Supplementary Material: Structured 3D Features for Reconstructing Controllable Avatars

Enric Corona[1†] Mihai Zanfir[2†] Thiemo Alldieck[3]
Eduard Gabriel Bazavan[3] Andrei Zanfir[3] Cristian Sminchisescu[3]
[1]UPC, Barcelona [2]Newton [3]Google Research

In this document, we describe the implementation details of our method extensively and provide more results and failure cases. We also include a Supplementary Video summarizing our contributions and results.

1. Implementation Details

Data. Our synthetic data is based on a set of 3D RenderPeople Scans [16]. We use 35 rigged and 45 posed scans for training. The rigged scans are re-posed to 200 different poses sampled randomly from the CMU motion sequences [5]. With probability 0.4 we render a scan from a frontal view, otherwise, we render from a random azimuth and a uniform random elevation in $[-20, 20]^\circ$. We render each scan using high dynamic range image (HDRI) [10] lighting and backgrounds using Blender [3]. We obtaining ground-truth albedo directly from the scan's texture map and bake the full scan's shading (including occluded regions) to obtain ground-truth shaded colors.

For real images, we use the HITI dataset [2], which contains 150K images in-the-wild with predicted foreground segmentation masks and annotated 2D human keypoints. We obtained initial GHUM parameters for each image by estimating pose and shape using [8]. We then further optimized pose and shape parameters by minimizing the 2D reprojection error, the normalizing flow pose prior from [21, 22] and a body shape prior. The weights for joint reprojection, body pose regularization and body shape regularization are 10, 1 and 10 respectively, and we assumed a perspective camera projection with fixed focal length. After fitting, we remove fits that have an average reprojection error greater than 3 pixels or where at least 10% of the body surface projects outside the segmentation mask, leading to 40k training images. For inference on images-in-the-wild we follow the same fitting procedure, by leveraging predicted 2D keypoints instead of ground-truth annotations.

Architecture. We take masked input images at 512×512 px resolution. We augment the input with rendered normal and semantic maps to provide information about the GHUM fit to the feature extractor network. The semantic map is obtained by rendering the original template vertex locations as vertex colors, essentially defining a dense correspondence map to GHUM's zero-pose. We normalize both maps between 0 and 1 and stack them with the original image before passing all to the image feature extractor network. We noticed that by concatenating normal and semantic maps, the network is better able to correct noisy GHUM fits and their geometry. The feature extractor network is a U-Net [17] with 6 encoder and 7 decoder layers with sizes $[64, 128, 256, 512, 512, 512]$ and $[512, 512, 512, 512, 256, 256, 256]$ respectively. The illumination code is extracted from the bottleneck and has shape $8 \times 8 \times 512$. The output per-pixel feature maps is of shape $512 \times 512 \times 256$, with 256-feature vectors.

We next detail how we sample points on the body surface and pool from image features. We explored different point densities sampled from the body surface, all based on the original GHUM mesh to maintain correspondences. To subdivide a mesh, we add a vertex in the center of each edge, increasing the number of faces by a factor of 4. The subdivision is fast to perform at train/test time and does not cause any significant overhead. However, naively subdividing points on the mesh leads to a large number of body points causing memory issues in later stages (*e.g.* processing them on the transformer encoder), thus we additionally run K-Means on the subdivided template mesh obtaining clusters of 2k, 5k, 8k, 10k, 12k, 15k and 18k body points. We ablate qualitatively the number of points on the model in Fig. 11. Using fewer points does not significantly affect the results, but produces blurrier color reconstructions. Our final model uses 18k points. During inference, given the GHUM fit of an image and estimated camera parameters, we project these points to the image and extract image features. We use the last 64 features of the previously extracted image features to predict per-vertex deformation, using a 2-Layer MLP with hidden shapes $[64, 3]$ and leaky-ReLU af-

† Work was done while Enric and Mihai were with Google Research.

ter the first layer. We project the deformed vertices again to obtain the remaining 192 per-pixel features.

