as input, while later layers receive the offset between the ground truth and the previous level's output along the conditioning inputs. The output's resolution matches that of the input. In order to be able to add the offsets from the next level, this output needs to be upscaled. For this, we use a custom model to upscale

[1]We commit to releasing the code upon the paper's acceptance.

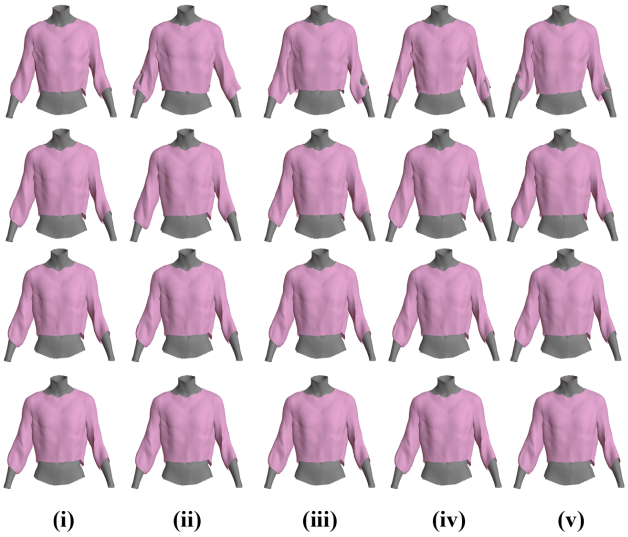

Figure 1. We show the pyramid sampling at each level, one level per row. The first element of each row is the base for the next row.

the UV images (see sec. 1.4). Repeating this at each level, we double the resolution after each level until we reach the final 512×256 resolution.

1.4. Upscaling model

As we mentioned in the main paper we design an upscaling model to deal with the values on the garment mask boundaries within the pyramid network. This model contains a network of 5 convolutional layers with relu activation, batch normalization, and 256, 128, 64, 32, and 3 feature sizes, respectively, and the kernel size follows the corresponding pyramid-level kernel size. The last convolution has a tanh activation instead of relu to predict the 3D coordinates. The upscaling procedure is as follows. First, the image is passed to the above network. This outputs an image that is multiplied with the background mask (inverse of the garment mask) and added to the input image. Finally, we resize this combined image using bilinear interpolation.

2. Training

Due to the number of modules to be trained, we first apply incremental training in which each sub-module is trained independently. Later we will try end-to-end training of the levels (conditioning encoders are still frozen). Next, we will present these sub-modules individually.

2.1. Conditiong autoencoders

The initial set of modules encodes the conditional inputs of the pipeline (as explained in sec. 1.2). We train these networks once on the highest resolution. We use Multiscale-SSIM [2] loss for the autoencoders, and add an extra KL

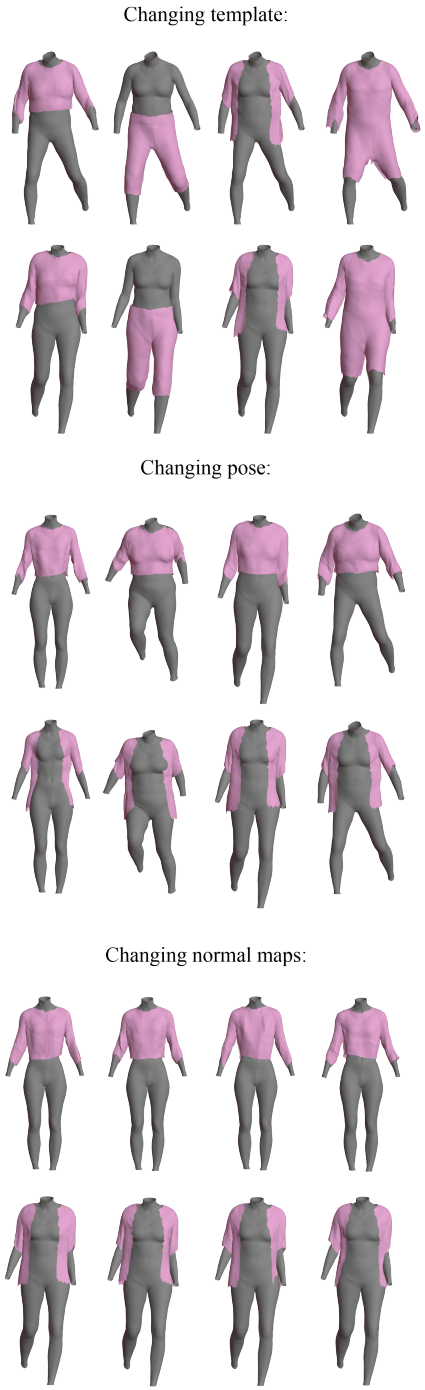

Figure 2. We show the sampling process when changing only one condition. In each case, the stated condition is changing, and the other two and the latent code are fixed. For each case we have two examples as the two rows.

