Supplementary Material for StylePeople: A Generative Model of Fullbody Human Avatars

Artur Grigorev [1,2*], Karim Iskakov [1*], Anastasia Ianina [1], Renat Bashirov [1], Ilya Zakharkin [1,2], Alexander Vakhitov [1], Victor Lempitsky [1,2]

[1] Samsung AI Center, Moscow
[2] Skolkovo Institute of Science and Technology, Moscow

1. Neural dressing

1.1. Model details

The Neural Dressing model consists of two elements: the neural texture and the renderer network.

The neural texture is a three-dimensional $512\times512\times16$ tensor. During training we store it as an array of severn mipmaps (from 8p to 512p), which allows us to regularize high-resolution mipmaps and encourage low-resolution ones to store more information (as was proposed in [18]).

The renderer network R is a standard U-net model from [19] with Resnet-16 encoder backbone. This network processes the neural texture warped onto the SMPL-X mesh along with additional two channels containing mesh with UV texture coordinates. The main part of the network outputs a 16-channel tensor which is then passed through two parallel shallow convolutional networks. The first of these networks generates the three-channel RGB image, while the second produces a three-channel segmentation mask. The first channel of the segmentation is the foreground mask, while the second and the third are hands and face masks respectively (which are used for additional supervision).

1.2. Supervision for video-based training

As we train our Neural Dressing model on high-resolution videos from *AzurePeople* dataset, it allows us to use several supervisory signals. Apart from standard **VGG19 perceptual loss** $\mathcal{L}_{vgg}$ between the output images and the ground truth frames, several other loss functions were used.

We have obtained three-channel segmentation maps (as described above) using Graphonomy network [4] and then refined it with Photoshop's chromakey feature. These segmentation maps M_3 are used to enforce rendering network to generate realistic foreground masks as well as to learn to distinquish face and hands regions for better visual quality of theese regions. For segmentation supervision, **Dice loss** is used:

$$\mathcal{L}_{segm} = -\log \frac{2\|M_3 \odot \hat{M}_3\|}{\|M_3\| + \|\hat{M}_3\|},$$

where $\hat{M}_3$ is the predicted segmentation and $\odot$ is Hadamard product.

To ensure realistic high-frequency details, a set of discriminators was used. The first discriminator distinguishes between real and fake samples of full images. The other two discriminators are given image crops around face and head regions respectively. The crops are calculated using OpenPose keypoint detector [3]. The cropped regions are also masked by respective channels in segmentation tensors. All discriminators employ PatchGAN structure with $l2$ adversarial losses:

$$\mathcal{L}^R_{\text{adv}} = \|1 - D_{\text{full}}(\hat{I})\|_2$$
$$\mathcal{L}^D_{\text{adv}} = \|D_{\text{full}}(\hat{I})\|_2 + \|1 - D_{\text{full}}(I)\|_2$$

$$\mathcal{L}^R_{\text{advface}} = \|1 - D_{\text{head}}(\mathcal{C}_{\text{head}}(\hat{I} \odot \hat{M}_{\text{head}}))\|_2$$
$$\mathcal{L}^D_{\text{advface}} = \|D_{\text{head}}(\mathcal{C}_{\text{head}}(\hat{I} \odot \hat{M}_{\text{head}}))\|_2 + \|1 - D_{\text{head}}(\mathcal{C}_{\text{head}}(I \odot M_{\text{head}})\|_2$$

$$\mathcal{L}^R_{\text{advhands}} = \|1 - D_{\text{hands}}(\mathcal{C}_{\text{hands}}(\hat{I} \odot \hat{M}_{\text{hands}}))\|_2$$
$$\mathcal{L}^D_{\text{advhands}} = \|D_{\text{hands}}(\mathcal{C}_{\text{hands}}(\hat{I} \odot \hat{M}_{\text{hands}}))\|_2 + \|1 - D_{\text{hands}}(\mathcal{C}_{\text{hands}}(I \odot M_{\text{hands}})\|_2$$

In the equations above, M_{hands} and M_{head} are the segmentation masks for hands and head regions respectively, ($\hat{M}_{\text{hands}}$ and $\hat{M}_{\text{hands}}$ correspond to the generated masks). $\mathcal{C}_{\text{hands}}$ and $\mathcal{C}_{\text{head}}$ are crop operators for hands and head regions. For hands, the crop for each hand is passed through

*equal contribution

the hand discriminator separately. I and $\hat{I}$ denote full ground truth and generated images respectively. To enhance face realism we also employ the **VGGFace perceptual loss** $\mathcal{L}_{\text{VGG Face}}$.

