Supplementary Materials for gDNA: Towards Generative Detailed Neural Avatars

Xu Chen[1,3] Tianjian Jiang[1] Jie Song[1] Jinlong Yang[3]
Michael J. Black[3] Andreas Geiger[2,3] Otmar Hilliges[1]

[1]ETH Zürich, Department of Computer Science [2]University of Tübingen
[3]Max Planck Institute for Intelligent Systems, Tübingen

*In this **supplementary document**, we first provide implementation details in Section 1 and evaluation details in Section 2. We then show additional qualitative results in Section 3. Finally, we discuss potential negative social impacts, limitations and future directions of our method in Section 4.*

1. Implementation Details

1.1. Network Architecture

We implement our models in PyTorch [18]. Our architectures for the networks are illustrated in Fig. 1. Note that we choose a relatively small network size for the skinning weight network and warping network, as the skinning weight field and warping field are typically smooth in 3D space and thus do not require high-capacity networks to be modeled well. We use a positional encoding [16] with 4 frequency components. The shape codes and details codes are 64-dimensional each.

1.2. Training

Losses: As described in the main paper, we split training into two stages: we first train the coarse shape, skinning, and warping networks and then train the normal network. For the first stage, we use the binary cross-entropy loss $\mathcal{L}_{\text{BCE}}$ between

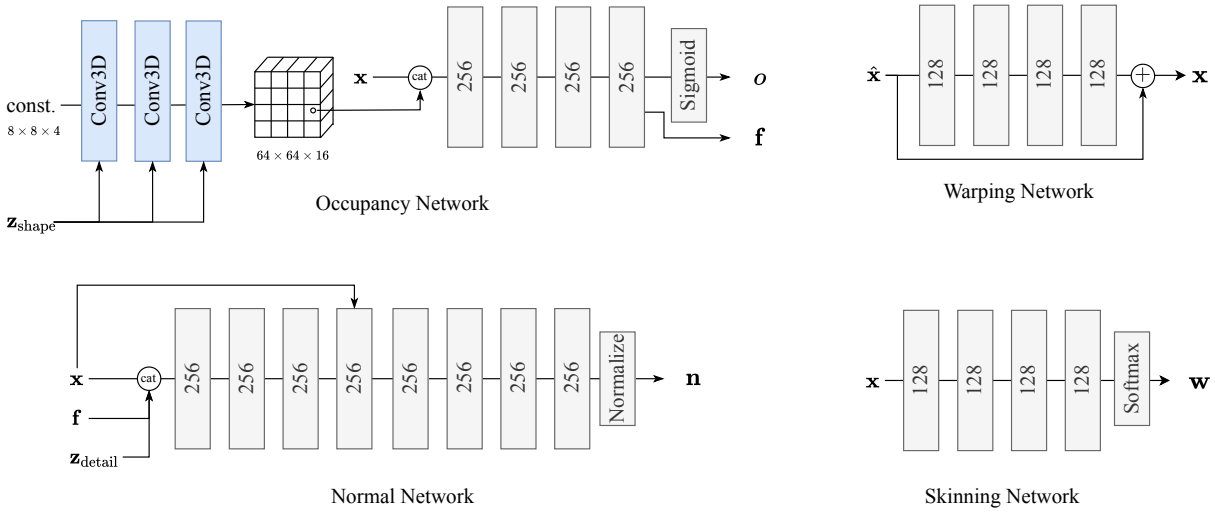


Figure 1. **Network Architecture.** Each gray block represents a linear layer with its output dimension specified in the inset, followed by a weight normalization layer [20] and a softplus [6] activation layer. Each blue block represents a sequence of the following layers: bilinear upsampler ($\times 2$) – 3D convolution (kernel size 3, stride 1) – adaIN [9] – leaky ReLU [12].

predicted occupancy $\mathcal{O}'(\mathbf{x}', \mathbf{z}_{\text{shape}}, \boldsymbol{\beta}, \boldsymbol{\theta})$ and ground-truth o_{gt}. Following [4], we add two auxiliary losses $\mathcal{L}_{\text{bone}}$ and $\mathcal{L}_{\text{joint}}$ to guide learning during early iterations:

$$\mathcal{L}_{\text{bone}} = BCE(\mathcal{O}(\mathbf{x}_{\text{bone}}, \mathbf{z}_{\text{shape}}), 1) \tag{1}$$

$$\mathcal{L}_{\text{joint}} = \|\mathbf{w}(\mathbf{x}_{\text{joint}}, \mathbf{z}_{\text{shape}}) - \mathbf{w}_{\text{joint, target}}\|_2^2 \tag{2}$$

where $\mathbf{w}_{\text{joint, target}}$ is a vector that is 0.5 for the neighboring bones and 0 elsewhere. To ensure that the warping field changes body size consistently, we enforce the warping field to warp SMPL vertices $\mathbf{v}(\boldsymbol{\beta})$ to the corresponding location in neutral shape $\mathbf{v}(\boldsymbol{\beta}_0)$:

$$\mathcal{L}_{\text{warp}} = \|\mathcal{M}(\mathbf{v}(\boldsymbol{\beta}), \boldsymbol{\beta}) - \mathbf{v}(\boldsymbol{\beta}_0)\|_2^2 \tag{3}$$

Finally, we regularize the latent code to be close to the origin of the latent space via $\mathcal{L}_{\text{reg,shape}} = \|\mathbf{z}_{\text{shape}}\|_2^2$. The resulting loss to train the coarse shape is defined as

$$\mathcal{L}_{\text{coarse}} = \mathcal{L}_{\text{occ}} + \lambda_{\text{bone}}\mathcal{L}_{\text{bone}} + \lambda_{\text{joint}}\mathcal{L}_{\text{joint}} + \lambda_{\text{warp}}\mathcal{L}_{\text{warp}} + \lambda_{\text{reg,shape}}\mathcal{L}_{\text{reg,shape}}, \tag{4}$$

with λ denoting the respective weights. We set $\lambda_{\text{warp}} = 10, \lambda_{\text{reg,shape}} = 10^{-3}$. During the first epoch, we set $\lambda_{\text{bone}} = 1, \lambda_{\text{joint}} = 10$ and 0 afterwards.

