Supplementary Material: 360-Degree Textures of People in Clothing from a Single Image

Verica Lazova Eldar Insafutdinov Gerard Pons-Moll

Max Planck Institute for Informatics
Saarland Informatics Campus, Germany
{vlazova, eldar, gpons}@mpi-inf.mpg.de

1. Obtaining ground-truth texture maps

After we have registered SMPL to the scan, we remap the texture map of the scan to the SMPL UV-template as shown in Figure 1. For this purpose we use a nearest neighbor approach. For every point on the registration, we find the closest point on the scan. We check where this point maps on the scan's UV map and we copy the color from this location to the corresponding location on the registration's UV map. After this step we have a full texture map remapped to the SMPL template.

2. Architecture and Training

For all three stages of our method: segmentation completion, texture completion and displacements prediction we use the same type of architecture. The generator has an encoder-decoder architecture with bottleneck of residual blocks, as shown in Figure 2. Skip connections are also used between the encoder and the decoder. These are more useful for the segmentation inpainting task where they can be used to propagate the input segmentation to the output. Detailed per-layer information regarding the generator is given in Table 1. The discriminator is a fully-convolutional patch-discriminator and detailed information regarding it's architecture is shown in Table 2.

The input for the generator in the texture inpainting case is 256×256 partial texture map, extracted using Denspose [2] and normalized to a range $[-1, 1]$. Additionally the partial texture map is concatenated with Gaussian noise tensor $\boldsymbol{N} \in \mathbb{R}^{256\times256\times3}$, $\boldsymbol{N} \sim \mathcal{N}(0, 1)$. The same setting is also adopted for the segmentation completion task, where the input is partial segmentation; and for the displacement prediction where the input is full segmentation map.

The discriminator and the generator are trained together for 100 epochs, with batch size of 8, using the Adam optimizer [4], with learning rate: 0.001, and parameters $\beta_1 = 0$ and $\beta_2 = 0.9$. We follow the training process proposed in [3]. The discriminator is trained more than the generator: for every optimization step for the generator we make 5 optimization steps for the discriminator. For training the discriminator we use additional gradient penalty term as introduced in [3]. The same settings are used for all three stages: texture completion, segmentation completion and displacements prediction.

2.1. Garment editing experiment

For editing the length of the sleeves and pants of the person we have trained a separate neural network. We train this network only with the patches from the texture map that contain the arms and the legs of the person. The patches are upsampled to size 256×256 and then they are partially masked out. We train separate generators and discriminator for both patches. The generators take the masked patch concatenated with it's segmentation and Gaussian noise as input. They are trained to complete the patches such that they look realistic and match the given segmentation. The whole editing process is shown in Figure 3. The new segmentation is also used to generate new displacement map, which when applied to the average SMPL model, textured with the edited texture map, yields the new 3D avatar.

3. Loss Functions

The effect of the different loss functions used during training for the texture inpainting task, is displayed in Figure 4. The input image is shown in the first column, and the rest of the columns show textures inpainted by models trained with different losses. All the textures are displayed on the average naked SMPL [6] body. The second column shows results from model trained with L_1 loss only; the third column: model trained with L_1

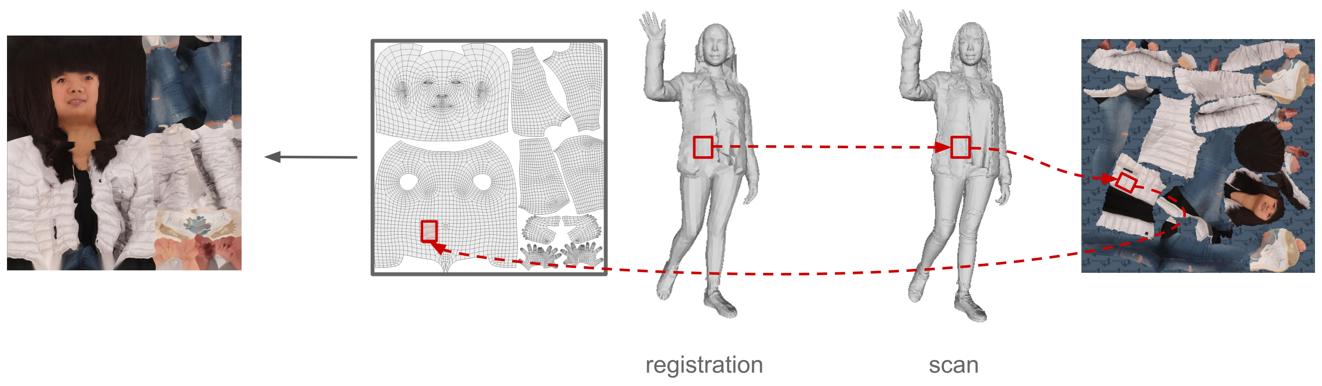

Figure 1: Remapping the texture map of the scan to the SMPL UV-template using nearest neighbor approach.