The last loss term regularizes the high-resolution texture mipmaps so that they store only high-frequency details:

$$\mathcal{L}_{\text{mipmap}} = \sum_{i=3}^{9} \alpha_i \|T^{(2^i)}\|_2,$$

where $T^{(2^i)}$ is a mipmap of resolution 2^i, and α_i has the following values: $\underset{i=\overline{3,9}}{\alpha} = \{0, 0, 0, 1, 2, 4, 8\}$

1.3. Training procedure

In the video-based setup, the model is trained in two phases. First, the renderer network is pretrained on all 56 people from *AzurePeople* dataset along with the corresponding neural textures. In this phase, the renderer is learning to convert 512p neural renders into 512p images.

	phase 1	phase 2
loss weights		
$\mathcal{L}_{vgg}$	1	1
$\mathcal{L}_{vggface}$	2e-1	2e-1
$\mathcal{L}_{segm}$	1e+2	1e+2
$\mathcal{L}_{adv}$	1e+1	1e+1
$\mathcal{L}_{advface}$	5e+1	5e+1
$\mathcal{L}_{advhands}$	5e+1	5e+1
$\mathcal{L}_{mipmap}$	1	1
learning rates		
Texture	5e-2	1e-3
R	1e-3	1e-3
$\mathcal{D}_{full}$	2e-4	2e-4
$\mathcal{D}_{head}$	2e-4	2e-4
$\mathcal{D}_{hands}$	2e-4	2e-4
other parameters		
R optim beta1	5e-1	5e-1
Texture optim beta1	5e-1	5e-1

Table 1. Hyperparameters for both phases of video-based neural dressing training procedure

In the second step, the network is finetuned for a single person (either from the same dataset or completely new one) at 1024p resolution, while the maximum resolution of neural texture mipmaps remains 512p. During both steps ADAM optimiser is used for all networks. Table 1 shows the list of hyperparameters used during both phases.

2. Generative Textures

2.1. Architecture details

Here we describe modifications to the Neural Dressing approach that allow us to sample human body neural texture from a generative model. In this setup we still have the rendering network R, while the stack of neural texture mipmaps is substituted by the generative network G.

The architecture of G closely follows the StyleGANv2 model [6]. The only modifications of the architecture of [6] are additional inputs to several convolutional layers of the network. We pass additional 16 channels of SMPL-X mesh vertices spectral coordinates alongside the outputs of previous layer. This modification affects layers receiving 64p, 128p and 256p feature maps, To acquire these maps, we calculate spectral coordinates for each SMPL-X mesh vertex and then rasterize them bilinearly in the UV texture space.

The generator G outputs 16-channel 256p feature maps, which are then warped onto the SMPL-X UV render and passed through the renderer R, which outputs three-channel RGB output along with one-channel foreground segmentation. The latter is then modified so that at every pixel on which the SMPL-X is projected, the value is set to one (i.e. the projection of the SMPL-X mesh is forced to belong to the foreground). This enforcement increases the stability of segmentation.

Apart from R and G, we use three discriminator networks: D_{unary}, D_{binary} and D_{face}. All of them follows the discriminator architecture from [6].

D_{unary} takes four-channel image (RGB+foreground segmentation) as an input. D_{binary} takes two such images of the same person in different poses stacked. D_{face} is given a four-channel 128p crop of the face region.

2.2. Loss functions

For our adversarial training procedure we use the loss from [6], adopting same non-saturating loss for discriminator outputs $\mathcal{L}_{adv\_unary}$, $\mathcal{L}_{adv\_binary}$ and $\mathcal{L}_{adv\_face}$ and the same regularization techniques, i.e. R1 regularization for discriminators $\mathcal{L}_{r1}$ and path regularization for generator $\mathcal{L}_{path}$.

We also add two new regularizations specific to our system described in the **Additional regularization** paragraph of the main paper.

$$\mathcal{L}_{wreg} = \|q_\xi(I) - \mathbf{w}\|^2$$
$$\mathcal{L}_{augreg} = \|\text{Tr}(f_\theta[J]) - f_\theta[\text{Tr}(J)]\|$$

2.3. Training procedure

The training of the generative model is also split in two phases. First, we train the pipeline to produce 256p images and then tune them for the 512p setup, while filtering out low resolution images from the training dataset. In both phases, the generator G has the same set of layers and outputs 256p neural textures, while the renderer R is given an additional upsampling and several convolutional layers during the switch of the phases. Therefore, the final version of R takes 256p neural renders and outputs 512p images. The discriminators D_{unary} and D_{binary} are also augmented with additional layers in the beginning so that they are able

to process 512p inputs. All additional layers are gradually (progressively) introduced into the training procedure following [5].