The goal of the transformer encoder is to efficiently map a query point $\boldsymbol{x}$ to the Structured 3D Features. We first map both deformed body points and query point positions to a higher-dimensional space using positional encoding with 6 frequencies, and apply a shared 2-Layer MLP with output size 256. In practice, we use two MLPs predicting geometry and albedo independently. We next apply attention [20] to combine per-point features and obtain $\boldsymbol{f}^{\star}_{\boldsymbol{x}}$. The final geometry and color heads are both MLPs with eight 512-dimensional fully-connected layers and Swish activation [15], an output layer with Sigmoid activation for the color component, and a skip connection to the fourth layer. The shading network s is conditioned on the previously extracted illumination code and consists of three 256-dimensional fully-connected layers with ReLU activation, including the output layer.

The weights of all layers are initialized with Xavier initialization [6], with the β parameter (Eq. 9) being initialized as 0.1. We train all network components jointly end-to-end for 500k iterations using the Adam optimizer [12], with an initial learning-rate of 1×10^{-4} that linearly decays with factor 0.9 every 50k steps. Training takes 5 days under 8 GPUs V100 at batch size 8. During training, the time bottleneck is the U-Net feature extractor. However, the transformer is the memory bottleneck when computing attention, which scales with the number of body points and query points. The 3D reconstructions are obtained after running Marching Cubes [13] at a densely sampled volume of $512 \times 512 \times 512$ points, as common in previous works [1, 18, 19]. After running Marching Cubes for geometry reconstruction, we query the network again on the reconstruction's vertices, to obtain albedo and color estimations and texture the mesh. To render new views, we explored the possibility of rendering the colored mesh vs. using neural rendering without obtaining the 3D mesh. We did not find significant differences between the two. In our results, we apply the former and first obtain the mesh with Marching Cubes for convenience.

Point and pixel sampling strategy. For training on the 3D RenderPeople scans, we sample 128 points uniformly on the scan's surface, and 128 points close to the body (by adding noise $\sim \mathcal{N}(0, 1 \text{ cm})$ to scan vertices). For the Eikonal Loss, we sample points around GHUM by adding noise $\sim \mathcal{N}(0, 10 \text{ cm})$ to GHUM's original vertices.

We rely on the original segmentation mask to compute image-based losses, starting by sampling pixels efficiently on foreground regions to let the network focus on occupied regions. We randomly sample 32 pixels from the input image, from which 75% are sampled inside the foreground mask, and the rest are outside the mask. For the VGG-loss [4] $\mathcal{L}_{\text{vgg}}$ we render a 16×16 patch by sampling its center pixel randomly from the foreground mask.

Loss functions. We next describe how we obtain pixel colors and our neural rendering losses more in detail. Following the notation of the main document, the color of the pixel $\mathbf{c}_k$ is approximated by a discrete integration between near and far bounds t_n and t_f of a camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ with origin $\mathbf{o}$:

$$\boldsymbol{c}_k = \int_{t_n}^{t_f} T(t)\sigma(r(t))\mathbf{c}(\mathbf{r}(t),)dt\,, \tag{1}$$

where:

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)\,, \tag{2}$$

and $\sigma(\cdot)$ is the predicted density of a query point, as defined in the main document. Note that the point color being integrated is the shaded point color, which is a composition of its albedo and normal, obtained from $\boldsymbol{n}_{\boldsymbol{x}} = \nabla_{\boldsymbol{x}} s_{\boldsymbol{x}}$ for point $\boldsymbol{x}$. In practice, for each target pixel, we sample 64 points uniformly along the ray, within the bounding box of GHUM's nearest and furthest z vertex locations padded with 10 cm.

Our experience is that the main trade-off during training is on excessive detail (noise in geometry) vs. excessive smoothness and found the weight of the Eikonal component to be the most important hyperparameter to balance the capacity of the model to generate wrinkles. In this manner, we run a hyperparameter sweep on λ_{eik} from 0.01 to 0.5. The loss weights for the final model are $\lambda_{\text{rgb}} = 10$, $\lambda_{\text{vgg}} = 30$, $\lambda_{\text{mask}} = 5$, $\lambda_{\text{eik}} = 0.1$, $\lambda_{\text{3D rgb}} = 200$, $\lambda_{\text{3D label}} = 30$. Losses $\mathcal{L}_{\text{synth}}$ and $\mathcal{L}_{\text{real}}$ contribute equally to the total loss.