The normal prediction network is trained subsequently. Here we penalize differences between the predicted and GT normal $\mathbf{n}'_{\text{gt}}$ for randomly sampled surface points:

$$\mathcal{L}_{\text{norm}} = 1 - {\mathbf{n}'_{\text{gt}}}^T \cdot \mathcal{N}'(\mathbf{x}', \mathbf{z}_{\text{detail}}, \mathbf{f}, \boldsymbol{\beta}, \boldsymbol{\theta}) \tag{5}$$

In addition, we apply non-saturating adversarial losses [7] with R_1 gradient penalty [14] on the predicted 2D normal maps I and the real normal maps rendered from the posed scans I_{real}

$$\mathcal{L}_{\text{adv,G}} = -h\left(D\left(I\right)\right) \tag{6}$$

$$\mathcal{L}_{\text{adv,D}} = h\left(D\left(I\right)\right) - h\left(D\left(I_{\text{real}}\right)\right), \tag{7}$$

$$\text{where } h(u) = -\log(1 + \exp(-u))$$

$$\mathcal{L}_{R_1} = \left|\nabla D(I)\right|^2 \tag{8}$$

The predicted and real normal maps are rendered from one of 18 different views with an azimuth in $\{0, 20, 40...160\}^\circ$ in 256×256 image resolution.

We further regularize $\mathbf{z}_{\text{detail}}$ with $\mathcal{L}_{\text{reg,detail}} = \|\mathbf{z}_{\text{shape}}\|_2^2$. The resulting loss for training the normal prediction network is defined as

$$\mathcal{L}_{\text{detail}} = \mathcal{L}_{\text{norm}} + \mathcal{L}_{\text{adv,G}} + \lambda_{\text{reg,detail}}\mathcal{L}_{\text{reg,detail}} \tag{9}$$

and the loss for the discriminator is:

$$\mathcal{L}_D = \mathcal{L}_{\text{adv,D}} + \lambda_{R_1}\mathcal{L}_{R_1} \tag{10}$$

We set $\lambda_{R_1} = 10, \lambda_{\text{reg,detail}} = 10^{-3}$.

Initialization: Latent codes for training samples are initialized to zero. All network weights are initialized using the default initialization scheme of PyTorch.

Optimization: We train our networks using the Adam optimizer [10] with a learning rate of $\eta = 10^{-3}$ for stage 1 and $\eta = 2 \times 10^{-3}$ for stage 2. No weight decay or learning rate decay is used.

1.3. Training Data

As described in the main paper, we use 588 scans from RenderPeople [2] and 3DPeople [1] covering male and female adults in varied clothing, including suits, shirts, pants, shorts etc. The list of scans will be made publicly available. We do not consider skirts and dresses in this paper due to several technical challenges discussed in Section 4.2.

For body pose and size $\boldsymbol{\theta}, \boldsymbol{\beta}$, we use registered SMPL parameters provided in the AGORA [19] dataset. Note that, unlike CAPE [11] and NPMs [17], we do not require any detailed surface registration but only low-dimensional SMPL parameters.

1.4. Inference

At inference time, we generate human avatars by randomly sampling $\mathbf{z}_{\text{shape}}$ and $\mathbf{z}_{\text{detail}}$ from the estimated Gaussian distribution. We then extract meshes in resized canonical space using MISE [15] from the implicit representation $\hat{\mathcal{S}}(\mathbf{z}_{\text{shape}}, \boldsymbol{\beta})$ in 256^3 spatial resolution and predict the vertex normal $\mathbf{n}$ and skinning weights with our normal field and skinning field. For consistency between predicted normals and the coarse shape, we aggregate the predicted normals with the spatial gradient of our occupancy field to obtain the final vertex normal direction $\bar{\mathbf{n}}$:

$$\bar{\mathbf{n}} = \gamma(\mathbf{n}^T \cdot \nabla o) \cdot \mathbf{n} + (1 - \gamma(\mathbf{n}^T \cdot \nabla o)) \cdot \nabla o, \text{ with } \gamma(u) = \exp(-\frac{(1-u)^2}{8\sigma^2}) \tag{11}$$

with $\sigma = 0.2$. The weight γ favors the predicted normals if they are similar to the normals of the coarse shape, thus preserving high-frequence details. Otherwise, the normals of the coarse shape are favored to ensure consistency. Finally, we pose the mesh-based avatar to desired body poses $\boldsymbol{\theta}$ using standard LBS. Mesh extraction takes 3 seconds on average for a grid resolution of 256^3, and 10 seconds for 512^3. Reposing takes around 0.3 seconds per frame.

1.5. Fitting

Losses: For fitting model to scans, we minimize the binary cross entropy loss $\mathcal{L}_{\text{BCE}}$ between predicted occupancy $\mathcal{O}'(\mathbf{x}', \mathbf{z}_{\text{shape}}, \boldsymbol{\beta}, \boldsymbol{\theta})$ and the occupancy obtained from the target scan o_{target}, regularized by $\mathcal{L}_{\text{reg,shape}} = \|\mathbf{z}_{\text{shape}}\|_2^2$. The resulting loss for fitting is $\mathcal{L}_{\text{fittng}} = \mathcal{L}_{\text{BCE}} + \lambda_{\text{reg,shape}}\mathcal{L}_{\text{reg,shape}}$ with $\lambda_{\text{reg,shape}} = 50$.

Initialization: We initialize body size $\boldsymbol{\beta}$ and pose $\boldsymbol{\theta}$ with the registered SMPL parameters provided in the SIZER dataset, shape code $\mathbf{z}_{\text{shape}}$ with 8 randomly sampled codes from the estimated Gaussian distribution, and details code $\mathbf{z}_{\text{detail}}$ with the mean of the details code distribution.

Optimization: We optimize body size $\boldsymbol{\beta}$ and shape code $\mathbf{z}_{\text{shape}}$ using the Adam optimizer [10] with a learning rate of $\eta = 10^{-2}$ for 500 iterations. No weight decay or learning rate decay is used. Other parameters and network weights are fixed during fitting. Among 8 solutions corresponding to 8 initial states, we select the one with the lowest bi-directional Chamfer distance to the target scan as the final solution.

2. Evaluation Details

2.1. FID Computation

We compute FID score using pytorch-fid library[1]. For each method, we randomly generate 500 shapes in random body poses and sizes sampled from the training set, and render 2D normal maps in resolution 256^2 from 18 viewpoints with randomly sampled azimuth angle in $[0, 360)^\circ$, resulting in 9000 normal maps. We also render all training scans from 18 different views, resulting in 10584 normal maps as real data. All samples are randomly sampled without curation.