loss weights	
$\mathcal{L}_{adv\_unary}$	1
$\mathcal{L}_{adv\_binary}$	1
$\mathcal{L}_{adv\_face}$	1
$\mathcal{L}_{r1}$	1e+1
$\mathcal{L}_{path}$	2
$\mathcal{L}_{augreg}$	1
$\mathcal{L}_{wreg}$	1e+1
learning rates	
G	1e-3
R	1e-4
q_ξ	1e-3
D_{unary}	2e-3
D_{binary}	2e-3
D_{face}	2e-3

Table 2. Hyperparameters for generative texture model training

For all networks in the pipeline we also employ equalized learning rate technique introduced in [5]. Training hyperparameters are the same for both phases and are listed in Table 2

3. Details about other methods

We list the methods we compare against, and provide details about the (re)-implementations.

Textured Neural Avatars [15]. We use the models provided by the authors.

Videoavatars [2] We use the models provided by the authors.

360 degree [8]. Since the authors provide only a non-animatable mesh of a person in the A-pose, we have rendered the mesh under a visually close global rotation and then aligned the image with ground truth using an affine warp computed from the OpenPose detections.

Octopus [1]. We use the 3D model provided by the authors. We have rendered the images in the SMPL poses provided with the dataset. However due to scale mismatch the alignment with ground truth was poor, and we therefore applied affine warping based on OpenPose as for the previous baseline.

LWGAN [9] We use the network provided by the authors. Since the method does not produce a segmentation mask, we have replaced the color of the background pixels with white in the input images in order to maintain consistency in the comparison.

FOMM [16]. We use the networks provided by the authors that was trained on TaiChi dataset (we refer to this baseline as FOMM-TaiChi). Since our TEDX dataset differs significantly from TaiChi (stage performances vs. martial arts), we also train FOMM from scratch on the TEDX data (the corresponding baseline is called FOMM-TEDX). FOMM-TEDX baseline is trained on 41233 sequences each containing eight images.

Qualitative comparison to PIFu and PIFuHD. In addition to the comparisons in the main text, we compare qualitatively to PIFu [13] and PIFuHD [14]. We use the networks provided by the authors. We use them to create 3D models by one frontal image from PeopleSnapshot dataset and render them from new viewpoint for comparison. Since PIFuHD model does not produce full-body texture, we use normals of the 3D model for its' visualization.

We present comparison against state-of-the-art one-shot methods on three test sequences of the TEDXPeople dataset in Figure 1 and to PIFu and PIFuHD methods on two people from PeopleSnapshot dataset in Figure 1.

4. Inference details

4.1. Encoders.

The encoder architecture used in all our experiments (Fig. 3) is inspired by the pSp-architecture proposed in [11]. As a backbone we utilize EfficientNet-B7 [17] initialized with ImageNet [12] pretrained weigths. Intermediate feature maps are aggregated with the upscaled lower resolutions maps and then passed through a small *map2style* network comprising Conv2d $\rightarrow$ ReLU $\rightarrow$ Adaptive Average Pooling $\rightarrow$ Linear layers.

We train encoders via backpropagation with ADAM optimizer [7]. The learning rate is fixed to 0.001 during training.

4.2. Optimization.

The optimization part of the inference consists of four stages: (i) the optimization of the latent vector w, (ii) the optimization of the generator parameters h_ψ, (iii) the optimization of the noise tensors and (iv) the direct texture optimization. Each stage is defined by the number of iterations, the learning rate and the loss weights. Table 3 summarizes hyperparameters for each stage that we found to work best.

	Stage 1	Stage 2	Stage 3	Stage 4
Number of iterations	100	70	50	100
Learning rate	0.01	0.01	0.1	0.15
LPIPS	1.0	1.0	1.0	1.0
MSE	0.5	0.5	0.5	0.5
Encoder deviation MAE	0.1	0.0	0.0	0.0
Generator parameters deviation MAE	0.0	1.0	0.0	0.0
Texture deviation MAE	0.0	0.0	0.0	0.1
Face LPIPS	0.5	1.0	0.1	2.0
Face discriminator feature matching	0.0	2.0	0.0	3.0

Table 3. Optimization hyperparameters during inference. Stages: the optimization of latent vector w, the optimization of generator parameters h_ψ, the optimization of noise tensors and the direct optimization of texture.

Figure 1. Comparison against state-of-the-art one-shot methods on three test sequences of the TEDXPeople dataset.