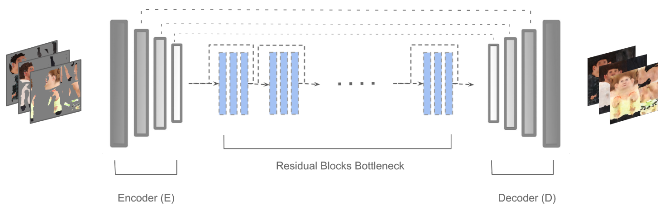

Figure 2: The residual-block-architecture of the generator.

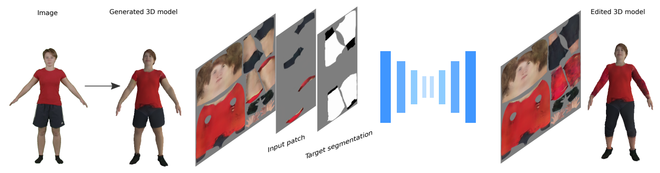

Figure 3: Garment editing pipeline: Given a completed texture map, arms and legs patches are extracted, partially masked and together with a target segmentation (black for skin, white for clothing) are fed to the generator to produce the new texture that match the segmentation.

reconstruction and GAN loss; the model from the fourth column additionally relies on the perceptual loss (features extracted from pretrained VGG-19) and the model from the last column additionally uses a structural dissimilarity loss. As expected, a large gain is obtained when adding the adversarial loss, without it the inpainted texture is blurry even for the visible parts (Figure 4, rows 5 and 8). The improvements from leveraging the perceptual and the structural dissimilarity loss are more subtle but contribute to more visually pleasing results. The weights for each of the loss functions during training are set as follows: $\lambda_1 = 1$, $\lambda_2 = 1$, $\lambda_3 = 10$, $\lambda_4 = 1$. Where λ_1, ..., $\lambda 4$ correspond to GAN, L_1, Perceptual and DSSIM loss accordingly.

4. Additional results

For additional results of generating 3D model of a person from a single image check Figures 5 and 6. The images shown in the figures are randomly selected from our test set consisting of images from the DeepFashion [5] and People Snapshot [1] datasets.

There you can also observe some of the typical failure cases of our method such as:

- Difficulty reconstructing skirts, dresses and any type of loose clothing (Figure 5, row 4). Since the SMPL model is a representation of a naked body, garment topologies that are far from the human body are challenging. The same holds for hats (Figure 5, row 9).
- Garments with rich texture and specific patterns are sometimes difficult to handle (Figure 5, row 5).
- Long hair and specific hair styles are often not properly reconstructed (Figure 6, row 4 and 10). Again this is because the SMPL model is not designed to handle hair. Additionally, more intricate hair styles are rare in our dataset.
- The segmentation completion can fail too.(Figure 5, row 1). Since we rely on off-the-shelf method for obtaining the ground-truth segmentation, even the ground-truth segmentation is not perfect. Actually, in many cases we have observed that the segmentation completed with our method looks better than the ground-truth.
- The face of the person is sometimes not preserved, instead similar face is generated from scratch. We believe this is because in our dataset we have less diversity in faces than in clothing. We often have scans of the same person wearing different clothes, hence the face it's easier to overfit. In this paper however our focus is more on the clothing.