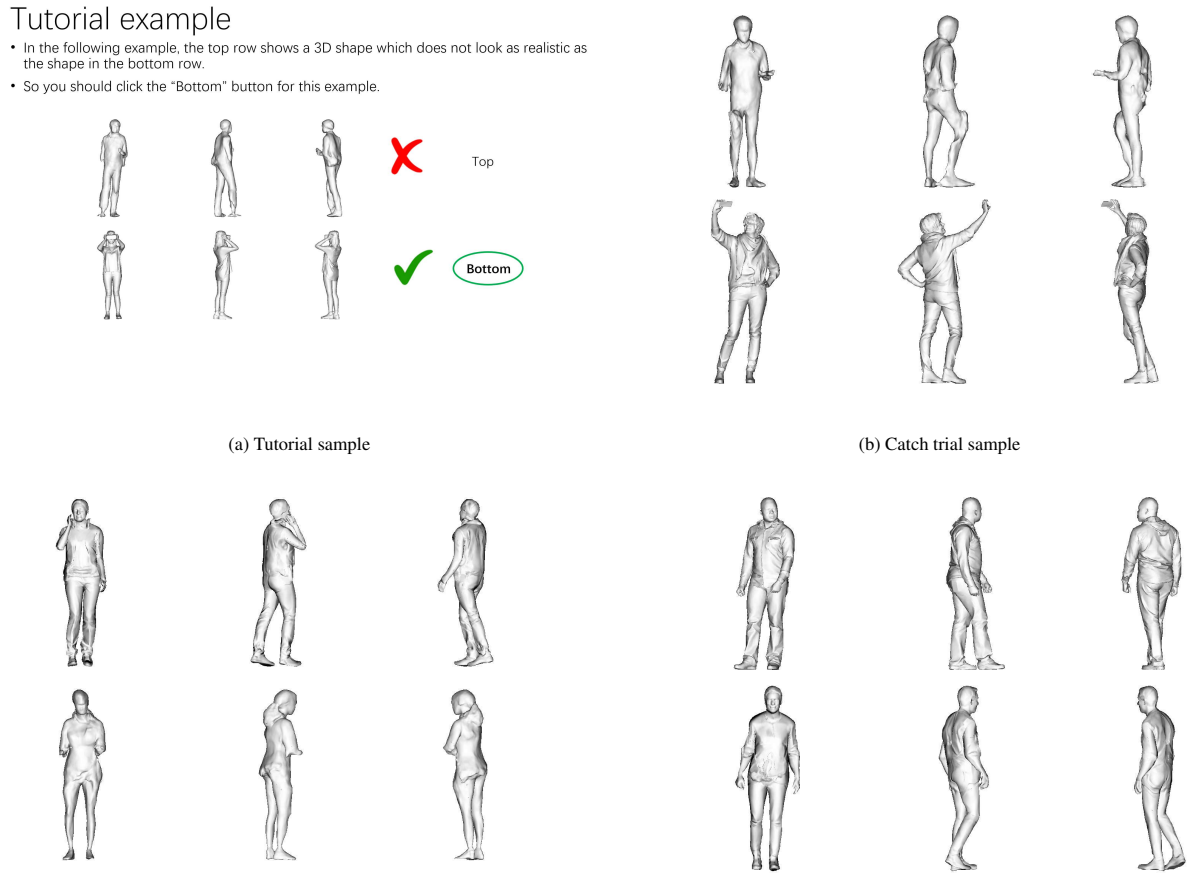
2.2. User Preference Study

We conduct a user preference study via Amazon Mechanic Turk (AMT). We sequentially show each subject 1) 1 tutorial sample 2) 10 warm-up samples and 3) 100 evaluation samples mixed with 10 catch trial samples. In each sample, we render shaded surfaces from three different views in resolution 256^2 of one 3D shape generated by our method and one 3D shape generated by one of the baseline methods. The subjects are required to select the more realistic shape. Examples of each type of samples are shown in Sec. 2.2. The warm-up samples are used to help subjects to understand and to get used to the task. They are randomly chosen from evaluation samples, but are not counted in the statistical results. The catch trial samples are used to detect outlier subjects. These samples consist of one row of ground truth scans and another row of shape generated by *Pose ONet* from latent codes that are at least $3 \times std$ away from the mean, which contain obvious artifacts. A subject is considered invalid if he/she fails on 30% or more of the catch trial samples. In total, we acquired 44 valid subjects out of 48 participants. We report the chance that a baseline method is considered more realistic than our result in the main paper. Only the results on evaluation samples count. All samples are randomly sampled without curation.

2.3. Fitting Comparison

We use SIZER dataset for fitting comparison with NPMs [17] and SMPLicit [5]. SIZER contains 21 garments in three sizes and worn by various subjects. We randomly sample one scan from each garment in the SIZER dataset for evaluation,

[1] https://github.com/mseitzer/pytorch-fid



(a) Tutorial sample

(b) Catch trial sample

(c) Two evaluation samples

Figure 2. **User Preference Study Example.**

resulting in 21 evaluation samples. We use the accompanied SMPL body pose and size parameters in the dataset to initialize SMPLicit [5] and our method. As NPMs [17] do not model articulation and cannot leverage pose initialization, we run their own initialization process, which predicts latent pose and shape code from the target scan using a neural network. For SMPLicit [5], we in addition provide the clothing segmentation as required. For both methods, we use the code and pre-trained weights from their official repositories[2].

3. Additional Qualitative Results

3.1. Random Sampling using Existing Methods

While NPMs [17] and SMPLicit [5] learn a latent space of clothing or clothed human shapes and could potentially generate random samples from this latent space, such functionality is not demonstrated in their paper. We show our best attempt to sample clothed human shapes using NPMs [17] and SMPLicit [5] in Fig. 3. We remark that random sampling is not the goal of these two methods, and this demonstration is not meant as an evaluation but to show that the community lacks a method to randomly generate clothed human shapes with details under pose control. We detail the sampling strategy we use for NPMs [17] and SMPLicit [5] below.

NPMs: NPMs are trained using the auto-decoding framework. Similar to ours, we fit a Gaussian distribution to the latent codes of NPMs and sample from the estimated distribution. Note that NPMs model the deformation as free-form displacement without articulation. As a result, some generated posed shapes are distorted due to the flexibility of free-form displacement.

[2]NPMs: https://github.com/pablopalafox/npms; SMPLicit: https://github.com/enriccorona/SMPLicit

Finally, as NPMs are designed for fitting, it uses a high-dimensional latent space ($\mathbb{R}^{256}$) for 118 identities to capture fine details. Supervising a high-dimensional latent space with sparse training data leads to degenerate random samples with limited shape diversity and geometric details.

SMPLicit: SMPLicit uses two latent codes: $\mathbf{z}_{\text{cut}}$ to model clothing type and $\mathbf{z}_{\text{style}}$ to model clothing styles within each type. $\mathbf{z}_{\text{cut}}$ is learned via auto-encoding from an occlusion map in UV space, and $\mathbf{z}_{\text{cut}}$ is learned via auto-decoding. As $\mathbf{z}_{\text{cut}}$ is not regularized during training, sampling is not straightforward. We thus use the mean codes for each clothing types provided in their official code repository and add minor Gaussian noise for diversity. We manually tune the magnitude of the noise to balance diversity and quality and obtain diverse valid clothing samples from SMPLicit. However, the generated clothing shapes lack details due to the synthetic training data.