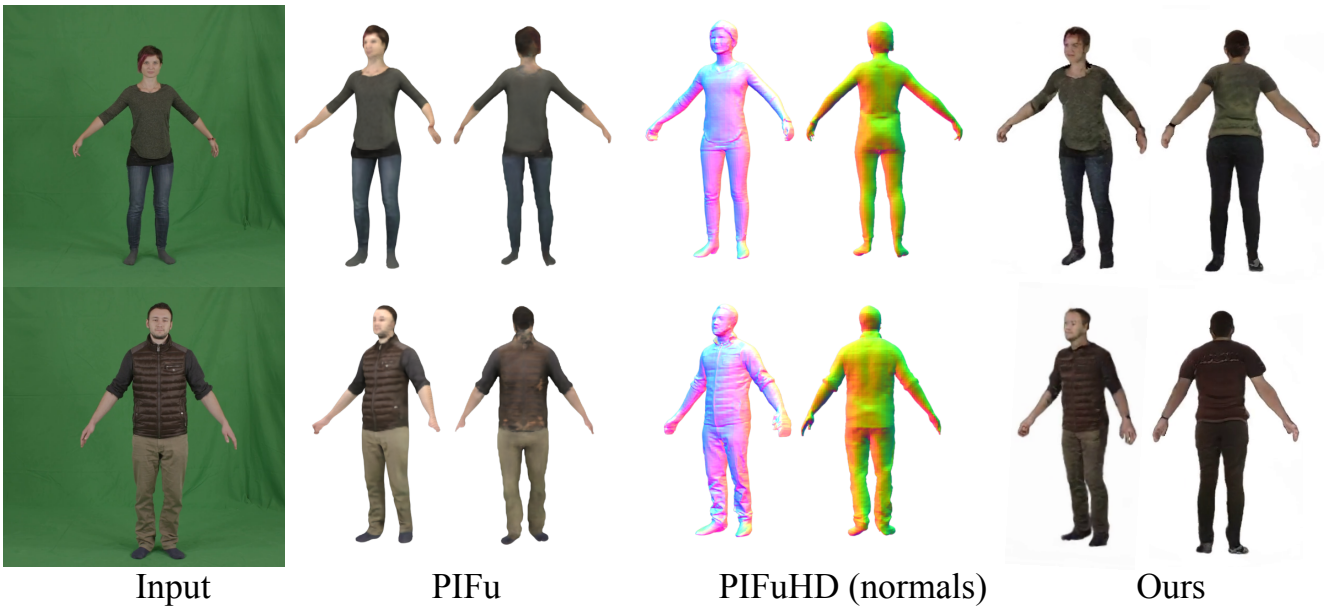

Figure 2. Comparison of our model with PIFu and PIFuHD in a one-shot mode. Normals are used for the visualization of PIFuHD since their system does not infer texture. Digital zoom is recommended. Note that unlike PIFu and PIFuHD, our model produces rigged avatar and does not require 3D models with clothing/hair during training.

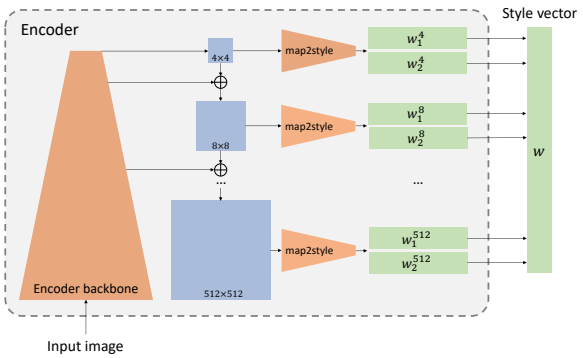

Figure 3. The encoder architecture inspired by [11]. The input image is passed through the encoder convolutional backbone to extract intermediate feature maps of different resolutions $(4\times4, 8\times8, \ldots, 512\times512)$. Each feature map except the last one is summed up with the upscaled map from the previous level and passed through a small *map2style* network to predict style vectors at the corresponding generator resolution.