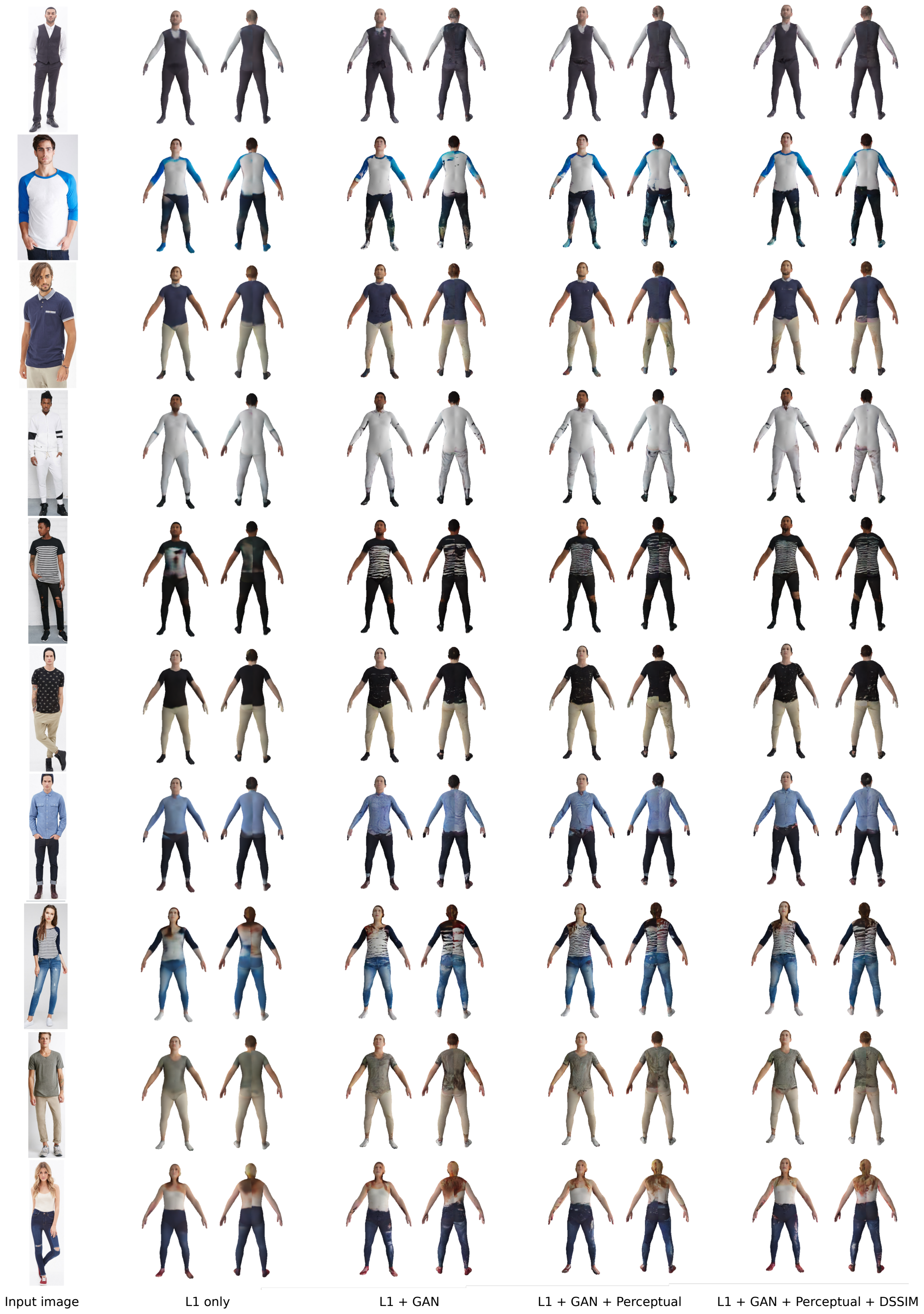

Figure 4: The effect of the different loss functions used for training our model.

Generator
Encoder
Conv1(filters: 32, filter size: 7×7, strides: 1) BatchNorm, ReLU
Conv2(filters: 64, filter size: 4×4, strides: 2) BatchNorm, ReLU
Conv3(filters: 128, filter size: 4×4, strides: 2) BatchNorm, ReLU
Conv4(filters: 256, filter size: 4×4, strides: 2) BatchNorm, ReLU
Bottleneck
Res1 (Residual Block): Conv(filters: 256, filter size: 3×3, strides: 1) BatchNorm, ReLU
$\vdots$
Res8 (Residual Block): Conv(filters: 256, filter size: 3×3, strides: 1) BatchNorm, ReLU
Decoder
Res8 $\oplus$ Conv4 Deconv1(filters: 128, filter size: 4×4, strides: 2) BatchNorm, ReLU
Deconv1 $\oplus$ Conv3 Deconv2(filters: 64, filter size: 4×4, strides: 2) BatchNorm, ReLU
Deconv2 $\oplus$ Conv2 Deconv3(filters: 32, filter size: 4×4, strides: 2) BatchNorm, ReLU
Deconv4(filters: 3, filter size: 4×4, strides: 1) Tanh

Table 1: The architecture of the generator.

- The body shape of the person and the shape of the clothing are not fully preserved. Since we are relying fully on the UV-space this is one of the drawbacks of our method. As we mentioned in the paper, additional component for predicting the shape of the person can be easily incorporated in our method. Nevertheless predicting the shape of the body is not part of our contribution.