3.2. Fitting and Reposing Comparison

We show additional results on fitting to raw scans and reposing the fitted model in Fig. 4. As explicit pose control is not straightforward with NPMs. we show their reposing results in randomly sampled latent pose codes instead of the target pose.

3.3. Additional Results

We show additional qualitative results for clothed human generation (Fig. 5, Fig. 6), disentangled generation of shape and details (Fig. 7), interpolation (Fig. 8) and generation comparison with ablative baselines (Fig. 9).

3.4. Disentanglement of Body Size and Clothing

By changing $\mathbf{z}_{\text{shape}}$ and $\boldsymbol{\beta}$ independently, our method can generate human avatars with the same body shape/size in different clothing, and also humans with different body shapes/sizes in the same clothing, as illustrated in Fig. 10.

4. Discussions

4.1. Potential Negative Impacts

While current generated human shapes are not indistinguishably realistic, as the technology matures, it will open up the possibility for full-body deep fakes with all the accompanying risks. These risks must also be balanced by the positive use cases in entertainment, telepresence, and future metaverse applications. Clear regulation will be needed to establish legal boundaries for its use. Furthermore, studying such technologies in the open, including disclosure of technical details, code and data as well as an discussion of the properties and limitations of such methods may inform nefarious use but can certainly also inform counter measures to deep-fake technologies. Developing detection and mitigation approaches would be significantly harder if this information were not publicly available.

4.2. Limitations and Future Works

Loose Clothing: While implicit surface representations are compatible with varying clothing topology, modeling very loose clothing such as skirts and dresses remains challenging, especially if learned from posed scans only without surface correspondences. The first challenge is the ambiguity between pants and skirts in posed space. Despite the very different topology between pants and skirt in canonical space, they might appear similar in posed space and both types of topology fulfill the 3D reconstruction loss. As a result, our method learns to generate pant and skirt topology randomly for training samples with skirts or dresses, as shown in Fig. 11. Besides this ambiguity, realistic animation of loose clothing further requires modeling pose-dependent or dynamic clothing deformations, which is currently missing (see next paragraph). Due to these challenges, we exclude loose clothing types in this paper.

Pose-dependent Clothing Deformations: Realistic clothing deformations wrt. pose are currently missing in our model because scans in common datasets are static and are limited in pose diversity. An exciting direction is to combine various data sources, e.g. cloth simulation and real-world scans, to learn cloth deformations with realistic geometry.

Semantic Control: Our method does not support controlling semantic attributes, e.g. clothing types, due to missing labels in the training dataset. Annotating existing datasets with such information or incorporating ideas in unsupervised disentanglement [3, 8] could enable semantic control over the generation process.

Accessories: Accessories, such as glasses or hand-held objects, are currently modeled as part of the human body. Future work could explore the compositionality of humans and accessories to enable object-level control.