loss (using a β-VAE strategy) when training VAE_{norm}. Finally, we optimize the networks with a batch size of 4 and

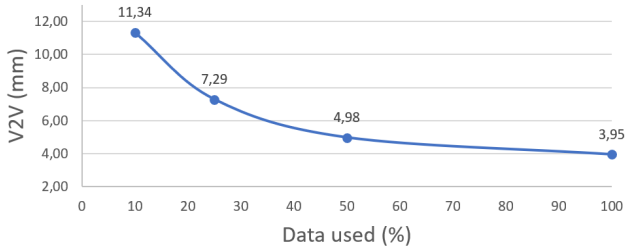

Figure 3. Generalization capability of our pyramid network on CLOTH3D dataset as a function of the amount of training data used.

Adam optimizer with a learning rate 10^{-4}.

After the training, we are able to achieve a reconstruction error of 1.1mm with the posed body and 9.3mm with the template garment autoencoders. Note that the available data for the template garments is only 1.6 thousand outfits which are further split into training and validation data. As for the VAE_{norm}, we achieve an MS-SSIM value of 0.993 with an L1 error of 0.04.

2.2. Pyramid

In the pyramid pipeline, each level is trained individually in the incremental approach and together in the end-to-end approach. The advantages of both cases are detailed in the main paper Sec. 3.1.1. During training, each level receives the same conditional features generated from the high-resolution conditional input, thereby avoiding the need to train three encoders per level and generate separate conditions for each level. This has significant speed improvements. For the variational encoders, each condition and the unposed garment are downscaled to the given resolution. The decoder outputs an image at the same resolution which is upscaled to be added to the next level's output.

The losses used are described in detail in the main paper. Note that we only start using the normal loss after 2000 iterations, since otherwise, it makes the initial steps of the training highly unstable.

We optimize the network with a batch size of 4 and Adam optimizer with a learning rate 10^{-4}.

3. Template UV coordinates

A template UV map is obtained by projecting a three-dimensional body mesh to a fixed two-dimensional shape. To construct this mapping, we start by annotating the body mesh vertices according to the front and back of the body. This is followed by defining some keypoints: 1) around the ends of legs and arms, 2) around the neck, and 3) bilaterally for knees, hips, and underarms. We also create a low-resolution 2D mesh as a template for the UV map. We define pairs of previously selected key points and corresponding points on the UV map template. Using these control points we apply thin plate spline interpolation (TPS) to morph the 3D vertices to the 2D template (a 3D flat surface) once for the front and then for the back vertices. The vertices along the border between the front and back belong to both parts, so they have to be duplicated. During the 3D reconstruction, these duplicated pixels will be averaged to obtain the 3D coordinate.

To further improve the construction of UV maps, we define more keypoints automatically. We take the vertices of the previously mentioned border and find the closest vertex on the 2D template. This vertex is obtained by calculating distances between the points and also from their normal vectors, the latter ensuring that the mesh is stretched outwards from the center to the template. With these new keypoint pairs, we apply TPS again. Finally, we obtain UV mapping that maximizes the data representation by covering a large portion of the given rectangular grid (as shown in Fig. 2(b) in the main paper).

4. Results

4.1. Pyramid sampling

We demonstrate how the sampling process works in the pyramid architecture. Given a set of conditioning variables, which are the template garment, posed body, and normal maps, we sample the VAE latent space, concatenate the encoded conditions, and pass it to the decoder. When we do this at the first pyramid level, we get some potential base garments as seen in Fig. 1. After this, we sample from the second level and generate the offsets over the first level, which adds some smaller details. The contribution of each level can be seen in the figure. In the case shown, after the second level, the changes are minimal. To further demonstrate the sampling possibilities, we show examples of only one of the conditions changing to better demonstrate how it works. This can be seen in Fig. 2.

4.2. Normal map sampling

The aim of the normals is to provide guidance and variability for the generation of samples. We study the VAE_{norm} model's ability to generate normal maps. We can see some examples in Fig. 4 which show how we can generate multiple plausible normal maps for a given input.