Negative results. We explored a range of ideas that ended up degrading or not affecting performance in our setting, and we briefly report them here. Our intention in reporting these results is to save time for future research and to give a more complete picture of our attempts. Note that these results are specific to our particular setup and are not meant to discourage potentially fruitful avenues of research.

1. Discriminator. We explored adding an additional adversarial loss [7, 14] to obtain more realistic details, specially from non-visible viewpoints. Nevertheless, the discriminator was leading to noise in geometry and incorrect wrinkles, and was often unstable during training. In practice, we obtained better results in geometry by just minimizing $\mathcal{L}_{\text{3D label}}$ on the real 3D scans.

2. Hessian Loss [23]. We explored the possibility of regularizing the predicted Signed Distance Field by encouraging small second-order derivatives with respect to input points, but this was highly memory consuming and increased training time. We also did not observe

significant changes with respect to regularizing first-order derivatives via an Eikonal loss [9].

3. Perceiver [11]. Since the transformer architecture is the memory bottleneck, we explored reducing the dimensionality of the Structured 3D Features before assigning features to query points. We found, however, that this was leading to very smooth geometry and texture. The intuition after this is that denser local features are better suited for 3D reconstruction of high-frequency details.

4. Canonicalization. Our initial attempts explored canonicalization as a way to simplify the learning problem and efficiently utilize the body prior by learning a single model in T-Pose. However, we observed that the canonicalization step is very sensitive to small GHUM errors, resulting in a lack of high frequencies like wrinkles in geometry or details in texture. Therefore we use our transformer in the posed space and map to canonical space only when re-posing, by using skinning weights after collecting per-point features.

5. 3D normal supervision on synthetic data. We explored having an additional loss function to minimize the difference between the predicted point normals and groundtruth 3D scan normals, evaluated on on-surface points. However, this led to smoother geometry, performing similarly to an additional Eikonal regularization. We tried unit normalizing $\nabla_{\boldsymbol{x}} s_{\boldsymbol{x}}$ before supervision to have this loss function specifically focused on the normal direction instead of the magnitude, though obtaining similar results.

6. Additional loss function to penalize the difference between imGHUM's SDF and our predicted SDF. This acts as a regularization for the learnt SDFs, but was tying the network excessively to the body and reduced its capacity to generate loose clothing or recover from noisy GHUM fits.

7. Pose-dependent SDF estimations. We briefly explored the possibility of having an additional MLP that takes GHUM pose/shape and predicts per-point deformation (both deforming query points or deforming feature points). The intuition after this is to try to learn additional pose-dependent effects within training, by also training on a significant pose diversity. However, the MLP converged to predict negligible displacements and we suspect that tackling this task effectively requires tailored data and objective functions.

2. Additional Results

In this section, we provide more analysis and results with respect to the experiments in the main paper. Figure 1 shows

Figure 1. Results of the full method (trained on both real and synthetic data) in comparison to a method trained only from real images. As observed in this example, training with no synthetic supervision leads to significant uncertainty in the z-axis during reconstruction. The body template is useful to regularize the process but the method becomes unable to generate realistic 3D reconstructions.

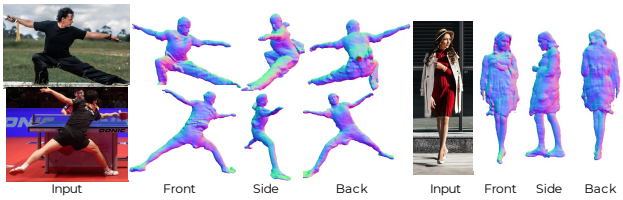


Figure 2. Examples showing the front, side and back views of the geometry reconstructed from our method. The proposed approach tackles extreme poses (left) or people with loose clothing (right).

a reconstruction of the full method trained on both real and synthetic data, in comparison to a method trained only on real images. We here show side views of the reconstructions to show the consistency of geometry in all views. In contrast, when training only on real images, the method is unable to properly solve the uncertainty along the Z-axis and often generates artifacts. Figure 2 depicts more results from the full method, including front, side and back views for different images. On the left, we show examples of extreme poses and the example on the right depicts a person with loose clothing. Note that monocular 3D reconstruction methods are inherently ambiguous w.r.t. depth, and the use of synthetic data and pose priors during pose estimation limits some of the ambiguities. Our multi-view extension further alleviates the problem.