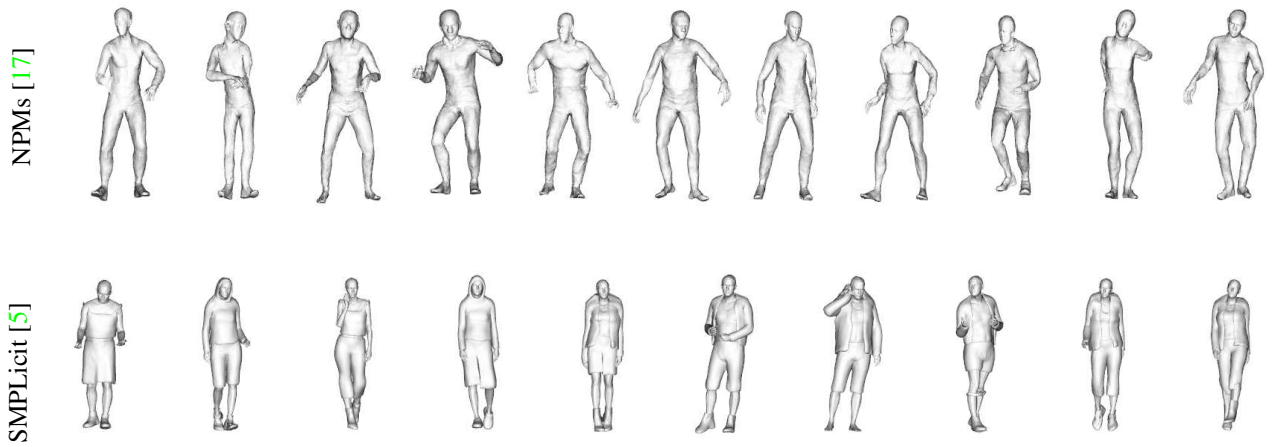


Figure 3. **Random Samples from Existing Methods.**

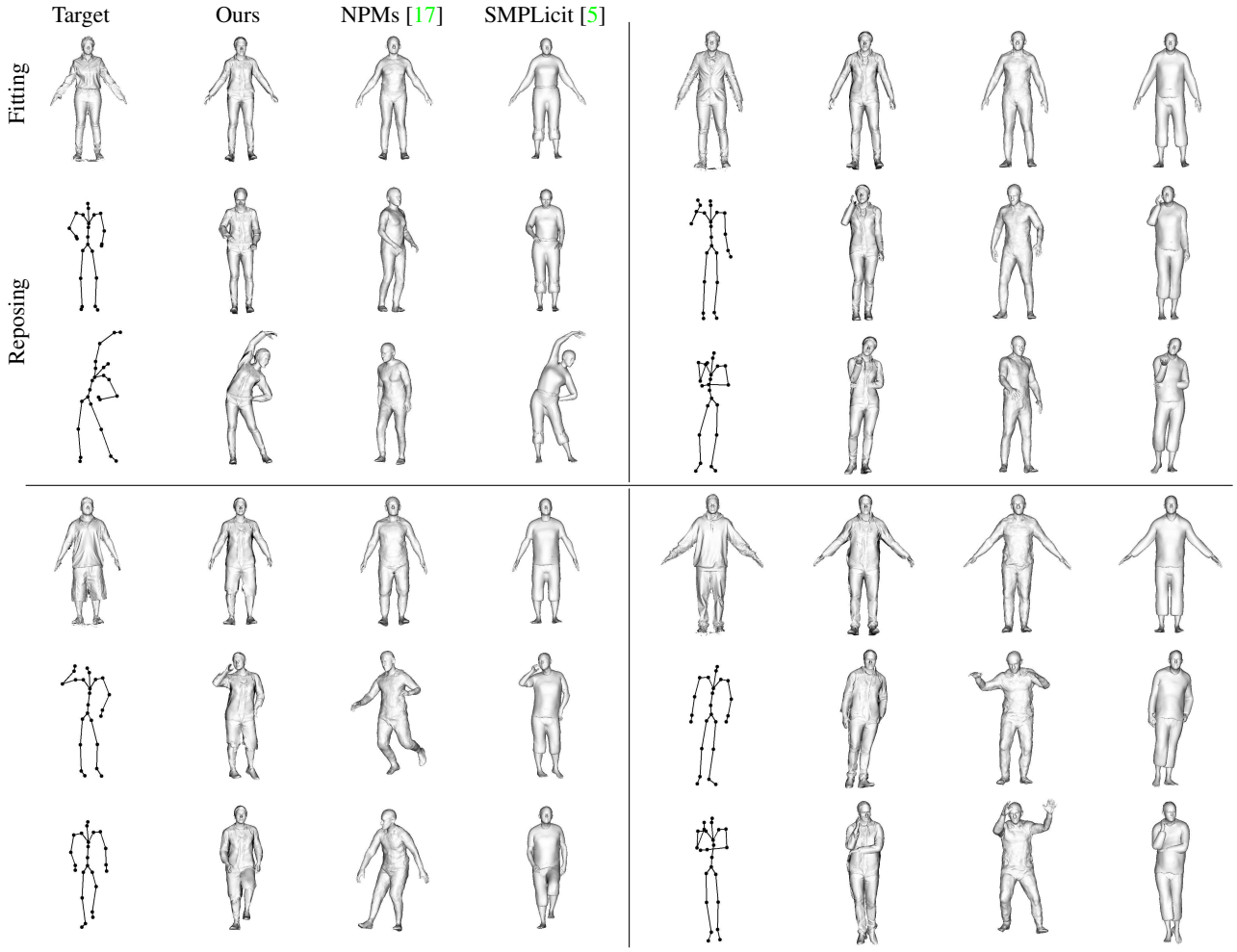


Figure 4. **Fitting and Reposing Comparison with SOTA.** In the first row of each grid cell, we show the fitting results (from left to right: *Targe Scan*, *Ours*, *NPMs*, *SMPLicit*). In the second and third row of each grid cell, we show reposing results (from left to right: *Targe Pose*, *Ours*, *NPMs*, *SMPLicit*). As explicit pose control is not straightforward with NPMs, we show their results in randomly sampled latent pose codes instead.

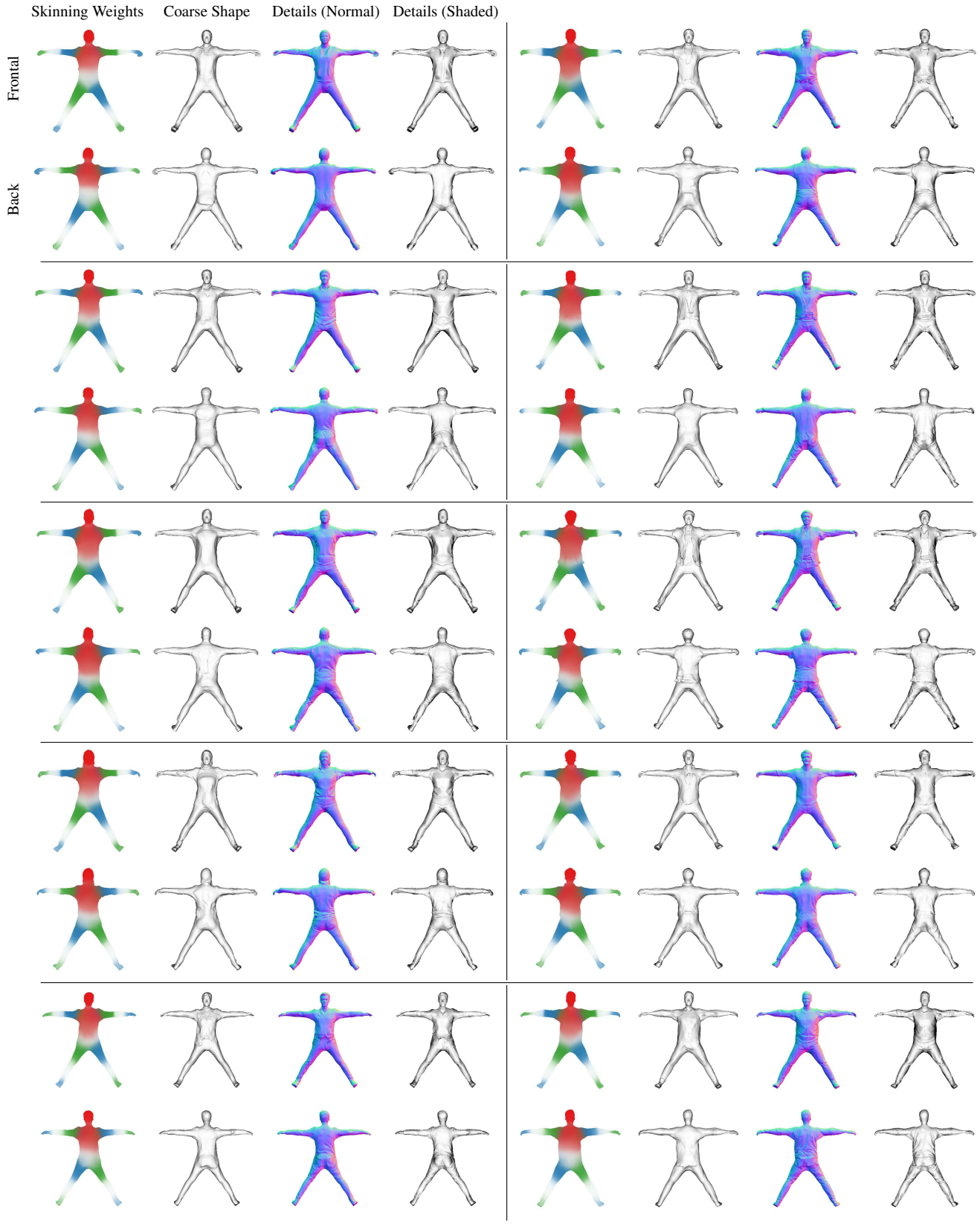


Figure 5. **Clothed Human Generation.** We show randomly sampled clothed human shapes generated by our method. In each group from left to right, we show learned skinning weights, coarse shape, fine details visualized as normals and shaded surfaces, from both frontal view (*first row*) and back view (*second row*). We encourage the reader to zoom-in for details.