4.3. Generalization

To show the generalization and representation capacity of our model we evaluate how the error scales when using a smaller portion of the training data. We run the training with the following portions of the data: 0.5, 0.25, and 0.1[2]. As seen in the error trend in Fig. 3, we can achieve state-of-the-art with only 10 percent of the data. It also shows we

[2]Note these portions are not applied on the whole CLOTH3D dataset but the 20% we initially selected in this paper.

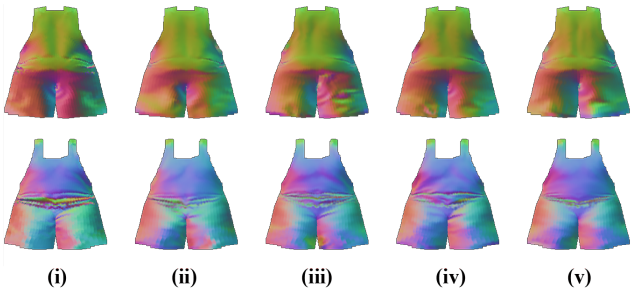

Figure 4. Examples of sampling normal maps. The left (i) is the original, and the four on the right (ii-v) are randomly sampled from the VAE_{norm} model. We can see high variance while they still represent realistic deformations.

can scale the method by increasing the amount of data, but the improvements will diminish soon.

4.4. Additional example results

We provide some additional qualitative examples from the CLOTH3D dataset in Fig. 5 and Fig. 6. Our model is able to reconstruct a high level of detail on a variety of different garments. We also show some additional comparison with HOOD in Fig. 7.