Next, we extend the results presented in the main paper and provide more qualitative results on different tasks. We show more monocular 3D Human Reconstruction results in Figures 3, 4 and 5, showcasing a variety of input poses, backgrounds, viewpoints and clothing. We also categorize and present failure cases in Figure 6, presumably all due to inaccurate GHUM fits or limited training data. We expect that better GHUM fits and accurate segmentations would lead to more consistent results on the presented images. Loose clothing is not well represented in our training set, since the dataset of real images used for training [2] com-

prises challenging poses of diverse sports mostly on tight clothing.

We also show additional results of 3D Human relighting and re-posing in Figures 7 and 8, and extend the 3D virtual try-on experiments from the paper in Figures 9 and 10 with examples of upper-body and lower-body clothes respectively. The input images are shown in the left column and reference clothing is shown in the upper-row. Note that the transferred cloth textures are shaded consistently with the illumination from the original scenes, leading to photorealistic results that are coherent with the original non-edited 3D reconstruction. More examples are shown in the Supplementary Video, where we additionally show interpolations between cloth texture, body poses (animation) and scene illumination. For cloth try-on, the interpolations showcase a remarkable level of robustness even when interpolating cloth features with very different textures, obtained from diverse body poses or shapes.

Finally, we provide an intuition of the effect of the number of points/features in the results, in Figure 11. Increasing the number of points leads to sharper reconstructions, although it does not significantly affect metrics quantitatively. Our final model has 18K points sampled on the body surface, which shows a good balance between memory consumption during training and texture sharpness.