Discriminator
Conv1(filters: 128, filter size: 4×4, strides: 1) Leaky ReLU
Conv2(filters: 256, filter size: 4×4, strides: 2) Leaky ReLU
Conv3(filters: 512, filter size: 4×4, strides: 2) Leaky ReLU
Conv4(filters: 1024, filter size: 4×4, strides: 2) Leaky ReLU
Conv5(filters: 2048, filter size: 4×4, strides: 2) Leaky ReLU
Conv6(filters: 1, filter size: 4×4, strides: 1)

Table 2: The architecture of the discriminator.

References

[1] T. Alldieck, M. Magnor, W. Xu, C. Theobalt, and G. Pons-Moll. Video based reconstruction of 3d people models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8387–8397, Jun 2018. CVPR Spotlight Paper. 2

[2] R. Alp Güler, N. Neverova, and I. Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7297–7306, 2018. 1

[3] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017. 1

[4] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[5] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[6] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):248, 2015. 1

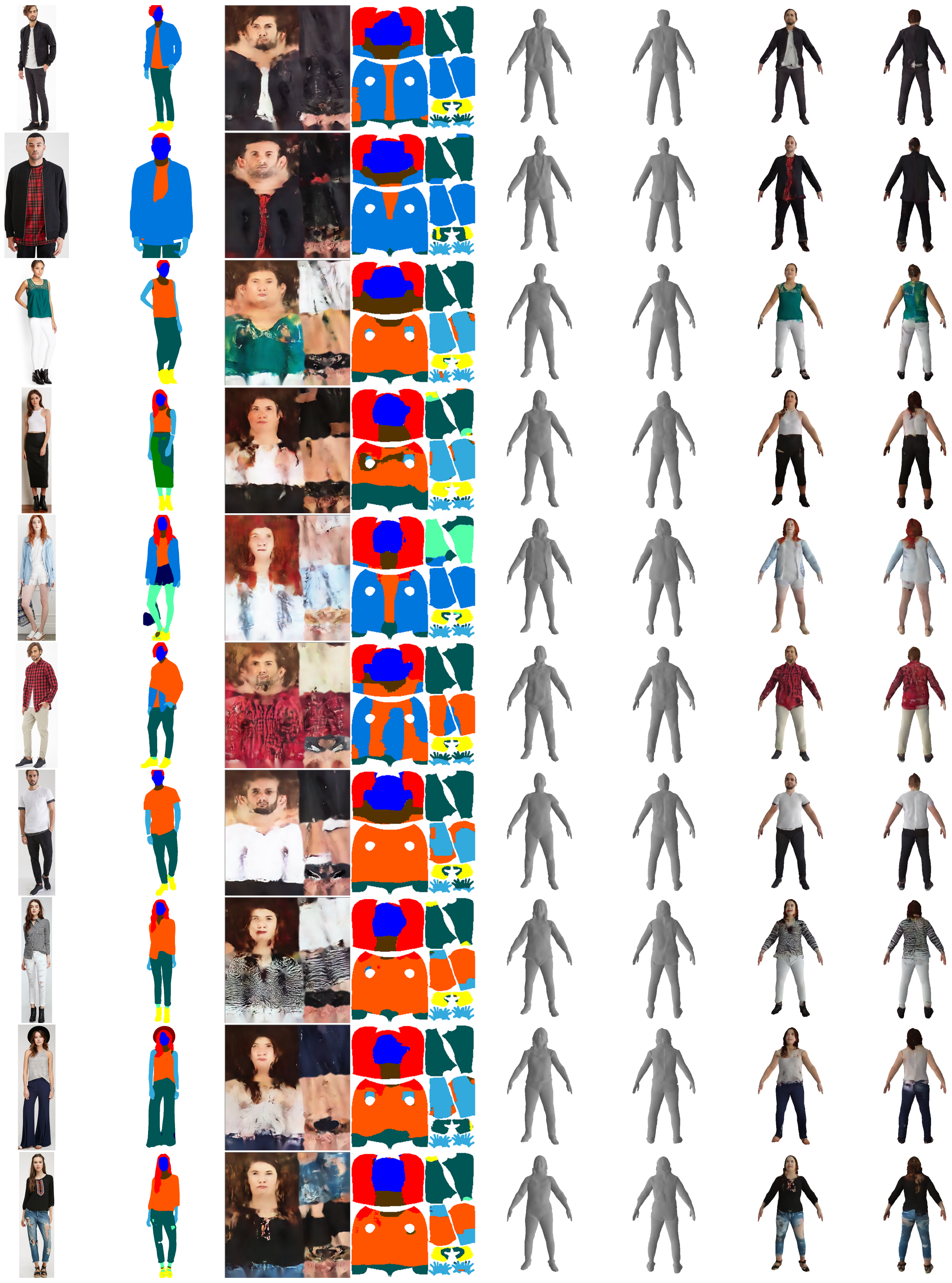

Figure 5: Image to 3D model, randomly selected results. Left to right: real image, segmented image, complete texture map, complete segmentation map, generated untextured avatar, generated textured avatar.