References

[1] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022. 1

[2] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402 Vol.2, 2003. 2

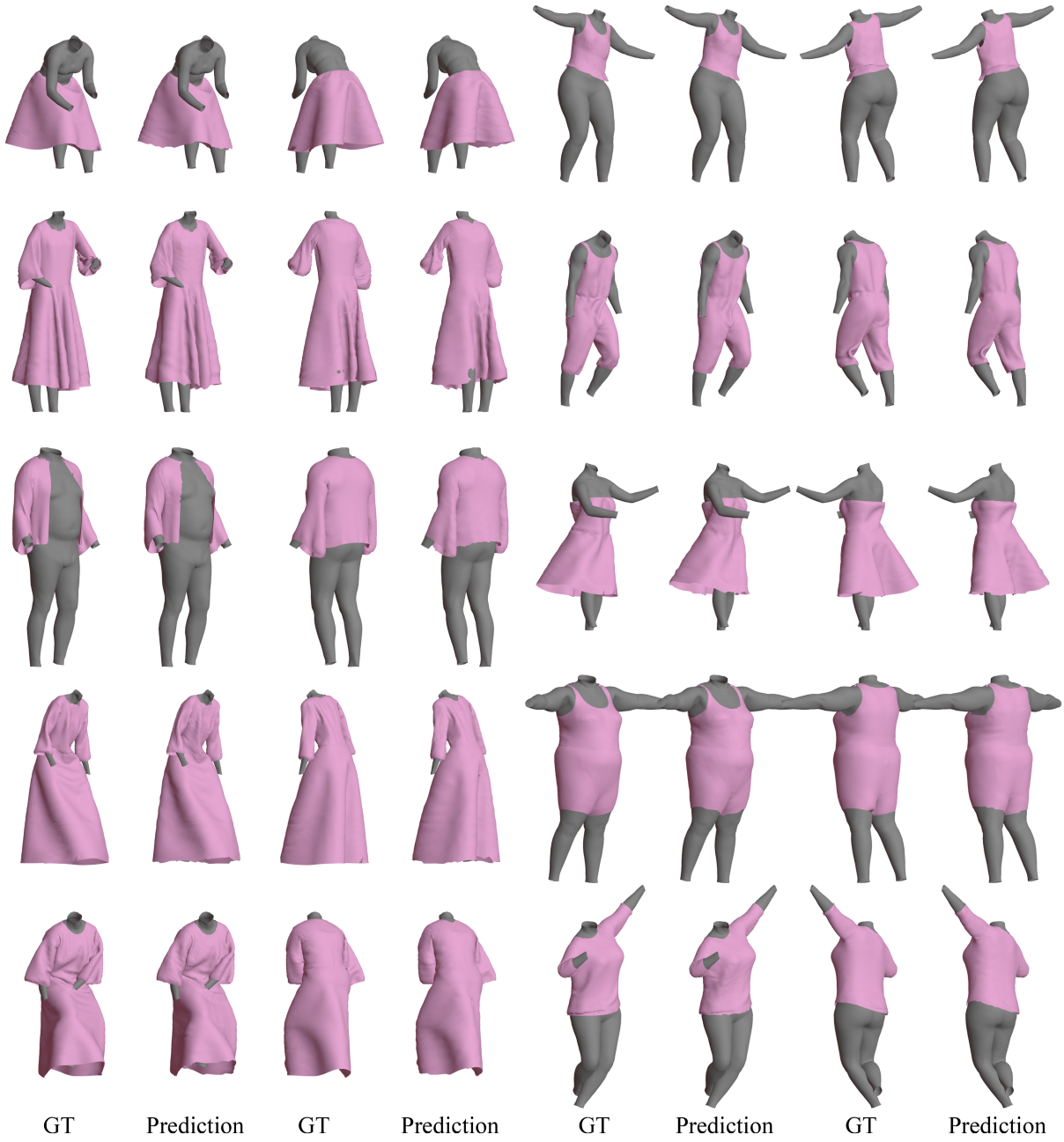

Figure 5. Additional qualitative examples from the CLOTH3D dataset: ground truth and proposed model's reconstruction.

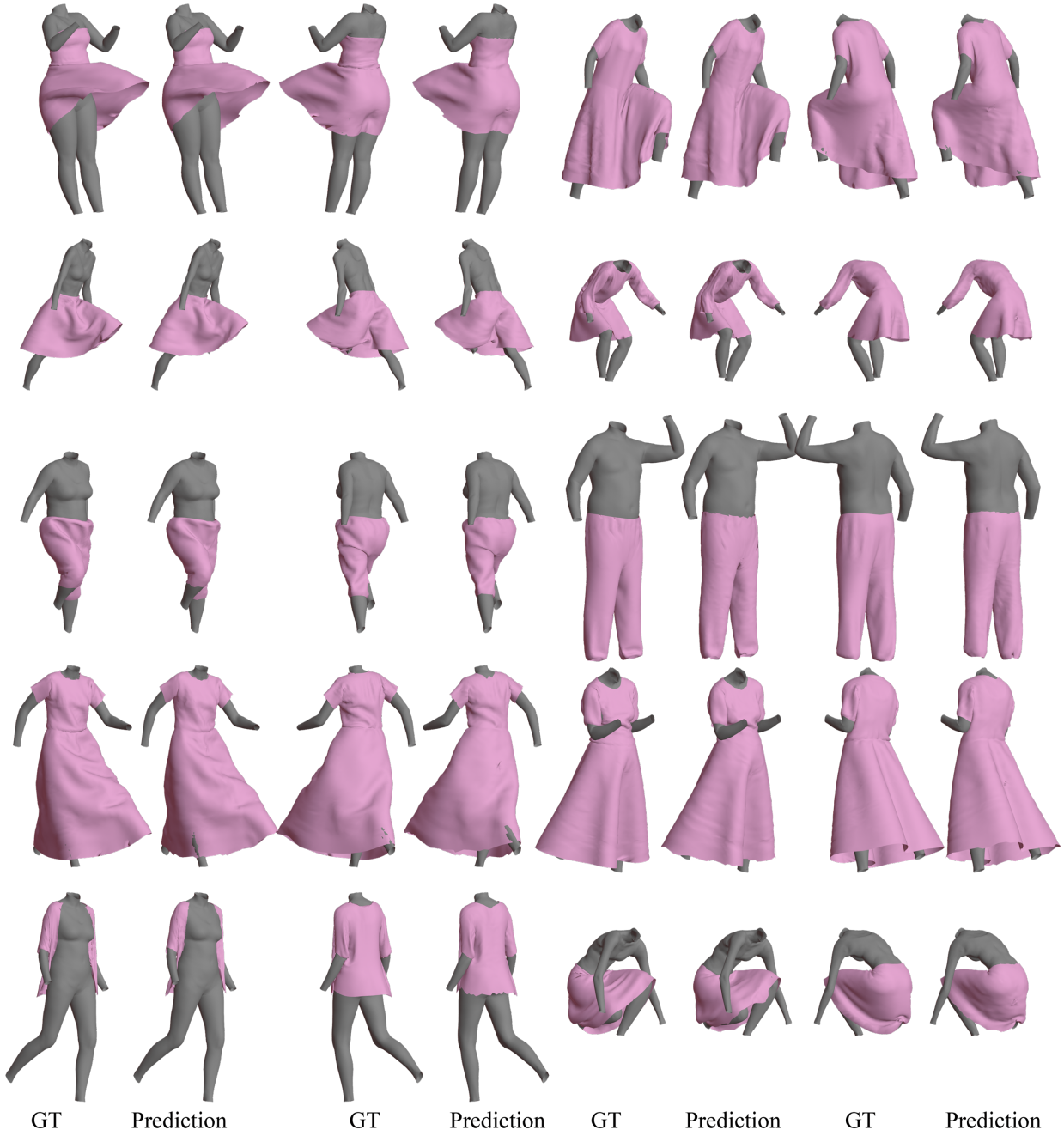

Figure 6. Additional qualitative examples from the CLOTH3D dataset: ground truth and proposed model's reconstruction.