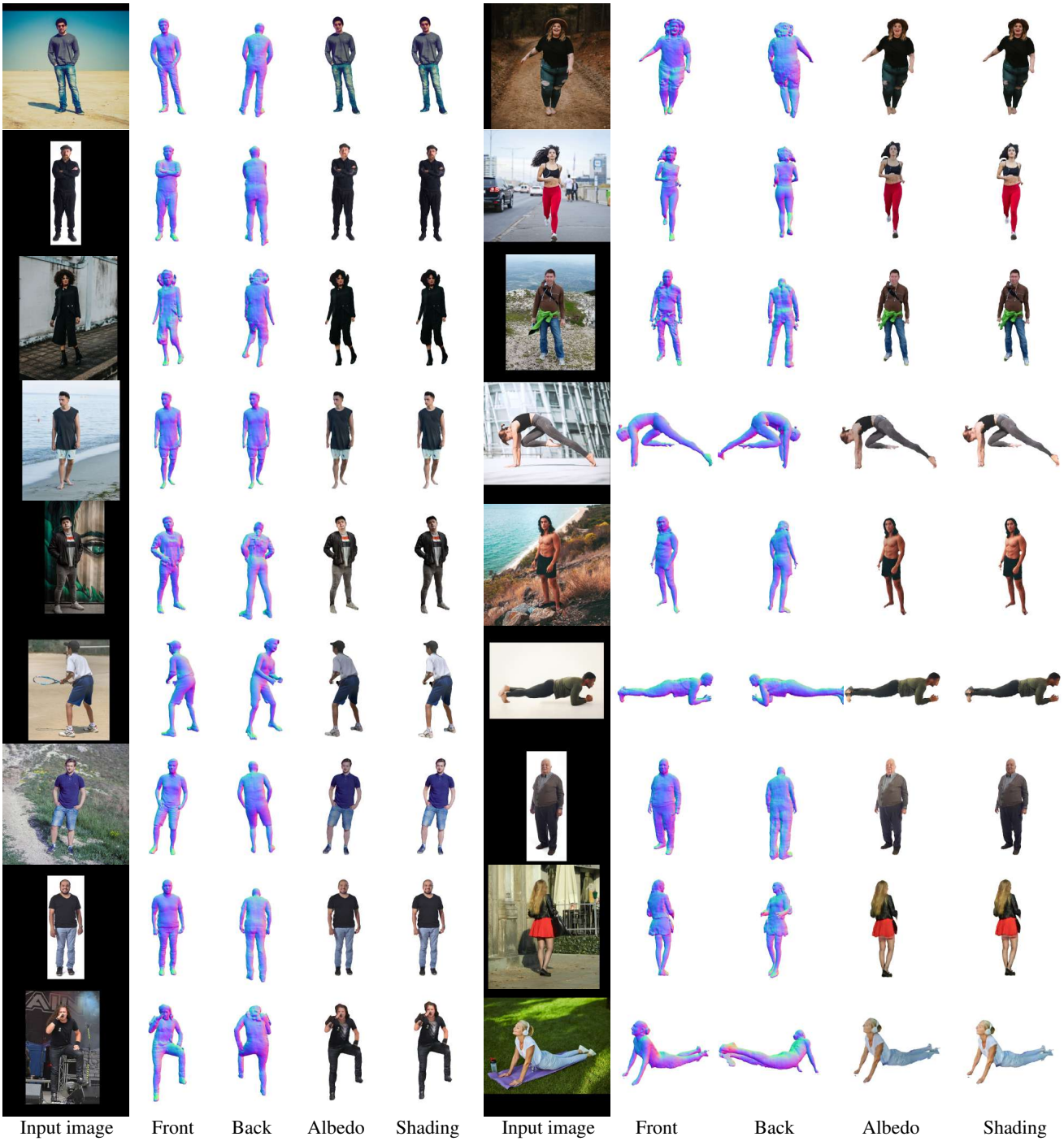


Figure 3. **Additional qualitative examples from our method**.

Figure 4. **Additional qualitative examples from our method**.

Figure 5. **Additional qualitative examples from our method**.