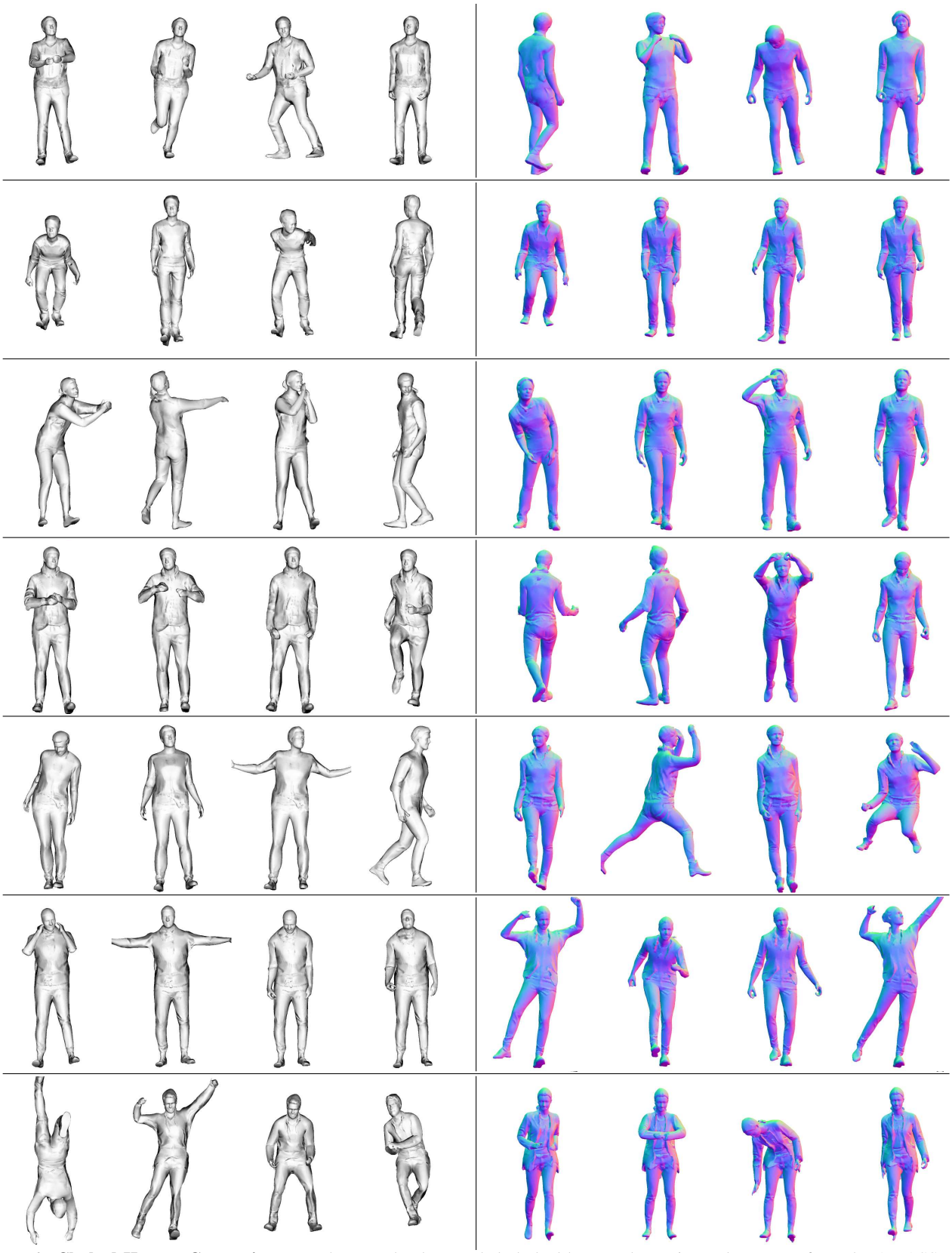


Figure 6. **Clothed Human Generation.** We show randomly sampled clothed human shapes in random poses from the AMASS [13] dataset. We visualize the generated samples as shaded surfaces (*left*) and normal maps (*right*). We encourage the reader to zoom-in for details.

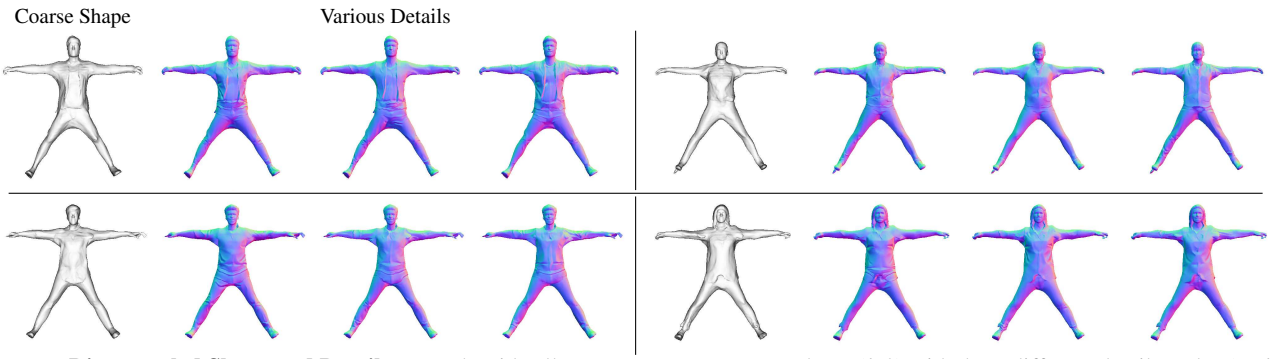


Figure 7. **Disentangled Shape and Details.** In each grid cell, we generate one coarse shape (*left*) with three different details codes (*right*).