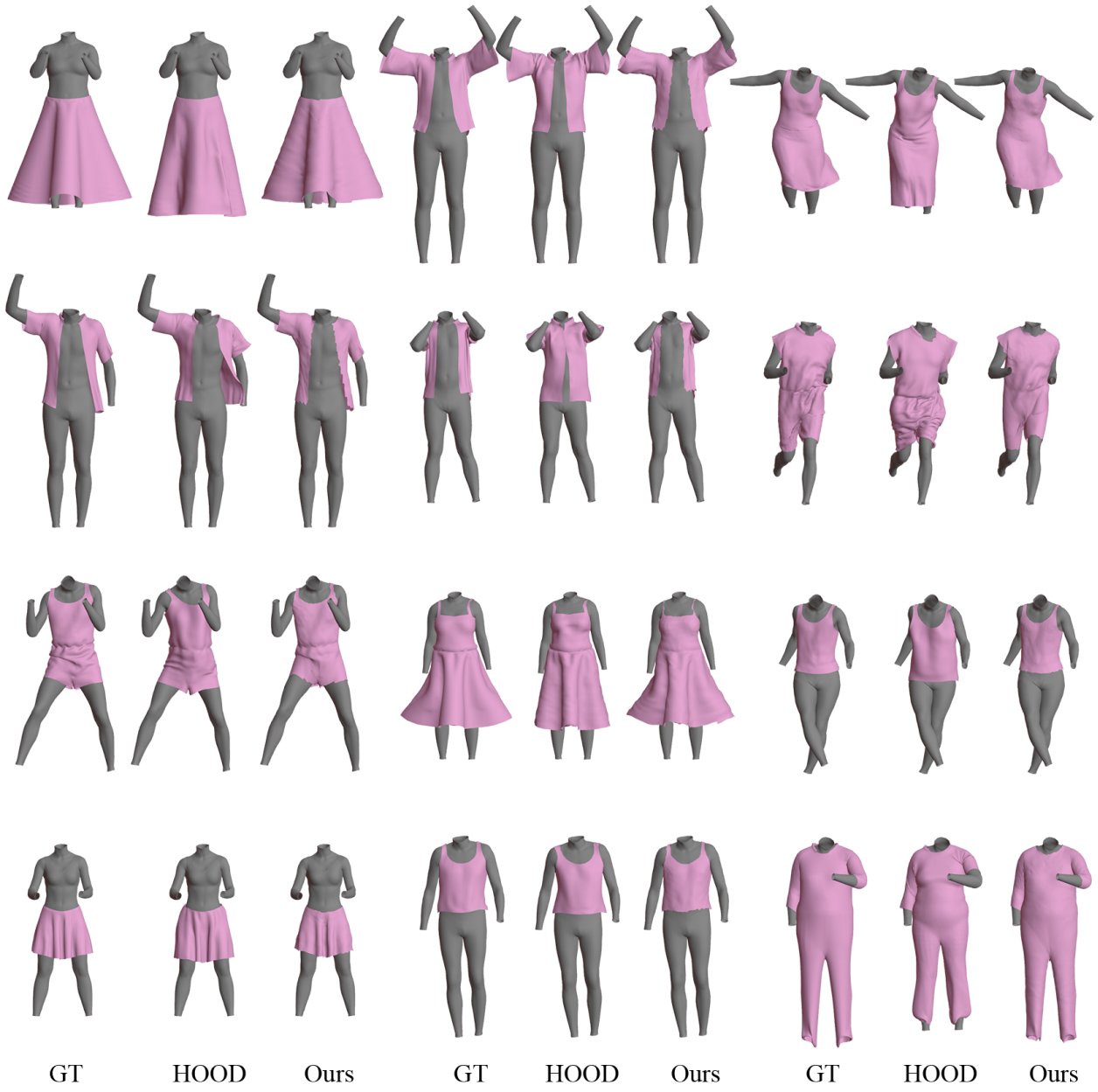

Figure 7. Additional qualitative comparison examples with HOOD, where we can see the ground truth, HOOD prediction, and ours.