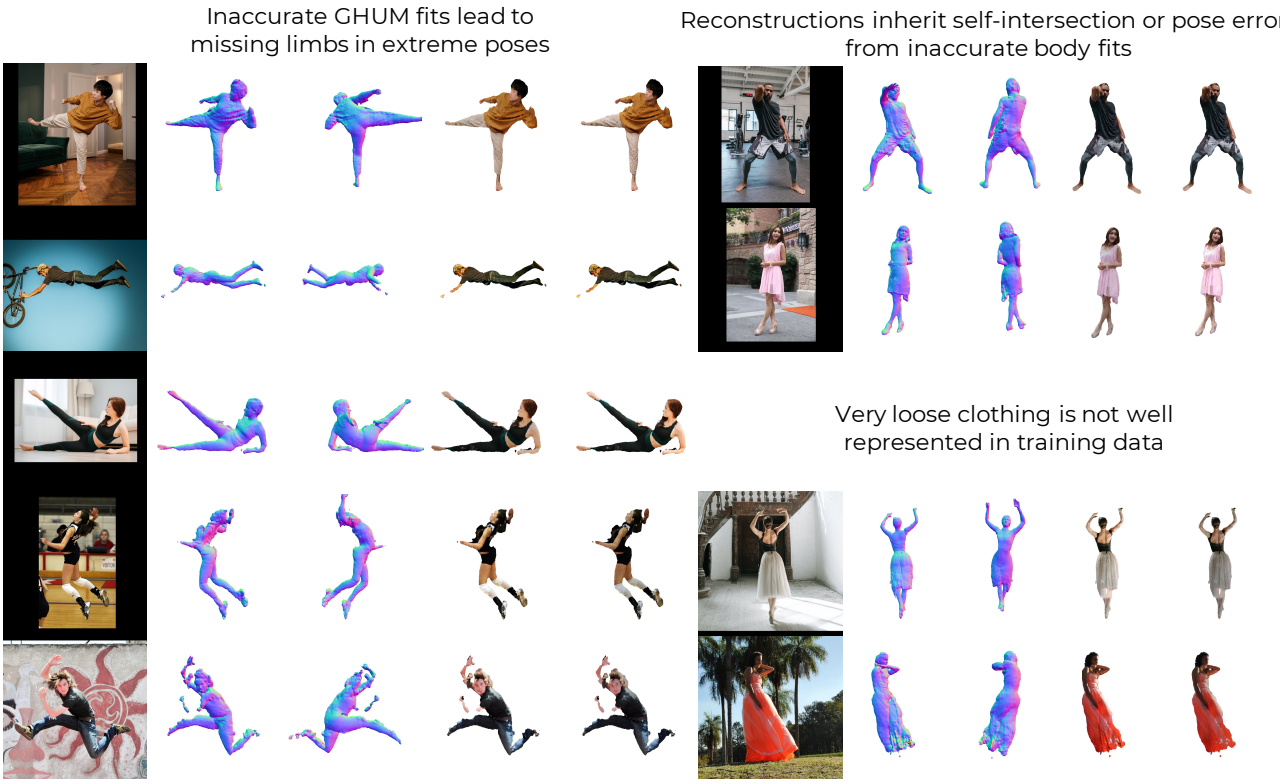