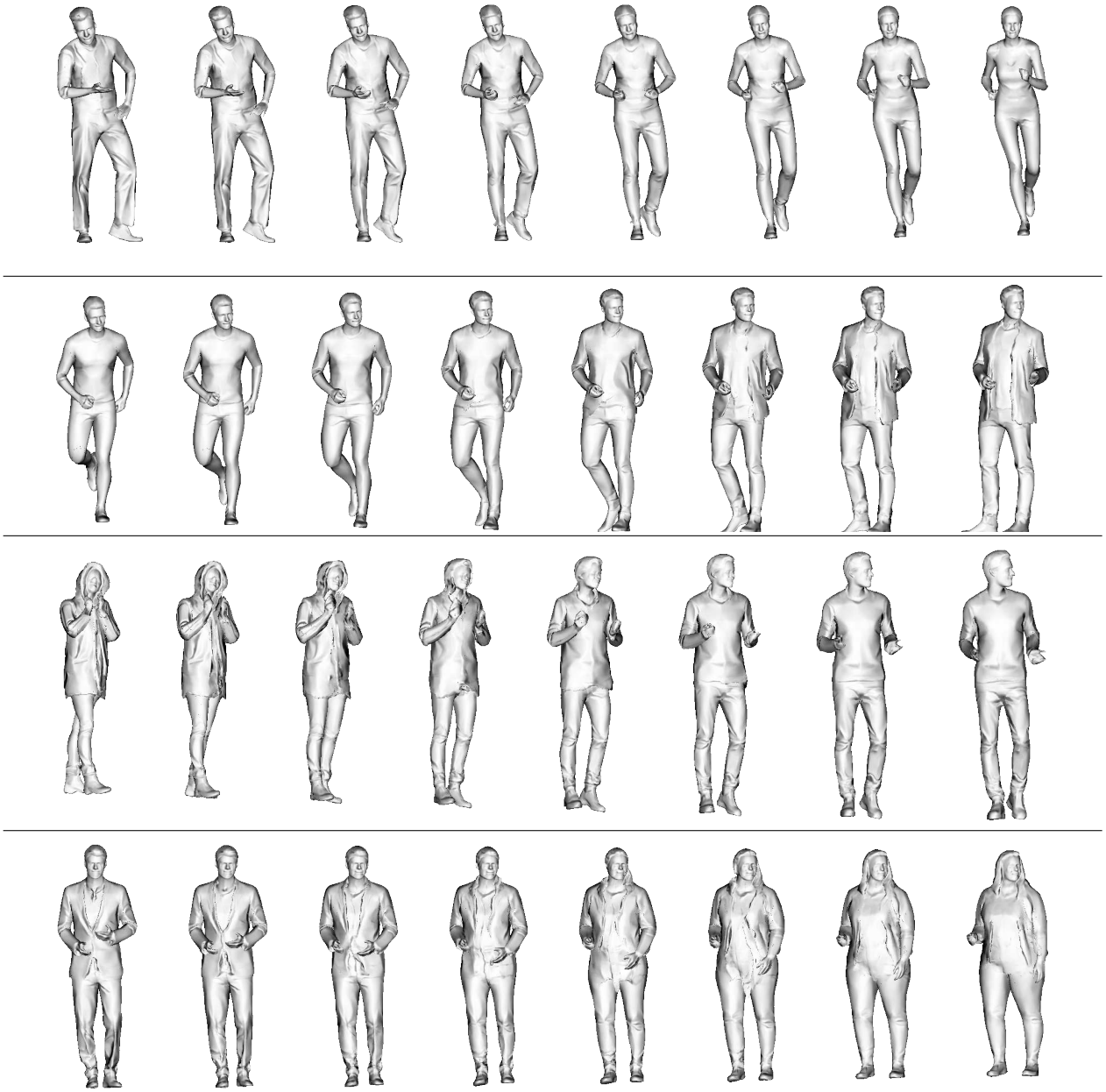


Figure 8. **Interpolation.** We interpolate the pose and shape code and the detail code between the leftmost and rightmost sample.

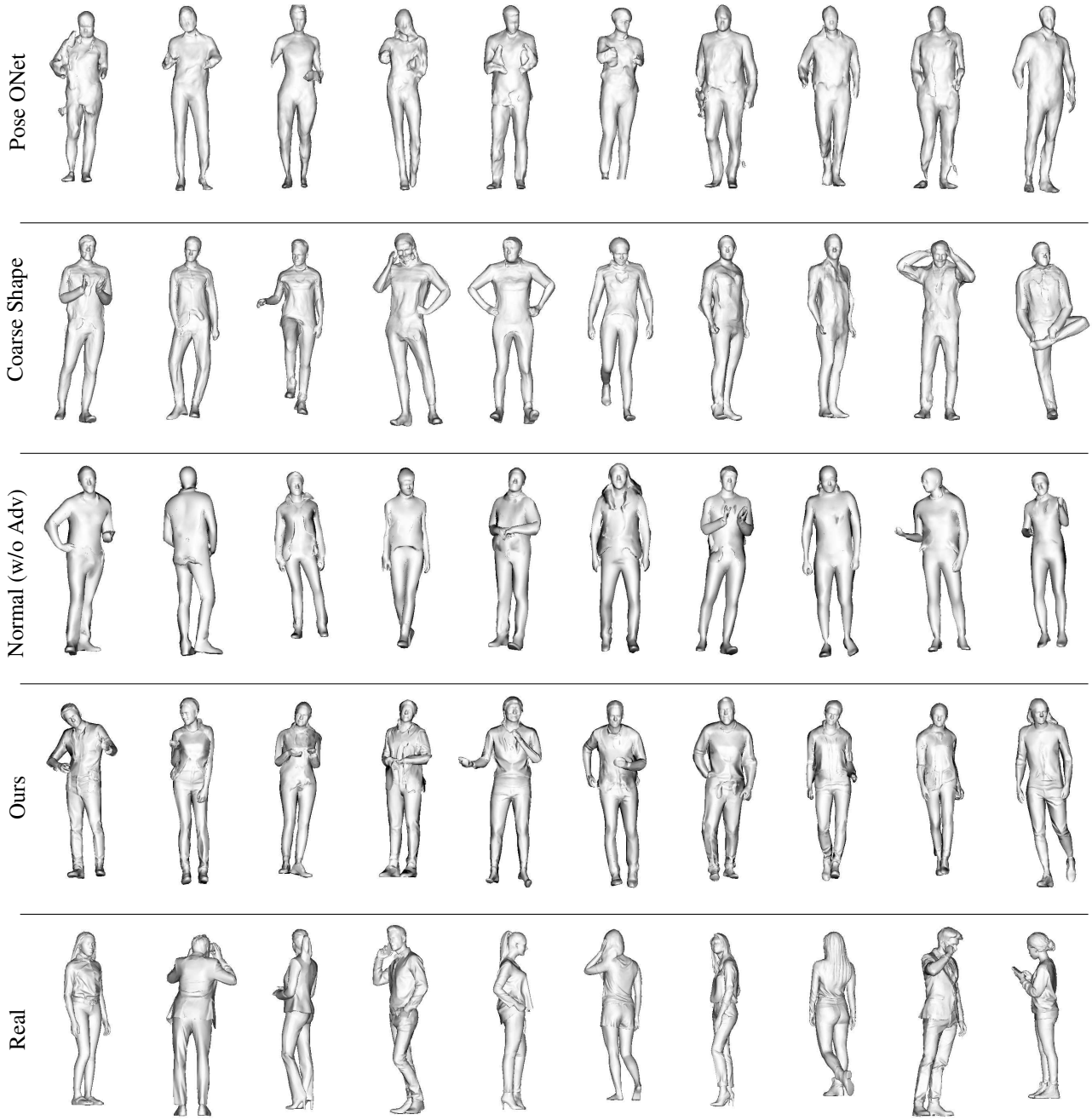


Figure 9. **Generation Comparison with Ablative Baselines.** We show random samples from ablative baselines and our method. For top to bottm: *Pose ONet*, *Coarse Shape*, *Detailed Normals w/o Adv. Loss*, *Ours*, *Real*.