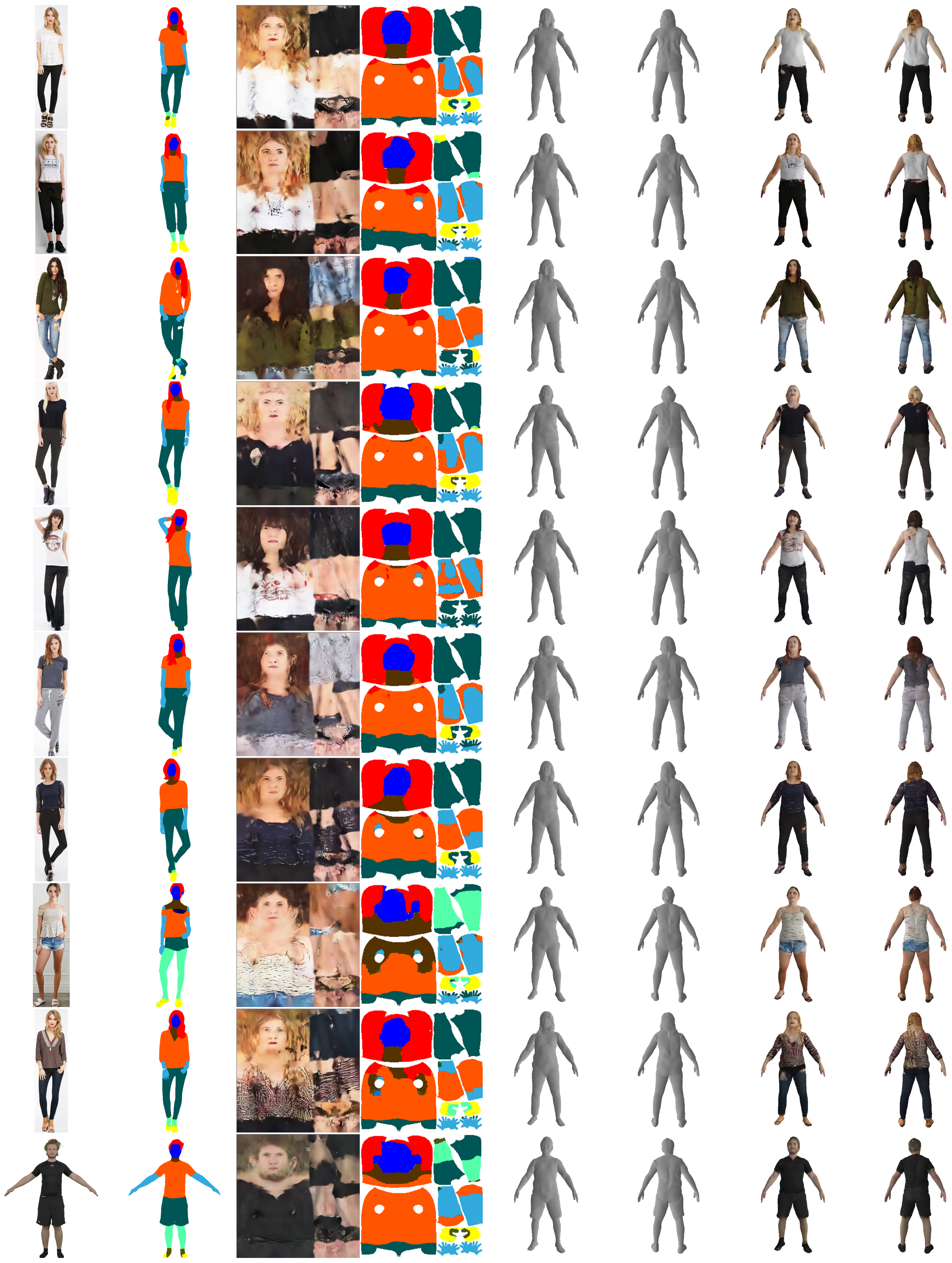

Figure 6: Image to 3D model, randomly selected results. Left to right: real image, segmented image, complete texture map, complete segmentation map, generated untextured avatar, generated textured avatar.