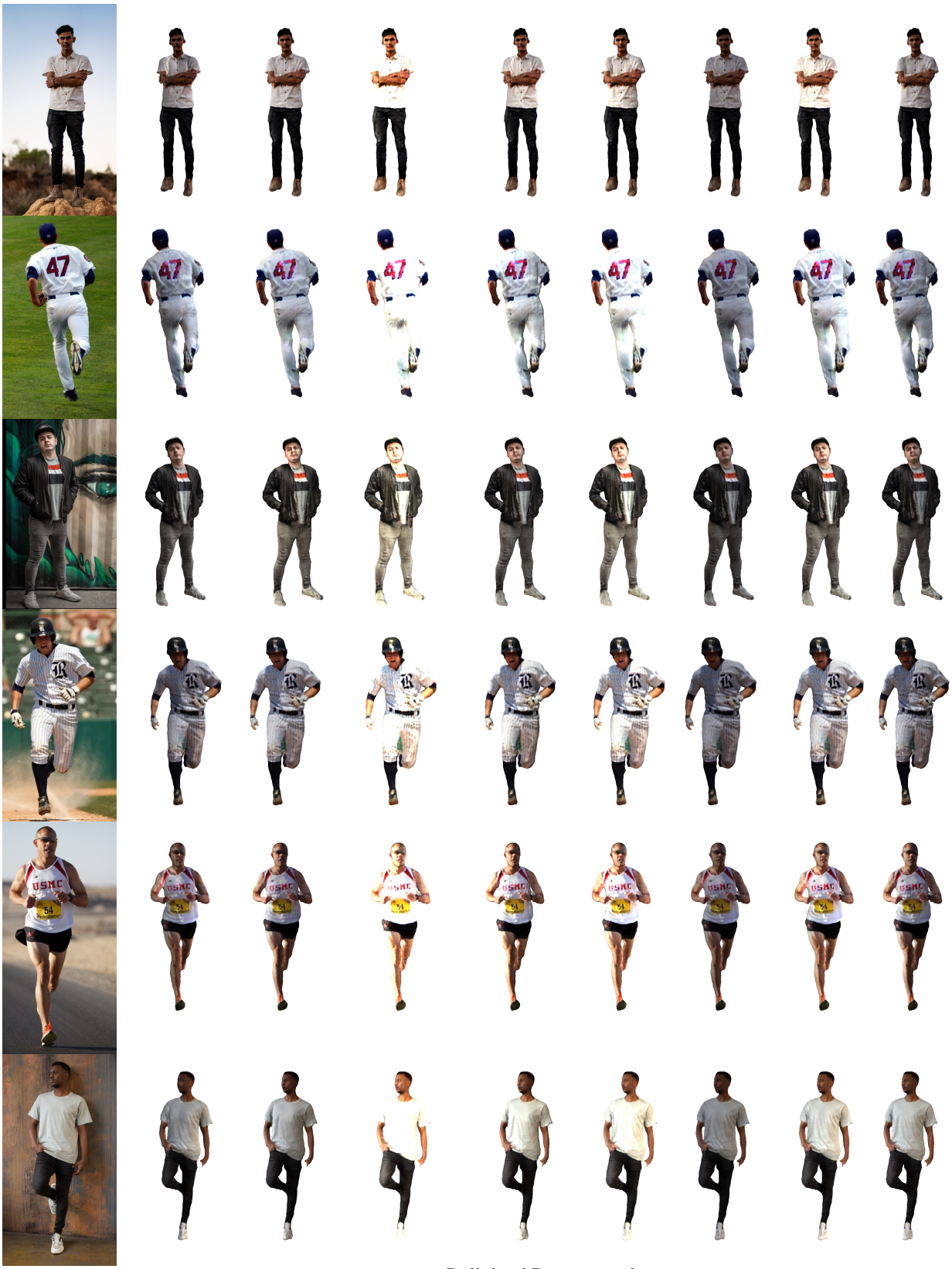