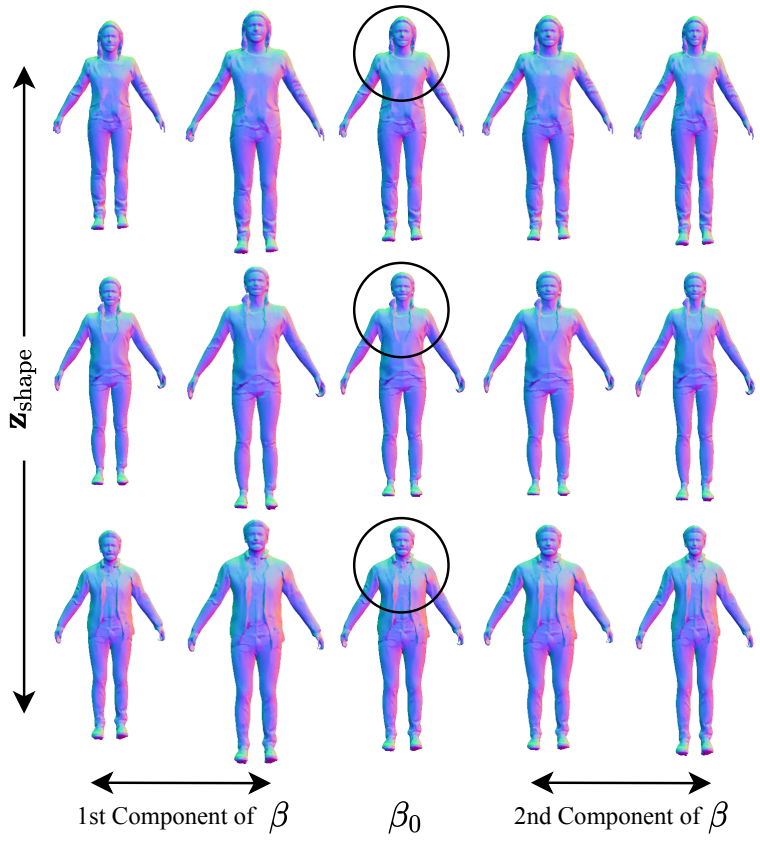


Figure 10. **Independent control of $\mathbf{z}_{\text{shape}}$ (columns) and β (rows)**. Note that $\mathbf{z}_{\text{shape}}$ controls clothing, hair and face (circle) and β controls overall body shape such as body height (left, 1st component) and width (right, 2nd component) analogously to SMPL.

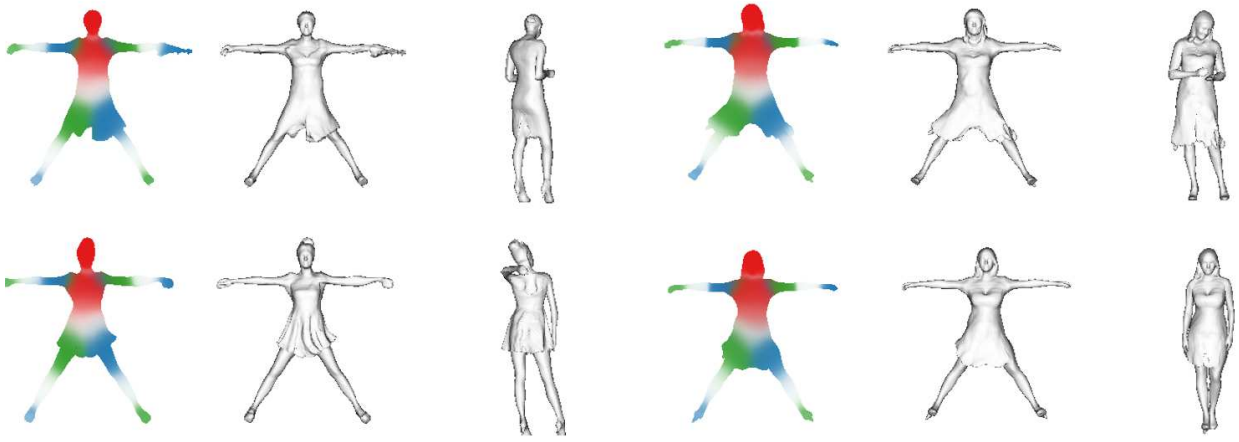


Figure 11. **Challenges on Modeling Loose Clothing.** Trained from posed scans only, our method cannot disambiguate skirt and pants topology and can generate either type for training samples with loose clothing. *Top:* pants topology generated by our method. *Bottom:* skirt topology generated by our method. Despite having different topology in canonical space, these generated shapes exhibit the shape of skirt after reposing.

References

[1] https://3dpeople.com/. 2

[2] https://renderpeople.com/. 2

[3] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 5

[4] Xu Chen, Yufeng Zheng, Michael J Black, Otmar Hilliges, and Andreas Geiger. SNARF: Differentiable forward skinning for animating non-rigid neural implicit shapes. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2

[5] Enric Corona, Albert Pumarola, Guillem Alenyà, Gerard Pons-Moll, and Francesc Moreno-Noguer. SMPLicit: Topology-aware generative model for clothed people. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3, 4, 6

[6] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating second-order functional knowledge for better option pricing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2000. 1

[7] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. 2

[8] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2017. 5

[9] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 1

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015. 2, 3

[11] Qianli Ma, Jinlong Yang, Anurag Ranjan, Sergi Pujades, Gerard Pons-Moll, Siyu Tang, and Michael J. Black. Learning to dress 3D people in generative clothing. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[12] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. of the International Conf. on Machine learning (ICML)*, 2013. 1

[13] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 8

[14] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In *Proc. of the International Conf. on Machine learning (ICML)*, 2018. 2

[15] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[16] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 1

[17] Pablo Palafox, Aljaž Božič, Justus Thies, Matthias Nießner, and Angela Dai. Neural parametric models for 3D deformable shapes. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 2, 3, 4, 6

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 1

[19] Priyanka Patel, Chun-Hao P. Huang, Joachim Tesch, David T. Hoffmann, Shashank Tripathi, and Michael J. Black. AGORA: Avatars in geography optimized for regression analysis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[20] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 1