	IS↑	FID↓	LPIPS↓	SSIM↑
G-encoder				
Ours (full)	**1.8324**	244.8	**0.0777**	0.9131
No MAE-texture	1.7768	**223.3**	0.0790	**0.9134**
No fm-face	1.8258	225.8	0.0780	0.9133
No LPIPS-face	1.8311	225.7	0.0794	0.9130
No MAE-generator	2.006	270.8	0.0803	0.9122
No MAE-encoder	1.8457	239.9	0.0797	0.9127
No MSE	1.8314	278.1	0.0809	0.9061
A-encoder				
Ours (full)	**1.8077**	231.3	**0.0785**	**0.9135**
No MAE-texture	1.7414	229.4	0.0794	0.9131
No fm-face	1.7758	**226.0**	0.0799	0.9125
No LPIPS-face	1.7177	234.4	0.0794	0.9134
No MAE-generator	1.7626	252.4	0.0812	0.9129
No MAE-encoder	1.6809	252.5	0.0820	0.9129
No MSE	1.7176	338.1	0.0833	0.9096

Table 4. Ablation study for inference module of neural textures (one-shot). We use two sequences of the People Snapshot dataset [2], taking the first frame as a source image and evaluating on the remaining frames of the sequences. See text for discussion.

5. Ablation Study

5.1. Inference ablations

We evaluate the contributions of different losses in the inference process. With each ablations we turn off one of the following losses: LPIPS-loss [20] between recovered image and input image, Mean Squared Error (MSE) between recovered image and input image, Mean Absolute Error (MAE) on latent variables w deviation from the initialization predicted by the encoder (MAE-encoder), MAE on generator parameters h_ψ deviation from the initial ones (MAE-generator), MAE on texture deviation from the texture values (MAE-texture), LPIPS-loss [20] on face regions of recovered and input images (LPIPS-face), feature matching loss based on trained face discriminator [10] (fm-face). We compare the ablations visually and quantitatively using four metrics: Inception Score (IS), Frechet Inception Distance (FID), Learned Perceptual Image Patch Similarity (LPIPS), structural similarity index measure (SSIM).

6. Additional experiment

Redressing. Our approach allows the generated people to try on clothes from the other models. Our model allows to replace the parts of a neural texture with parts from a different avatar, thus mixing visual appearance. The results of such a redressing for three randomly selected Azure people are shown in Figure 5. In each case, we create a hybrid model combining the head texture from one avatar and the non-head texture parts from the other avatar.

References

[1] Thiemo Alldieck, Marcus Magnor, Bharat Lal Bhatnagar, Christian Theobalt, and Gerard Pons-Moll. Learning to reconstruct people in clothing from a single rgb camera. In *Proc. CVPR*, pages 1175–1186, 2019.

[2] Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. Video based reconstruction of 3d people models. In *Proc. CVPR*, pages 8387–8397, 2018.

[3] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1812.08008*, 2018.

[4] Ke Gong, Yiming Gao, Xiaodan Liang, Xiaohui Shen, Meng Wang, and Liang Lin. Graphonomy: Universal human parsing via graph transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7450–7459, 2019.

[5] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[6] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. *CoRR*, abs/1912.04958, 2019.

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] Verica Lazova, Eldar Insafutdinov, and Gerard Pons-Moll. 360-degree textures of people in clothing from a single image. In *Proc. 3DV*, pages 643–653. IEEE, 2019.

[9] Wen Liu, Zhixin Piao, Jie Min, Wenhan Luo, Lin Ma, and Shenghua Gao. Liquid warping gan: A unified framework for human motion imitation, appearance transfer and novel view synthesis. In *Proc. ICCV*, pages 5904–5913, 2019.

[10] Xingang Pan, Xiaohang Zhan, Bo Dai, Dahua Lin, Chen Change Loy, and Ping Luo. Exploiting deep generative prior for versatile image restoration and manipulation. *Proc. ECCV*, 2020.

[11] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. *arXiv preprint arXiv:2008.00951*, 2020.

[12] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

[13] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2304–2314, 2019.

[14] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 84–93, 2020.

[15] Aliaksandra Shysheya, Egor Zakharov, Kara-Ali Aliev, Renat Bashirov, Egor Burkov, Karim Iskakov, Aleksei Ivakhnenko, Yury Malkov, Igor Pasechnik, Dmitry Ulyanov, et al. Textured neural avatars. In *Proc. CVPR*, pages 2387–2397, 2019.

[16] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. In *Proc. NeurIPS*, pages 7135–7145, 2019.

[17] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[18] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[19] Pavel Yakubovskiy. Segmentation models pytorch. https://github.com/qubvel/segmentation$_m odels.pytorch$, 2020.

[20] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. CVPR*, pages 586–595, 2018.

Figure 4. Ablation study for inference module of neural textures. Upper row: full model with all losses, in the following lines we are eliminating losses one-by-one following the order from Table 4.

Figure 5. **Neural redressing**. Given two avatars (left and middle), we create a hybrid ("redressed") avatar by taking the head neural texture from the first avatar and non-head neural texture from the second avatar.