Figure 6. **Failure cases**. We categorize failure cases in three main classes, all of them presumably due to noisy GHUM fits or insufficient training data.



Input image

Relighted Reconstructions

Figure 7. **Qualitative results on 3D Human Relighting**.

Figure 8. **Qualitative results on animation of 3D reconstructions**.

Figure 9. **More examples of cloth texture transfer**. We extend the experiment from the main paper, showcasing an example of upper-body cloth try-on, and show the input images (left) and source clothing (upper-row). The 3D reconstructions look realistic and consistent accross all examples. Note that we only take one single image from both subject and clothing, and the network re-poses the affected S3Fs, allucinates occluded texture, and shades the clothing in the new scene.

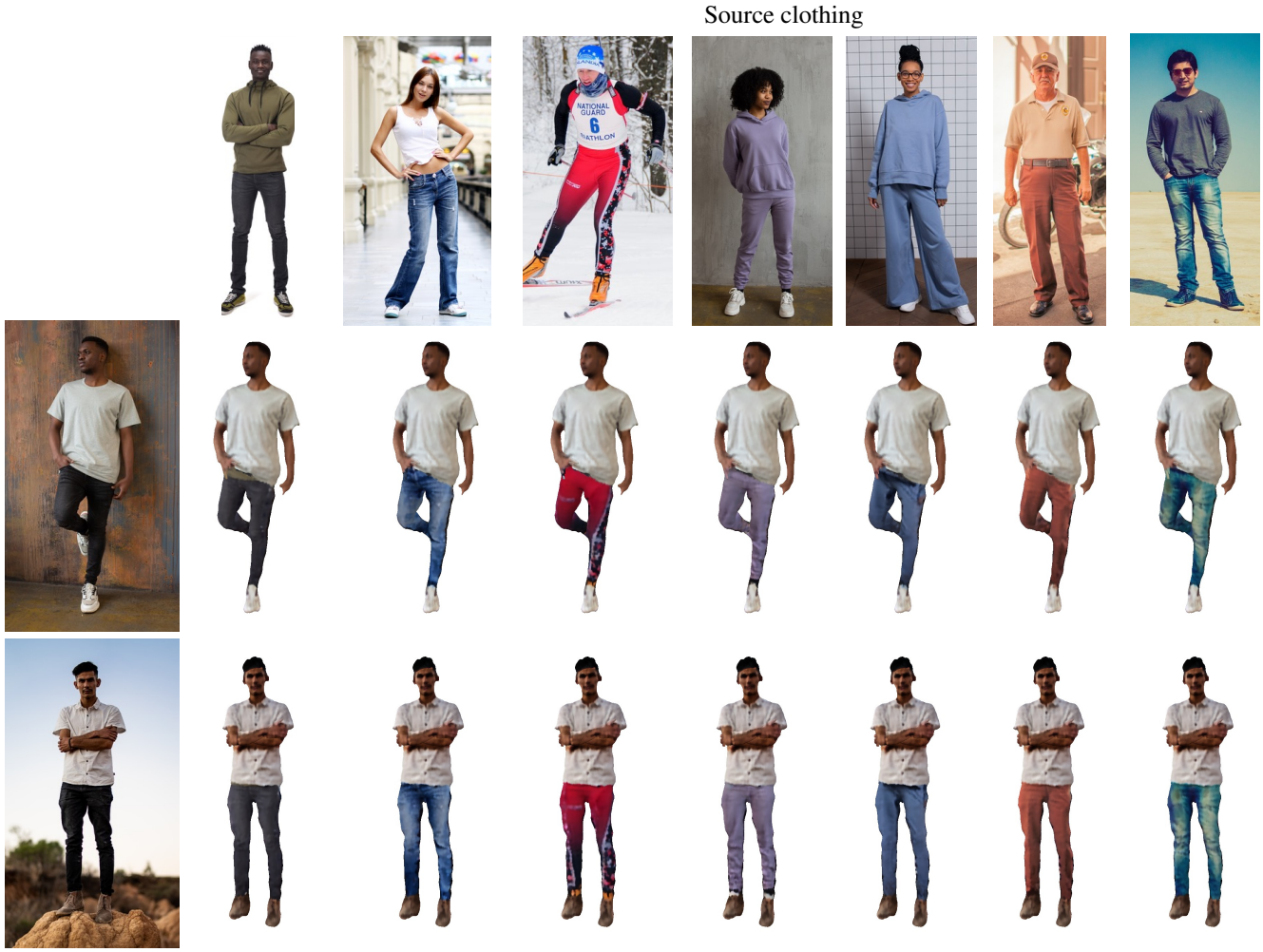


Figure 10. **More examples of cloth texture transfer**. This example features try-on examples from lower-body clothing. See the Supplementary Video for more examples.

Figure 11. **Qualitative ablation of number of points sampled in the body surface**, storing 3D Features. Our final model has 18000 points which provides the best tradeoff between GPU memory and sharpness. Lower number of points do not affect significantly quantitative results but are less capable of representing high-frequency details.

References

[1] Thiemo Alldieck, Mihai Zanfir, and Cristian Sminchisescu. Photorealistic monocular 3d reconstruction of humans wearing clothing. In *CVPR*, 2022. 2

[2] Eduard Gabriel Bazavan, Andrei Zanfir, Mihai Zanfir, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Hspace: Synthetic parametric humans animated in complex environments. *arXiv*, 2021. 1, 3

[3] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2020. 1

[4] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *ICCV*, 2017. 2

[5] Cmu graphics lab motion capture database. http://mocap.cs.cmu.edu/. 1

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. 2

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 2

[8] Ivan Grishchenko, Valentin Bazarevsky, Andrei Zanfir, Eduard Gabriel Bazavan, Mihai Zanfir, Richard Yee, Karthik Raveendran, Matsvei Zhdanovich, Matthias Grundmann, and Cristian Sminchisescu. Blazepose ghum holistic: Real-time 3d human landmarks and pose estimation. *arXiv*, 2022. 1

[9] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *ICML*, 2020. 3

[10] https://polyhaven.com/. 1

[11] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021. 3

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv*, 2014. 2

[13] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH*, 1987. 2

[14] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018. 2

[15] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. 2

[16] Renderpeople dataset. https://renderpeople.com/. 1

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1

[18] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *ICCV*, 2019. 2

[19] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *CVPR*, 2020. 2

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017. 2

[21] Hongyi Xu, Eduard Gabriel Bazavan, Andrei Zanfir, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Ghum & ghuml: Generative 3d human shape and articulated pose models. In *CVPR*, 2020. 1

[22] Andrei Zanfir, Eduard Gabriel Bazavan, Hongyi Xu, William T Freeman, Rahul Sukthankar, and Cristian Sminchisescu. Weakly supervised 3d human pose and shape reconstruction with normalizing flows. In *ECCV*, pages 465–481. Springer, 2020. 1

[23] Jingyang Zhang, Yao Yao, Shiwei Li, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. Critical regularizations for neural surface reconstruction in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6270–6279, 2022. 2